The IT University of Copenhagen

Categorical and domain theoretic models of parametric polymorphism

Rasmus Ejlers Møgelberg

PhD Dissertation

Abstract

Parametric polymorphism in functional programming languages with explicit polymorphism is the property that polymorphic programs behave the same way at all type instantiations. This can be formulated more precisely using Reynold's notion of relational parametricity, which states that polymorphic functions should preserve relations. It has been known for a long time that parametric polymorphism can be used to encode inductive and coinductive data types, and this has been shown in a logic for parametricity suggested by Abadi and Plotkin.

In this dissertation we propose new category theoretic formulations of parametricity for models of the second-order lambda-calculus and models of a polymorphic lambda-calculus with linear function types and fixed points. These parametric models are models of Abadi and Plotkin's logic for parametricity, called parametric APL-structures and LAPL-structures, respectively. We show how that the encodings of inductive and coinductive types using parametric polymorphism give rise to initial algebras and final coalgebras in APL- and LAPL-structures and, using Plotkin's encodings, we show how to solve recursive domain equations in LAPL-structures.

Moreover, we show that the notions of APL- and LAPL-structures are general by constructing different examples. We construct a parametric APL-structure based on the per-model and a domain-theoretic parametric LAPL-structure. Based on recent work by Simpson and Rosolini we show how to construct parametric LAPL-structures using synthetic domain theory, and we device general ways of constructing parametric LAPL- and APL-structures using parametric completion processes.

Using the LAPL-structure constructed using synthetic domain theory we prove consequences of parametricity for a variant of the Lily programming language.

Acknowledgments

I would like to thank my supervisor Lars Birkedal for participating actively in my research over the past three years, and in particular for (almost) always finding time to see me even though he is a busy man. Also thanks to Alex Simpson, whom I visited in Edinburgh for the spring of 2004, for his help with both practical and mathematical matters and for many interesting discussions, and to Pino Rosolini for many helpful discussions and for inviting me to come to Genoa for a short visit.

I would also like to thank the following people for helpful discussions: Carsten Butz, Martin Hyland, Milly Maietti, Andy Pitts, John Reynolds, Ivar Rummelhoff, Thomas Streicher and Phil Wadler.

Finally thanks to everyone at the Department of Theoretical Computer Science at ITU, in particular Rasmus Lerchedahl Petersen for valuable collaboration. Also thanks to Noah Torp-Smith and Bodil Biering for proof-reading, and to everyone who participated in the topos theory seminar.

Contents

Introduction

This PhD dissertation is a collection of five papers on models of parametric polymorphism, which we shall refer to as Paper 1, etc. in this introduction. The introduction at hand is organized as follows: Sections 1, 2 contain background material on parametric polymorphism, Section 3 discusses models of parametric polymorphism, Section 4 gives a summary of the results of this dissertation, Section 5 discusses related work, Section 6 contains an overview of the papers in this dissertation, and Section 7 concludes and discusses future work.

1 Parametric Polymorphism

Polymorphism in typed programming languages enables the programmer to write functions that can act on input of many types. Consider for example the function rev that reverses a list. This function can act on integer-lists, string-lists or lists of any type. In languages with explicit polymorphism, such as ML and the second-order lambda calculus, the function rev will have the type (in the syntax of the second-order lambda calculus)

$$\prod \alpha \colon \mathsf{Type}.\, \mathrm{lists}(\alpha) \to \mathrm{lists}(\alpha),$$

to be read as "for all types α, $\mathrm{lists}(\alpha)$ to $\mathrm{lists}(\alpha)$". An element of this type is a family, indexed over types A, of functions taking A-lists and returning A-lists.

Christopher Strachey [37] identified two types of polymorphism. The first, called *ad-hoc polymorphism*, allows the behavior of a polymorphic function to depend on the type of in-data. The second type, called *parametric polymorphism*, only includes functions based on a common algorithm for all input types. For example rev is parametric, whereas the function that adds one to each element of an integer list, but is the identity on lists of all other types is ad-hoc.

A programming language is said to have *parametric polymorphism*, if it has explicit polymorphism and all polymorphic programs are parametric. In the following we sketch two reasons why such programming languages are interesting. We argue informally and use the syntax of the second order lambda calculus, but the arguments are not limited to the second-order lambda calculus.

1.1 Encoding of inductive and coinductive types

Consider the type

$$\prod \alpha \colon \mathsf{Type}.\, (\alpha \to \alpha) \to (\alpha \to \alpha)$$

in a language with parametric polymorphism. A function of this type takes for any type A a function $f \colon A \to A$ and produces a new function $A \to A$. For each natural number n, we can define the function that maps f to f^n (f^0 is the identity on A), and this way we can think of the type $\prod \alpha \colon \mathsf{Type}.\, (\alpha \to \alpha) \to (\alpha \to \alpha)$ as containing a copy of the natural numbers.

Since a parametric function of the type $\prod \alpha \colon \mathsf{Type}.\, (\alpha \to \alpha) \to (\alpha \to \alpha)$ is not allowed to use specific information about the type A, the only access it has to A is the function f, and so intuitively all it can do is map f to f^n[1]. Since parametric functions should use the same algorithm for all types of input, this n should be the same for all types A, and all functions f.

The above establishes the intuition why the type $\prod \alpha\colon \mathsf{Type}.\, (\alpha \to \alpha) \to (\alpha \to \alpha)$ in a language with parametric polymorphism can be used as a reasonable type of natural numbers. Of course we have not given a formal argument for this, and we have not defined what we mean by a reasonable type of natural numbers. Notice that the natural numbers were always present in the type, and we used parametricity to argue that no other elements of the type could exist. In general, encoding of all inductive and coinductive types such as finite lists, potentially infinite lists, trees etc. exist.

1.2 Data abstraction

In this section we will assume we are working in a language with parametric polymorphism and data types for natural numbers Nat, products and lists. This is not an unreasonable assumption as these data types can be encoded in languages with parametric polymorphism as described in Section 1.1.

Suppose that a programmer is writing a program for which he needs to use a data type for stacks of natural numbers, which should be implemented by another programmer. Such a data type would have operations

$$\begin{gathered}\mathrm{new}\colon \mathrm{Stack}\\ \mathrm{push}\colon \mathrm{Nat} \times \mathrm{Stack} \to \mathrm{Stack}\\ \mathrm{pop}\colon \mathrm{Stack} \to \mathrm{Stack}\\ \mathrm{top}\colon \mathrm{Stack} \to \mathrm{Nat}\end{gathered}$$

where new creates a new stack, push pushes numbers onto the stack, pop pops the number on top of the stack, and top returns the number on top of the stack. A concrete implementation of the type Stack could for example implement it using lists, with new being the empty list, push adding a new element to the first position of the list, pop taking the first element out of the list, and top returning the first number in a list.

Even though the programmer may not have the implementation of the type Stack yet, he can still write his program as a function P taking as input a concrete implementation of Stack. If for example the program should return a natural number, P would have the type

$$\prod \mathrm{Stack}\colon \mathsf{Type}.\, \mathrm{Stack} \to (\mathrm{Nat} \times \mathrm{Stack} \to \mathrm{Stack}) \to (\mathrm{Stack} \to \mathrm{Stack}) \to (\mathrm{Stack} \to \mathrm{Nat}) \to \mathrm{Nat}.$$

P then takes as input a concrete type and concrete operations.

Since the program P is parametric, it should only be able to access the type Stack through the operations $\mathrm{new}, \mathrm{pop}, \mathrm{push}, \mathrm{top}$ provided, since this is the only available information about the type Stack, and it should never be able to use information about a specific implementation of the type it is instantiated with.

We can use this to prove that if two concrete implementations of the type Stack behave the same way with respect to the interface operations $\mathrm{new}, \mathrm{push}, \mathrm{pop}, \mathrm{top}$ then the result of P instantiated with either of the two concrete implementations will be the same. This is a way of ensuring robust modularized programming.

Existential types present a different approach to data abstraction [20]. Existential types can be encoded using parametric polymorphism.

Data abstraction can be seen as a sort of information hiding; we hide information about the specific implementation of a data type from the programmer using the data type. Parametricity has also been used to implement other forms of information hiding such as hiding local variables from called procedures in imperative languages (see Section 5.4).

[1] In this example, we have assumed that the polymorphic language does not have fixed points. If the language has fixed points, the situation is different, as we describe in Section 2.2

1.3 Relational parametricity

Of course the arguments above are quite informal, since we have not formulated the concept of parametric polymorphism very precisely. John Reynolds has given a precise formulation of parametricity called *relational parametricity* [30]. The basic idea is that the parametric elements of a polymorphic type are those that preserve relations. For example, a polymorphic function f of type $\prod \alpha \colon \mathsf{Type}.\, \alpha \to \alpha$ is parametric if for all pairs of types A, B and all relations R between them: if $x \colon A, y \colon B$ are related in R, then so are $f(x)$ and $f(y)$.

Let me sketch how this captures data abstraction. We can express the notion of two implementations of Stack behaving the same way with respect to the interface operations using relations as follows: There should be a relation relating elements of the first implementation of Stack to elements of the other, such that the interface operations preserve the relations. This means that the stacks created by the two new operations should be related, pushing the same number onto related stacks should produce related stacks, popping related stacks should produce related stacks and top maps related stacks to equal numbers. Relational parametricity states that the program P of Section 1.2 applied to related implementations of Stack should produce related results, which, since the type of Stack does not occur in the result type of P should mean that the results are equal.

Martín Abadi and Gordon Plotkin have devised a logic for reasoning about parametricity for the second-order lambda calculus [29]. In this logic one can prove correctness of encoding of inductive and coinductive types from parametricity.

Of course, to use relational parametricity in practice for a specific programming language, one will have to specify what is meant by relations.

2 Models of Polymorphism

In this section we sketch the two polymorphic languages we consider in this dissertation, namely the second-order lambda calculus and PILL_Y (Polymorphic Intuitionistic / Linear Lambda calculus with fixed point combinator Y). We also sketch the categorical notions of models for these languages. The purpose of this section is not to give precise definitions, but to give an idea of the models used, to prepare for the discussion of parametric models of these calculi.

2.1 The second-order lambda calculus

The second-order lambda calculus (λ_2) is the simply typed lambda calculus (with products) extended with (impredicative) polymorphism. Types are given by the grammar

$$\sigma ::= \alpha \mid \sigma \to \sigma \mid \sigma \times \sigma \mid 1 \mid \prod \alpha.\, \sigma$$

where α ranges over an infinite set of type variables. The construction $\prod \alpha.\, \sigma$ binds the type variable α. We use σ, τ, ω to range over types. Terms are given by the grammar

$$t ::= x \mid \lambda x \colon \sigma.t \mid t(t) \mid \langle t, t \rangle \mid \pi t \mid \pi' t \mid \Lambda \alpha \colon \mathsf{Type}.\, t \mid t(\sigma) \mid \star.$$

Terms exist in contexts of free type variables and ordinary variables written as

$$\alpha_1, \ldots, \alpha_n \mid x_1 \colon \sigma_1, \ldots, x_m \colon \sigma_m \vdash t \colon \tau$$

where the free type variables of the σ_i and τ are among $\alpha_1, \ldots, \alpha_n$. We will often write Ξ for $\alpha_1, \ldots, \alpha_n$ and Γ for $x_1 \colon \sigma_1, \ldots, x_m \colon\ \sigma_m$, and we shall often omit the $:$ Type in types and terms. Most of the typing rules are as in the simple typed lambda calculus, so we just mention the two related to polymorphism.

If

$$\alpha_1, \ldots, \alpha_n, \alpha_{n+1} \mid x_1 \colon \sigma_1, \ldots, x_m \colon\ \sigma_m \vdash t \colon \tau$$

is a term and α_{n+1} is not free in any of the types $\sigma_1, \ldots \sigma_m$ then

$$\alpha_1, \ldots, \alpha_n \mid x_1 \colon \sigma_1, \ldots, x_m \colon\ \sigma_m \vdash \Lambda\alpha_{n+1}.\, t \colon\ \prod \alpha_{n+1}.\, \tau.$$

If $\Xi \mid \Gamma \vdash t \colon\ \prod \alpha.\, \tau$, and σ is a type with all free variables in Ξ, we may form $\Xi \mid \Gamma \vdash t(\sigma) \colon \tau[\sigma/\alpha]$, where $\tau[\sigma/\alpha]$ denotes capture free substitution of σ for free appearances of α in τ defined as usual.

We notice two properties of λ_2. First, for every collection of free type variables Ξ we have a simple typed lambda calculus of terms with free type variables in Ξ. Second, λ_2 has a very strong notion of polymorphism called impredicative polymorphism, meaning that terms of polymorphic types may be instantiated at all types. If for example t is a term of type $\prod \alpha.\, \alpha$, then $t(\prod \alpha.\, \alpha)$ also has type $\prod \alpha.\, \alpha$, and so applying a polymorphic term to a type need not result in a term with a simpler type. Impredicativity is what has made models of λ_2 difficult to find.

For a long time it was hoped that one could find set-theoretic models of λ_2. By this we mean models based on a set or class of sets U such that one can model types with n free variables as maps $U^n \to U$, and model product types and exponent types pointwise using set theoretic products and exponents. In fact Reynolds defined parametric polymorphism [30] hoping that such set theoretic models could be constructed using parametric polymorphism in the interpretation of polymorphic types.

In 1984 Reynolds [31] (see also [32]) showed that set theoretic models of λ_2 can not exist unless they are trivial. However, if one replaces set theory with other more constructive universes, such as certain toposes, models as described above may exist [26, 24].

The most famous example of such a model is the per-model, which can be seen as a set-theoretic model living inside the effective topos, or the quasi-topos of assemblies. The per-model is based on the set **Per** of partial equivalence relations on the natural numbers (symmetric, transitive, but not necessarily reflexive relations). A type with n free variables is modeled by a map

$$\mathbf{Per}^n \to \mathbf{Per}.$$

Exponents are modeled pointwise by defining for each pair of pers R, S a per $R \to S$ relating n, m if

$$\forall x, y \colon \mathbb{N}.\, R(x, y) \supset n \cdot x \downarrow \wedge m \cdot y \downarrow \wedge S(n \cdot x, m \cdot y)$$

where $n \cdot x$ denotes Kleene application, i.e., application of the n'th partial recursive function to x. Finally, if $f \colon\ \mathbf{Per}^{n+1} \to \mathbf{Per}$ is a type, we model the polymorphic type obtained by abstracting the last type variable by intersection, i.e., if $R_1, \ldots, R_n$ are pers then $(\prod f)(R_1, \ldots R_n)(n, m)$ holds iff

$$\forall R_{n+1} \in \mathbf{Per}.\, f(R_1, \ldots, R_{n+1})(n, m)$$

holds.

Terms of the form

$$\alpha_1, \ldots, \alpha_n \mid x \colon \sigma \vdash t \colon \tau$$

are modeled as families of morphisms

$$(\llbracket \vec{\alpha} \mid x \colon \sigma \vdash t \colon \tau \rrbracket(\vec{R}) \colon \mathbb{N}/\llbracket \vec{\alpha} \vdash \sigma \rrbracket(\vec{R}) \to \mathbb{N}/\llbracket \vec{\alpha} \vdash \tau \rrbracket(\vec{R}))_{\vec{R} \in \mathbf{Per}^n},$$

where $\mathbb{N}/[\![\vec{\alpha} \vdash \sigma]\!](\vec{R})$ denotes the set of equivalence classes of the partial equivalence relation $[\![\vec{\alpha} \vdash \sigma]\!]$, such that $[\![t]\!]$ is uniformly tracked, i.e., there exists a natural number n such that for all $\vec{R}$, $[\![\vec{\alpha} \mid x \colon \sigma \vdash t \colon \tau]\!](\vec{R})$ is given by $[m]_{[\![\vec{\alpha} \vdash \sigma]\!](\vec{R})} \mapsto [n \cdot m]_{[\![\vec{\alpha} \vdash \tau]\!](\vec{R})}$.

In general, second-order lambda calculus is modeled in λ_2-*fibrations*. These are defined to be fibred cartesian closed fibrations, with cartesian base and a generic object and simple products. We sketch what this means, but choose for simplicity to describe split fibrations and split generic objects. The reader interested in further details should consult [15].

Suppose $p \colon \mathbb{E} \to \mathbb{B}$ is a functor. For each object $\Xi \in \mathbb{B}$ we can consider the fibre $\mathbb{E}_\Xi$ of $\mathbb{E}$ over Ξ, defined to be the subcategory of $\mathbb{E}$ on objects mapped to Ξ via p and morphisms mapped to the identity on Ξ. A (split) fibration is a functor $p \colon \mathbb{E} \to \mathbb{B}$ satisfying a technical condition basically ensuring that every morphism $f \colon \Xi \to \Xi'$ in $\mathbb{B}$ induces a functor $f^* \colon \mathbb{E}_{\Xi'} \to \mathbb{E}_\Xi$, and further $(f \circ g)^* = g^* \circ f^*$ and $id^* = id$. The categories $\mathbb{E}$ and $\mathbb{B}$ are called the *total category* and *base category* respectively and a functor of the form f^* is called a *reindexing functor*.

A *fibred cartesian closed* fibration has cartesian closed fibres, and this structure is preserved by reindexing functors. A λ_2-fibration further has products in the base category and a (split) *generic object*, i.e., an object $\Omega \in \mathbb{B}$ such that for any $\Xi \in \mathbb{B}$ there exists a bijective correspondence between maps $\Xi \to \Omega$ in $\mathbb{B}$ and objects of $\mathbb{E}_\Xi$. This correspondence should be natural in Ξ in the sense that if $f \colon \Xi \to \Omega$ corresponds to $X \in \mathbb{E}_\Xi$ and $g \colon \Xi' \to \Xi$, then fg corresponds to $g^*X \in \mathbb{E}_{\Xi'}$.

Finally a λ_2-fibration is required to have *simple products* with respect to projections of the form $\pi \colon \Xi \times \Omega \to \Xi$. This means that for each such π, the reindexing functor

$$\pi^* \colon \mathbb{E}_\Xi \to \mathbb{E}_{\Xi \times \Omega}$$

is required to have a right adjoint $\prod_\pi$.

We model λ_2 in λ_2-fibrations as follows. Types with n free variables are modeled in the fibre category $\mathbb{E}_{\Omega^n}$ and terms with n free type variables

$$\alpha_1, \ldots, \alpha_n \mid x_1 \colon \sigma_1, \ldots, x_m \colon \sigma_m \vdash t \colon \tau$$

are modeled as maps in $\mathbb{E}_{\Omega^n}$ from $\prod_i [\![\vec{\alpha} \vdash \sigma_i]\!]$ to $[\![\vec{\alpha} \vdash \tau]\!]$, where $\prod_i$ denotes product in the fibre. Since the generic object induces a correspondence between maps $\Omega^n \to \Omega$ in $\mathbb{B}$ and objects $\mathbb{E}_{\Omega^n}$ we can model $\alpha_1, \ldots \alpha_n \vdash \alpha_i$ as the object corresponding to the i'th projection. The simple type constructions are modeled using the cartesian closed structure of $\mathbb{E}_{\Omega^n}$, and polymorphic types $\vec{\alpha} \vdash \prod \alpha_{n+1}.\, \sigma$ are modeled as

$$\textstyle\prod_\pi [\![\vec{\alpha}, \alpha_{n+1} \vdash \sigma]\!]$$

where $\pi \colon \Omega^n \times \Omega \to \Omega^n$ is the projection.

The per-model can be seen as a λ_2-fibration as follows. The base category has as objects natural numbers, and as morphisms from n to m set theoretic maps $\mathbf{Per}^n \to \mathbf{Per}^m$. The total category has as objects maps $f \colon \mathbf{Per}^n \to \mathbf{Per}$ for some n, and a morphism from $f \colon \mathbf{Per}^n \to \mathbf{Per}$ to $g \colon \mathbf{Per}^m \to \mathbf{Per}$ is a pair (h, k) such that $h \colon \mathbf{Per}^n \to \mathbf{Per}^m$ is a map, and k is an indexed family of maps

$$(k(\vec{R}) \colon \mathbb{N}/f(\vec{R}) \to \mathbb{N}/g \circ h(\vec{R}))_{\vec{R} \in \mathbf{Per}^n}$$

with a uniform tracker as defined above. The fibration maps an object $f \colon \mathbf{Per}^n \to \mathbf{Per}$ to n and a morphism (h, k) to h.

Modeling λ_2 in this fibration gives the per-model described above. Since types and terms with n free variables are modeled in the fibre over n, types are modeled as maps $\mathbf{Per}^n \to \mathbf{Per}$ and terms $\vec{\alpha} \mid x\colon \sigma \vdash t\colon \tau$ are modeled as vertical maps, i.e., families of maps of the form

$$(\llbracket \vec{\alpha} \vdash x\colon \sigma \vdash t\colon \tau \rrbracket(\vec{R})\colon \mathbb{N}/\llbracket \vec{\alpha} \vdash \sigma \rrbracket(\vec{R}) \to \mathbb{N}/\llbracket \vec{\alpha} \vdash \tau \rrbracket(\vec{R}))_{\vec{R}}$$

with a uniform tracker.

2.2 Adding fixed points

The second-order lambda calculus is a strongly normalizing language, and so does not have very strong computational power. To study a more expressive language we would like to add fixed points to the language, but since parametricity should give encodings of sum types, one can show, using a general result from [14], that adding fixed points to parametric λ_2 causes inconsistencies.

One way to deal with this problem is to think of the domain theoretic models. The category of cpos with continuous maps has a fixed point combinator, and is cartesian closed. It does not have coproducts, but the category of cpos with strict continuous maps does. Based on this observation, Gordon Plotkin [28, 27] suggested to study a polymorphic calculus in which one could distinguish between strict and non-strict maps. The encoding of sum types using parametricity would then work in the category of strict maps.

Gordon Plotkin also realized that in this language the encoding of inductive and coinductive types using parametricity could be generalized to an encoding of recursive types, such as types satisfying $A \cong [A \to A]$, where the isomorphism is in the category of strict maps. This means that this language can be considered an alternative approach to axiomatic domain theory, where the mentioned encoding of recursive types replaces the well-known limit-colimit construction.

We now sketch the language suggested by Plotkin. The language is called PILL$_Y$ and is an extension of DILL [3] with polymorphism and a fixed point combinator.

The grammar for types of PILL$_Y$ is

$$\sigma ::= \alpha \mid I \mid \sigma \otimes \sigma \mid \sigma \multimap \sigma \mid !\sigma \mid \prod \alpha.\, \sigma$$

where α ranges over an infinite set of type variables. The type constructor $\multimap$ gives linear function types. The grammar for terms is

$$\begin{array}{rcl} t & ::= & x \mid \star \mid Y \mid \lambda^{\circ} x\colon \sigma.t \mid t\, t \mid t \otimes t \mid !t \mid \Lambda\alpha\colon \mathsf{Type}.\, t \mid t(\sigma) \mid \\ & & \text{let } x\colon \sigma \otimes y\colon \tau \text{ be } t \text{ in } t \mid \text{let } !x\colon \sigma \text{ be } t \text{ in } t \mid \text{let } \star \text{ be } t \text{ in } t. \end{array}$$

Terms of PILL$_Y$ are written as

$$\vec{\alpha} \mid x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n; y_1\colon \tau_1, \ldots, y_m\colon \tau_m \vdash t\colon \omega.$$

The α's are type variables as in λ_2, the x_i's are intuitionistic variables and the y_j's are linear variables which can only occur linearly in t. The λ-abstraction $\lambda^{\circ} x\colon \sigma.\, t$ produces terms of linear function type $\sigma \multimap \tau$, and since linear variables of type $!\sigma$ behave as intuitionistic variables of type σ, we may define a type of ordinary functions $\sigma \to \tau = !\sigma \multimap \tau$. The fixed point combinator Y has type $\prod \alpha\colon \mathsf{Type}.\, (\alpha \to \alpha) \to \alpha$.

The encoding of inductive and coinductive data types in PILL$_Y$ is different from that of λ_2. For example the type of natural numbers can be encoded as

$$\prod \alpha.\, (\alpha \multimap \alpha) \to (\alpha \multimap \alpha).$$

For further details on PILL$_Y$ we refer to Paper 2.

We derive the notion of models of PILL$_Y$ from the models of DILL [3, 17]. A model of DILL is a symmetric monoidal adjunction

$$\begin{tikzcd}\mathbb{C} \arrow[r, bend right] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right]\end{tikzcd}$$

such that $\mathbb{C}$ is symmetric monoidal closed, $\mathbb{D}$ is cartesian, and $\mathbb{D}$ is the category of finite products of coalgebras for the comonad on $\mathbb{C}$ induced by the adjunction (see Paper 3 for an explanation of these concepts).

A PILL$_Y$-model is a fibred symmetric monoidal adjunction

$$\begin{tikzcd}\mathbb{C} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{D} \arrow[ll, bend right, "F"'] \arrow[dl] \\ & \mathbb{B} & \end{tikzcd}$$

(basically a family of symmetric monoidal adjunctions between fibre categories, with all structure commuting with reindexing) such that $\mathbb{C}$ is fibred symmetric monoidal closed, $\mathbb{D}$ is fibred cartesian, and $\mathbb{D}$ is the category of finite products of coalgebras for the comonad on $\mathbb{C}$ induced by the adjunction. We further require that $\mathbb{B}$ is cartesian, and that the fibration $\mathbb{C} \to \mathbb{B}$ has a generic object, over Ω in $\mathbb{B}$ say, and simple products with respect to projections $\Xi \times \Omega \to \Xi$ for $\Xi \in \mathbb{B}$. Finally, we require that there is a term modeling the fixed point combinator.

The language PILL$_Y$ is modeled in the fibration $\mathbb{C} \to \mathbb{B}$ using the fibred symmetric monoidal structure to model $\otimes, \multimap, I$. The type constructor ! is modeled by the fibred comonad FG on $\mathbb{C} \to \mathbb{B}$. Polymorphism is modeled using the simple product as was the case for λ_2. A term

$$\Xi \mid \vec{x} \colon \vec{\sigma}; \vec{y} \colon \vec{\sigma}' \vdash t \colon \tau$$

is modeled as a vertical morphism

$$[\![t]\!] \colon (\bigotimes_i FG[\![\Xi \vdash \sigma_i]\!]) \otimes (\bigotimes_j [\![\Xi \vdash \sigma'_j]\!]) \to [\![\Xi \vdash \tau]\!]$$

in $\mathbb{C}$.

The reader may be wondering why a PILL$_Y$-model is an adjunction and not just a fibred comonad satisfying certain conditions. Of course we might as well have given the definition this way, but we like to keep the category of finite products of algebras for the comonad in the picture for the following reason.

Suppose

$$\Xi \mid \vec{x} \colon \vec{\sigma}; - \vdash t \colon \tau$$

is a term. Then t is modeled as a map

$$[\![t]\!] \colon \bigotimes_i FG[\![\Xi \vdash \sigma_i]\!] \to [\![\Xi \vdash \tau]\!].$$

One can prove that for any symmetric monoidal adjunction the left adjoint is strong, i.e., $F(A) \otimes F(B) \cong F(A \times B)$, and so using the adjunction $F \dashv G$, $[\![t]\!]$ corresponds to a map

$$\widehat{[\![t]\!]} \colon \prod_i G[\![\Xi \vdash \sigma_i]\!] \to G[\![\Xi \vdash \tau]\!].$$

in $\mathbb{D}$. Thus, the fibration $\mathbb{D} \to \mathbb{B}$ models the part of the calculus consisting of terms with purely intuitionistic variable contexts.

3 Models of Parametric Polymorphism

Having seen what models of polymorphism are, a natural question to ask is “What does it mean for a λ_2-fibration or a PILL$_Y$-model to model *parametric* polymorphism?”. This dissertation proposes an answer to this question, but before presenting it we discuss what a good notion of parametric model should be.

General requirement. *A good notion of parametricity for models of polymorphism should be such that all parametric models satisfy the consequences of parametricity described in Sections 1.1,1.2. This means that we should be able to prove correctness of the encoding of inductive / coinductive types and data abstraction results.*

Recall the example of the λ_2-type

$$\mathrm{Nat} = \prod \alpha \colon \mathsf{Type}.\, (\alpha \to \alpha) \to (\alpha \to \alpha)$$

from Section 1.1. The interpretation of this type in a λ_2-fibration modeling parametric polymorphism should be a type of natural numbers, which in the language of category theory means that it should be a natural numbers object. Since terms are interpreted as maps in the fibre categories of the λ_2-fibration, the interpretation of Nat should be a natural numbers object in the fibres. For any λ_2 fibration one can prove that $[\![\vec{\alpha} \vdash \mathrm{Nat}]\!] = !_{\Omega^n}^* [\![- \vdash \mathrm{Nat}]\!]$ where $!_{\Omega^n} : \Omega^n \to 1$ is the unique map into the terminal object of the base category. We require that for each Ξ object in the base category, $!_{\Xi}^*[\![- \vdash \mathrm{Nat}]\!]$ is a natural numbers object in the fibre over Ξ. Notice that the family $(!_{\Xi}^*[\![- \vdash \mathrm{Nat}]\!])_{\Xi}$ is closed under reindexing.

In general — since the category theoretic correspondent to inductive types is initial algebras — the interpretations of the encodings of inductive types should induce families of initial algebras in any parametric λ_2-fibration. Likewise the interpretation of coinductive types should induce families of final coalgebras. In parametric models of PILL$_Y$ the interpretations of the encodings of recursive types should produce solutions to recursive domain equations in the model.

To my knowledge no definitive categorical formulation of data abstraction has emerged. One approach is to ask for the existence of a logic to reason about the internal language of the model, in which one can formulate data abstraction properties. Another approach is to require existential types to exist in the fibres of the model, in which case this requirement resembles that of inductive and coinductive data types. In this dissertation I have focused on the requirements for encoding of data types.

3.1 Models of Abadi & Plotkin’s logic

Our notion of parametricity for models of polymorphism will be based on relational parametricity. As mentioned, to formulate relational parametricity one must specify what is meant by relations. Some models may be parametric with respect to one notion of relations but not with respect to other (as is the case for the domain theoretic model of Paper 2).

Many models considered in the literature (such as the per-model) exist inside an ambient set theory (such as the internal language of a topos) and thus have a natural notion of relations available. In such cases a natural definition of parametric model is obtained by formulating the parametricity schema in the set theory available. Basically, having modeled the parametricity schema in the ambient logic, one should be able to do the proofs as presented in Abadi & Plotkin’s logic (or variants of it) in the ambient logic and use this to prove correctness of the encoding of data types of Section 1.1.

Often, however, only a subset of the relations available in the set theory is used in the formulation of parametricity. Examples include $\top\top$- closed relations as in [25, 5] and relations given by subdomains as in [35].

Generalizing the cases mentioned above, in this dissertation a parametric model of λ_2 will be a model of Abadi & Plotkin's logic for parametricity satisfying the parametricity schema.

The interest in working out the details of such a definition is two-fold. First, we will be able to unify the proofs of consequences of parametricity worked out in specific models (such as [35, 5]). These consequences should not be worked out in each specific model, but be consequences of the parametric structure on the model, proved once and for all. We should also be able to use these results on models obtained from parametric completion [33]. To my knowledge the proofs of correctness of encoding of data types for these in general do not exist in the literature.

Second, we should be able to identify what exactly is needed to model the logic for parametricity and reasoning with it. For example, models of Abadi & Plotkin's logic often come from some ambient logic of a model, but exactly how close to set theory does this logic have to be? It has also been unclear whether parametricity only implied correctness of encoding of data types for well-pointed models [7] (the answer is negative). Finally, as mentioned, some models use only a subset of the relations available in the logic when reasoning about parametricity. What exactly is required for such a subset to be usable for reasoning about parametricity?

4 Contributions of this dissertation

In this section we list the main contributions of this dissertation. The discussion here will be a bit more precise than the text above, but still the results will not always be described in full detail, and so we refer to the full papers.

4.1 Abadi & Plotkin's logic

As said, we define models of parametric polymorphism to be models of Abadi & Plotkin's logic for parametricity. Before discussing the models however, we sketch Abadi & Plotkin's logic. A full description of the logic can be found in Paper 1.

Abadi & Plotkin's logic is a logic for reasoning about parametricity for λ_2. We need to be able to formulate propositions quantifying over types and terms in λ_2 and relations on types in λ_2. Therefore propositions of the logic live in contexts of free type variables, free ordinary variables and free relational variables. We write

$$\vec{\alpha} \mid x_1 \colon \sigma_1, \ldots, x_n \colon \sigma_n \mid R_1 \colon \mathrm{Rel}(\tau_1, \tau_1'), \ldots, R_m \colon \mathrm{Rel}(\tau_m, \tau_m') \vdash \phi \colon \mathsf{Prop}.$$

The vector $\vec{\alpha}$ is a vector of type variables and each $\sigma_i, \tau_j, \tau_j'$ is a type of λ_2 with free variables in $\vec{\alpha}$. The x_i's are the free variables and the R_j's are the free relational variables. Atomic propositions can be formed using equality: if t, u are terms of λ_2 of type ω in the context

$$\vec{\alpha} \mid x_1 \colon \sigma_1, \ldots, x_n \colon \sigma_n$$

then $t =_\omega u$ is a proposition.

In the logic, we also have a notion of definable relations. Any relation $R_j \colon \mathrm{Rel}(\tau_j, \tau_j')$ in the context is a definable relation. If ϕ is a proposition in the logic with free variables $x \colon \sigma, y \colon \tau$ then we can form the relation $(x \colon \sigma, y \colon \tau).\, \phi \colon \mathrm{Rel}(\sigma, \tau)$. As an example, we mention the equality relation eq_σ on a type σ defined by

$$(x \colon \sigma, y \colon \sigma).\, x =_\sigma y.$$

If $\rho\colon \mathrm{Rel}(\sigma,\tau)$ is a definable relation and $t\colon \sigma, u\colon \tau$ are terms, then $\rho(t,u)$ is a proposition. In particular, if $R_j\colon \mathrm{Rel}(\tau_j,\tau_j')$ is a relation in the context, and t,u are terms of type τ_j,τ_j' respectively then $R_j(t,u)$ is a proposition.

Further constructions in the logic include the constructions of propositional logic and quantification over type variables, ordinary variables and relational variables.

Finally, there is a *relational interpretation* of types: If $\sigma(\vec{\alpha})$ is a type with n free type variables and $\rho_1\colon \mathrm{Rel}(\tau_1,\tau_1'), \ldots \rho_n\colon \mathrm{Rel}(\tau_n,\tau_n')$ are definable relations, then $\sigma[\rho_1,\ldots\rho_n]\colon \mathrm{Rel}(\sigma(\vec{\tau}),\sigma(\vec{\tau}'))$ is a definable relation.

The relational interpretation of types is used to formulate relational parametricity (as Reynolds did) as the *identity extension schema* stating that $\sigma[\mathrm{eq}_{\vec{\alpha}}]$ is the equality relation on $\sigma(\vec{\alpha})$. The intuition is that for any type of the form $\prod\alpha.\,\sigma$ (let us assume that this type is closed) and any element x of that type (x,x) is in the relational interpretation of $\prod\alpha.\,\sigma$, which by axioms of the logic should be equivalent to requiring that

$$\forall\alpha,\beta\colon \mathsf{Type}.\,\forall R\colon \mathrm{Rel}(\alpha,\beta).\,\sigma[R](x(\alpha),x(\beta)).$$

In words, for all pairs of types α,β and all relations between them $R\colon \mathrm{Rel}(\alpha,\beta)$ the α- and β-components of x are related in the relational interpretation of σ.

The definition of the relational interpretation of types differs from the original presentation of the logic [29], where $\sigma[\vec{\rho}]$ is defined by induction over the structure of σ. What we require is basically a relational interpretation of all type constants in the language as well. Suppose for instance that some type construction $\diamond$ between pairs of types is added to λ_2. To talk about parametricity for the new language, we should add a relational interpretation of $\diamond$, i.e., for each pair of relations $R\colon \mathrm{Rel}(\sigma,\sigma'), S\colon \mathrm{Rel}(\tau,\tau')$ we must define the relation $R\diamond S\colon \mathrm{Rel}(\sigma\diamond\tau,\sigma'\diamond\tau')$. This means that we may reason about parametricity at types formed using also these type constructors.

The inductive definition of the relational interpretation of types of [29] is captured in axioms of the logic.

The correctness of the encodings of data types can be expressed in Abadi & Plotkin's logic, and can be proved to follow from parametricity. This was stated in theorems in [29], but the proofs were not included in the paper. Some arguments of this sort appear in [39] and some proofs are written out for a specific model in [12]. However, even with these references at hand, the proofs are non-trivial to construct, and so we have included them in this dissertation.

4.2 APL-structures

An *APL-structure*, is a model of Abadi & Plotkin's logic. To define the notion of APL-structure we first define a notion of pre-APL-structure. A *pre-APL-structure* is a diagram

$$\begin{tikzcd}
 & \mathbf{Prop} \arrow[d] \\
\mathbf{Type} \arrow[r, "I"] \arrow[dr] & \mathbf{Ctx} \arrow[d] \\
 & \mathbf{Kind}
\end{tikzcd}$$

where $\mathbf{Type} \to \mathbf{Kind}$ is a λ_2-fibration (the model we reason about) and I is a fibred faithful product-preserving inclusion of $\mathbf{Type}$ into a larger category containing for each pair of objects σ,τ of the same fibre of $\mathbf{Type}$ an object $U(\sigma, \tau)$ of all relations between σ, τ. $\mathbf{Prop} \to \mathbf{Ctx}$ is a logic fibration in which we interpret the formulas of Abadi & Plotkin's logic. In $\mathbf{Ctx}$ we can model the full contexts of propositions as

$$\begin{gathered}[\![\alpha \mid x_1 \colon \sigma_1, \dots, x_n \colon \sigma_n \mid R_1 \colon \mathrm{Rel}(\tau_1, \tau_1'), \dots, R_m \colon \mathrm{Rel}(\tau_m, \tau_m')]\!] = \\ \prod_i I([\![\sigma_i]\!]) \times \prod_j U([\![\tau_j]\!], [\![\tau_j']\!])\end{gathered}$$

using the inclusion I and modeling $\mathrm{Rel}(\tau_i, \tau_i')$ as the object of all relations from τ_i to τ_i' in $\mathbf{Ctx}$. The products in this definition are the products of the fibre category.

From a pre-APL-structure we can define a λ_2-fibration of relations denoted

$$\mathbf{Relations} \to \mathbf{RelCtx}.$$

Basically the objects of each fibre are relations, and the λ_2 structure is defined using the same constructions that give the inductive definition of relational interpretation of types in [29]. For example, for relations $\rho \colon \mathrm{Rel}(\sigma, \tau)$, $\rho' \colon \mathrm{Rel}(\sigma', \tau')$ the relation $\rho \to \rho'$ is defined as the relation

$$(f \colon \sigma \to \sigma', g \colon \tau \to \tau'). \forall x \colon \sigma, y \colon \tau. \rho(x, y) \supset \rho'(f(x), g(y)).$$

There exists a pair of maps of λ_2-fibrations

$$\begin{tikzcd} \left(\begin{matrix} \mathbf{Relations} \\ \downarrow \\ \mathbf{RelCtx} \end{matrix}\right) \arrow[r, shift left, "\partial_0"] \arrow[r, shift right, "\partial_1"'] & \left(\begin{matrix} \mathbf{Type} \\ \downarrow \\ \mathbf{Kind} \end{matrix}\right) \end{tikzcd}$$

mapping a relation to its domain and codomain respectively. An APL-structure is a pre-APL-structure such that there is a map of λ_2-fibrations J going the other way satisfying $\partial_0 \circ J = \partial_1 \circ J = id$. The functor J models the relational interpretation of types.

We show that the interpretation of Abadi & Plotkin's logic in an APL-structure is sound. Moreover, the class of APL-structures is complete with respect to Abadi & Plotkin's logic, i.e., any sentence of Abadi & Plotkin's logic that holds in all APL-structures is provable in the logic.

We can reason about APL-structures using Abadi & Plotkin's logic. Thus, if the parametricity schema holds in the internal logic of the APL-structure, we can prove correctness of the encoding of inductive and coinductive types in the internal logic. However, to conclude from the statement in the internal logic to the structure of the fibres of $\mathbf{Type}$, we need to know that morphisms in $\mathbf{Type}$ that can be proved equal in the internal logic of the APL-structure in fact are equal in the category $\mathbf{Type}$. This property is a well-known property of logic fibrations called *very strong equality*.

A key ingredient in the proofs is *extensionality* for functions and polymorphic elements, i.e. the logical rules

$$\begin{gathered}\forall x \colon \sigma. f(x) =_\tau g(x) \supset f =_{\sigma \to \tau} g \\ \forall \alpha \colon \mathsf{Type}. t\, \alpha =_\sigma u\, \alpha \supset t =_{\prod \alpha. \sigma} u.\end{gathered}$$

We thus define a *parametric APL-structure* to be an APL-structure with very strong equality in which parametricity and extensionality holds in the internal language.

The main theorem of APL-structures states that they model inductive and coinductive types. Before we state it, we should be more precise about what we mean by inductive types. First we introduce the distinction between *pure* λ_2 and λ_2 calculi in general. Pure λ_2 has no extra type or term constants. We may also talk about λ_2-calculi in general. These have added type and term constants, and include for example the internal language of a λ_2-fibration.

A type $\alpha \vdash \sigma(\alpha)$ defined in pure λ_2 in which α occurs only positively (see for example Paper 1 or [29]) induces a functor in the sense that there exists a term

$$M\colon \prod \alpha, \beta\colon \mathsf{Type}.\, (\alpha \to \beta) \to (\sigma(\alpha) \to \sigma(\beta))$$

preserving identities and composition. The interpretations of σ and M induce a fibred functor

$$\begin{tikzcd}
\mathbf{Type} \arrow[rr] \arrow[dr] & & \mathbf{Type} \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

and we shall be interested in initial algebras and final coalgebras for the restrictions of this functor to the fibres of $\mathbf{Type} \to \mathbf{Kind}$.

In general we define a *polymorphically strong* fibred functor to be a functor with a corresponding type σ and term M existing *in the model* but not necessarily in pure λ_2. This is clearly a generalization of the above construction.

The main theorem is the following.

Theorem 4.1. *Every polymorphically strong fibred functor has families of initial algebras and final coalgebras, i.e., there exists a family of initial algebras / final coalgebras for each restriction of the functor to a fibre over* $\mathbf{Kind}$ *and these families are closed under reindexing along maps in* $\mathbf{Kind}$.

For example, we can show that each fibre has coproducts and the initial algebra corresponding to the type $\alpha \vdash \alpha + 1$ is a natural numbers object. This natural numbers object is the interpretation of $\prod \alpha.\, (\alpha \to \alpha) \to \alpha \to \alpha$.

Thus the notion of parametric APL-structure gives a categorical notion of models of parametric polymorphism satisfying our requirements.

As an example of a model we consider a well-known parametric variant of the per-model [2]. This model has as types pairs (f^p, f^r) of maps such that $f^p\colon \mathbf{Per}^n \to \mathbf{Per}$ and for each vector

$$R_1\colon \mathrm{Rel}(A_1, B_1), \ldots, R_n\colon \mathrm{Rel}(A_n, B_n),$$

of relations on pers

$$f^r(\vec{R})\colon \mathrm{Rel}(f^p(\vec{A}), f^p(\vec{B})),$$

where by relations $R\colon \mathrm{Rel}(A, B)$ for pers A, B we mean subsets of $\mathbb{N}/A \times \mathbb{N}/B$. We require that f^r applied to a vector of equality relations gives an equality relation. We show that this model can be embedded into a parametric APL-structure, such that Theorem 4.1 applies. A variant of this construction in relative realizability [6] gives us *non-wellpointed* parametric APL-structures (the fibres of $\mathbf{Type}$ are not well-pointed). This shows that well-pointedness is not necessary for correctness of the encodings of data types to hold.

It is also worth noticing that the construction of models of Abadi & Plotkin's logic has proved consistency of the logic.

I have not studied morphisms between APL-structures, since it is not clear to me why these could be interesting. One weakness of APL-structures as models of Abadi & Plotkins logic, which would probably show up when giving such a definition of morphisms, is that Abadi & Plotkin's logic only gives notation for the objects in $\mathbf{Ctx}$ of the form $I(\sigma)$ or $U(\sigma, \tau)$ for σ, τ objects of $\mathbf{Type}$. Thus, there would not be a bijective correspondence between maps between APL-structures and translations between the internal languages of the APL-structures. However, this is of no concern to us as long as we are only interested in using the APL-structure for reasoning about the included λ_2 fibration.

4.3 LAPL-structures

The language PILL$_Y$ was first sketched by Plotkin in [28] in which he also sketched a version of the logic for parametricity for PILL$_Y$, and gave a rough sketch of a concrete parametric model of PILL$_Y$. In this dissertation we give a full presentation of the logic and a notion of LAPL-structures (Linear Abadi-Plotkin Logic) which model the logic. We have also worked out the details of the concrete model.

As mentioned, many of the concrete parametric domain theoretic models we consider have a canonical logic, but are only parametric with respect to a subset of the relations in the logic. To handle these cases, our logic for parametricity will have to include a notion of admissible relations. For reasoning about parametricity one needs a good supply of these relations, in particular graphs of linear functions should be admissible relations. We state a number of rules that the set of admissible relations should be closed under.

Even though the language PILL$_Y$ is combined linear and intuitionistic, the logic we present is purely intuitionistic, i.e., it only has intuitionistic variables. Expressions in the logic are written as

$$\vec{\alpha} \mid \vec{x} \colon \vec{\sigma} \mid \vec{R} \colon \mathrm{Rel}(\vec{\tau}, \vec{\tau}'), \vec{S} \colon \mathrm{AdmRel}(\vec{\omega}, \vec{\omega}') \vdash \phi \colon \mathsf{Prop}.$$

The proposition ϕ can contain terms t such that

$$\vec{\alpha} \mid \vec{x} \colon \vec{\sigma}; - \vdash t \colon \tau$$

is a term of PILL$_Y$. The constructions in the logic are much as in the logic for λ_2 except that we also have admissible relations. We omit the details here, but mention that for types σ with n free variables, the relational interpretation $\sigma[\vec{\rho}]$ is only defined for $\vec{\rho}$ a vector of admissible relations.

As with the APL-structures, to define the notion of LAPL-structure, we must first define the notion of pre-LAPL-structure. Roughly, a pre-LAPL-structure is a diagram

$$\begin{tikzcd}
 & & \mathbf{Prop} \arrow[d] \\
\mathbf{LinType} \arrow[r, bend right] \arrow[rrd] & \mathbf{Type} \arrow[l, bend right] \arrow[r, "I"] \arrow[rd] & \mathbf{Ctx} \arrow[d] \\
 & & \mathbf{Kind}
\end{tikzcd}$$

The left hand side of the diagram is the model of PILL$_Y$ that we reason about. The functor I is a fibred product preserving faithful functor, and as usual $\mathbf{Prop} \to \mathbf{Ctx}$ is a logic fibration and $\mathbf{Ctx}$ contains objects of relations for all pairs of types σ, τ in the same fibre of $\mathbf{LinType}$. A notion of admissible relations for a pre-LAPL-structure is a family of subobjects of the objects of relations in $\mathbf{Ctx}$ closed under the rules for admissible relations in the logic.

From a pre-LAPL-structure with a notion of admissible relations, one can construct a model of PILL (it does not necessarily model the fixed point combinator Y). The model is denoted

$$\begin{tikzcd}
\mathbf{LinAdmRelations} \arrow[rr, bend right] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{AdmRelations} \arrow[ll, bend right] \arrow[dl] \\
 & \mathbf{AdmRelCtx}. &
\end{tikzcd}$$

The objects of $\mathbf{LinAdmRelations}$ are admissible relations, and the morphisms are pairs of strict morphisms preserving relations.

As for the APL-structures there exists two maps ∂_0, ∂_1 of PILL-models out of the constructed PILL-model mapping an admissible relation to its domain and codomain respectively. An LAPL-structure is a pre-LAPL-structure such that there exists a map of PILL-models J going the other way satisfying $\partial_0 \circ J = \partial_1 \circ J = id$. Again J gives a relational interpretation of types.

We show soundness of the interpretation of Abadi & Plotkin's logic in LAPL-structures and we show a completeness result as for APL-structures.

As in the case of APL-structures a parametric LAPL-structure should be an LAPL-structure with very strong equality such that parametricity and extensionality holds in the internal logic.

We can define a notion of polymorphically strong fibred functor and show that these have initial algebras and final coalgebras as we did with APL-structures, but as mentioned the new setting here should also enable us to solve recursive domain equations.

Suppose $\alpha \vdash \sigma$ is a type in *pure* PILL_Y. A solution to the recursive domain equation induced by σ is a closed type τ such that $\sigma(\tau)$ is isomorphic to τ. If σ had all its occurrences of α as positive, it would define a functor, and the initial algebra as well as the final coalgebra would be solutions to the domain equation $\sigma(\tau) \cong \tau$.

We may split the occurrences of α in σ into positive and negative obtaining a type $\alpha, \beta \vdash \sigma(\alpha, \beta)$ such that α occurs only negatively and β only positively. Such a type induces a functor which is contravariant in the first variable and covariant in the second, in the sense that there exists a term

$$M\colon \prod \alpha, \alpha', \beta, \beta'. \, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to (\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta'))$$

preserving composition and identities. Such a term induces a fibred functor

$$\begin{tikzcd}
\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType} \arrow[rr] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind}. &
\end{tikzcd}$$

The category $\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType}$ is the fibrewise product of the category obtained by taking fibrewise opposite category of $\mathbf{LinType}$ and $\mathbf{LinType}$. In general, such fibred functors are *polymorphically strong* if there exists a corresponding type σ and term as above in the internal language of the model (i.e. not necessarily in pure PILL_Y).

A solution to a domain equation induced by such a functor F is a family $(\tau_\Xi)_\Xi$ indexed over Ξ in $\mathbf{Kind}$ closed under reindexing such that $F(\tau_\Xi, \tau_\Xi) \cong \tau_\Xi$, i.e., a family of fixed points for the functor.

Theorem 4.2. *For parametric LAPL-structures*

- *every polymorphically strong fibred endofunctor on* $\mathbf{LinType} \to \mathbf{Kind}$ *has a family of initial algebras and a family of final coalgebras.*
- *every polymorphically strong fibred functor*

$$\begin{tikzcd}
\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType} \arrow[rr] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

has a family of fixed points closed under reindexing.

The logical part of the proof of Theorem 4.2 was sketched by Plotkin in [28]. Our contribution has been to write out the details and to show how this could be applied to LAPL-structures.

As mentioned, we also construct a concrete LAPL-structure based on the one sketched by Plotkin. This model of PILL$_Y$ involves admissible pers over a reflexive domain, i.e., a domain (a cpo with a least element $\bot$) such that $[D \to D]$ is a retract of D. An admissible per is a partial equivalence relation which is closed under lub's of chains and which relates $\bot$ to itself. The concrete model is then constructed as the parametric variant of the per-model, where we only consider admissible pers.

4.4 Completion Processes

Recall from Section 2 that even though no classical set theoretic models of polymorphism exist, set theoretic models of polymorphism might still exist in intuitionistic set theories. The examples we have in mind are internal cartesian closed subcategories $\mathbf{C}$ in quasi-toposes. If $\mathbf{C}$ is sufficiently complete, we can construct a model of λ_2 in which types with n free variables are modeled as morphisms

$$\mathbf{C}_0^n \to \mathbf{C}_0$$

in the topos, where $\mathbf{C}_0$ is the object of objects for $\mathbf{C}$ (i.e. the model is the externalization of $\mathbf{C}$). We call such internal categories internal λ_2-models.

In this dissertation we show how the ambient set theory of the model gives rise to a canonical pre-APL-structure corresponding to the interpretation of Abadi & Plotkin's logic in the internal logic of the quasi-topos.

For this restricted class of models of λ_2 there exists a parametric completion process constructing parametric models based on the original model. This process was originally described in [33]. Our contribution has been to show that this process can be extended to construct parametric APL-structures.

The completion process described in [33] goes as follows: Since the quasi-topos $\mathbb{E}$ models an intuitionistic set theory, we may construct an internal category $\mathbf{LR}(\mathbf{C})$ whose objects are logical relations on objects of $\mathbf{C}$ from the quasi-topos, and whose morphisms are pairs of morphisms in $\mathbf{C}$ preserving relations (i.e. mapping related elements to related elements). There exists a diagram of internal functors in the quasi-topos

$$\begin{tikzcd}\mathbf{LR}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l]\end{tikzcd}$$

mapping a relation to its domain and codomain respectively, and mapping an object of $\mathbf{C}$ to the identity relation on the same object. This graph is reflexive, meaning that the two compositions starting and ending in $\mathbf{C}$ are the identity.

The diagram $\begin{tikzcd}\mathbf{LR}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l]\end{tikzcd}$ makes up an internal category in the quasi-topos of reflexive graphs in $\mathbb{E}$. We denote this quasi-topos by $\mathbb{E}^G$. We can now apply the construction above to this internal category and obtain a λ_2-fibration.

We can describe this model more explicitly. A type in the parametrically completed model with n free variables is a type in the original model $\sigma\colon \mathbf{C}_0^n \to \mathbf{C}_0$ plus a map ρ that takes n-vectors of relations $(R_1\colon \mathrm{Rel}(A_1, B_1), \dots, R_n\colon \mathrm{Rel}(A_n, B_n))$ and produces a new relation

$$\rho(\vec{R})\colon \mathrm{Rel}(\sigma(\vec{A}), \sigma(\vec{B}))$$

such that $\rho(\mathrm{eq}_{A_1}, \dots, \mathrm{eq}_{A_n}) = \mathrm{eq}_{\sigma(\vec{A})}$. Terms are terms in the old model preserving relations.

A type in the parametrically completed model has a built-in relational interpretation (ρ). Since this relational interpretation satisfies identity extension, the model should be parametric.

In this dissertation, we show that the parametric completion process produces models that fit into a parametric APL-structure. This provides formal proofs of the correctness of the encodings of inductive and coinductive types in these models. This result is of course expected, but to our knowledge it has not been formally proved before in this generality. The APL-structure is also interesting, because we clarify with respect to which logic the completed category is parametric. The parametrically completed model is not parametric with respect to the internal logic of the quasi-topos $\mathbb{E}^G$, but with respect to a related logic corresponding to the internal logic of $\mathbb{E}$.

The concrete APL-structure mentioned in Section 4.2 arises as the result of a completion process, when considering the category of pers as an internal category in the category of assemblies. Since the category of assemblies is a quasi-topos, this provides the motivation for using quasi-toposes instead of toposes. Of course, the category of pers is also an internal category in the effective topos, but this viewpoint gives a different logic.

We also construct a parametric completion process for LAPL-structures. First we describe which kind of data is needed for an internal model of PILL$_Y$ to give rise to an LAPL-structure as above. Next we describe the parametric completion process. This is basically the same as for APL-structures, but still some constructions in this process are new and so the construction is non-trivial.

The parametric LAPL-structure mentioned in Section 4.3 can be seen as a result of the parametric completion process for LAPL-structures.

4.5 An LAPL-structure from Synthetic Domain Theory

In recent work [35] Alex Simpson and Pino Rosolini have studied a language which we shall call Lily$_{\text{strict}}$. This language is basically PILL$_Y$ with linear functions replaced by strict functions. Lily$_{\text{strict}}$ is equipped with two operational semantics: a call-by-name semantics and a call-by-value semantics (with these operational semantics, Lily$_{\text{strict}}$ is simply Lily [5] with linearity replaced by strictness).

Simpson and Rosolini give an interpretation of this language using Synthetic Domain Theory (SDT), and prove this interpretation to be adequate with respect to the two notions of contextual equivalence obtained from each of the operational semantics. Using this they show that the two contextual equivalence relations coincide. Since Lily$_{\text{strict}}$ and Lily are almost the same language, this result was basically proved in [5] using operational tools.

The interpretation lives inside an intuitionistic set theory. The construction resembles that of the parametric completion process, and so all types in the interpretation are equipped with a relational interpretation satisfying an identity extension condition. Thus the interpretation is parametric with respect to the interpretation of parametricity in the ambient set theory and we would expect that the encoding of the inductive and coinductive data types is correct, but [35] does not formally prove this.

We construct a parametric LAPL-structure based on the interpretation of Lily$_{\text{strict}}$ using SDT. Since linear functions are strict we may translate PILL$_Y$ into Lily$_{\text{strict}}$, and up to this translation, the interpretation of PILL$_Y$ in the parametric LAPL-structure we construct agrees with the interpretation of Lily$_{\text{strict}}$ given by Simpson and Rosolini.

The construction of this LAPL-structure serves two purposes: first it helps to show that the notion of LAPL-structures is general enough to handle different types of models. In this case, it strengthens the idea that parametric PILL$_Y$ is a good language for domain theoretic models of parametric polymorphism. Second, using adequacy of the interpretation of Lily$_{\text{strict}}$ we can use the parametric model to show consequences of parametricity (i.e. correctness of the encodings of data types) in Lily$_{\text{strict}}$ up to operational equivalence.

This is very much in the spirit of Simpson and Rosolini's proof of coincidence of the contextual equivalence relations using the adequate interpretation [35].

5 Related Work

In this section we focus on three related directions of research, Ma & Reynold's categorical definition of parametricity, Dunphy's parametricity graphs and the work on consequences of parametricity for the programming language Lily by Bierman Pitts and Russo. Finally, we sketch some of the other directions of research related to parametricity.

5.1 Ma & Reynolds notion of parametricity

QingMing Ma and John Reynolds [30] have proposed a category-theoretic definition of parametricity for models of λ_2 [16]. The definition can basically be restated as follows: Suppose $\mathbb{E} \to \mathbb{B}$ is a λ_2 fibration, and suppose we are given a logic fibration $\mathbb{D} \to \mathbb{E}_1$ on the fibre of $\mathbb{E}$ over the terminal object (this is the category of *closed* types).

Ma & Reynolds define $\mathbb{E} \to \mathbb{B}$ to be parametric if there exists a reflexive graph of λ_2-fibrations

$$\begin{tikzcd} \begin{pmatrix}\mathbb{E}\\ \big\downarrow\\ \mathbb{B}\end{pmatrix} \arrow[r, shift left=2, leftarrow] \arrow[r] \arrow[r, shift right=2, leftarrow] & \begin{pmatrix}\mathbb{F}\\ \big\downarrow\\ \mathbb{C}\end{pmatrix} \end{tikzcd}$$

(i.e. a graph, where the two compositions starting at $\mathbb{E} \to \mathbb{B}$ are the identity) whose restriction to the fibres over the terminal objects is isomorphic to

$$\begin{tikzcd} \mathbf{LR}(\mathbb{E}_1) \arrow[r, shift left=2] \arrow[r, leftarrow] \arrow[r, shift right=2] & \mathbb{E}_1 \end{tikzcd}$$

where $\mathbf{LR}(\mathbb{E}_1)$ is a category of relations on $\mathbb{E}_1$ formed using the logic $\mathbb{D} \to \mathbb{E}_1$ and the morphisms map a relation to its domain and codomain respectively and a closed type to the equality relation on that type.

An APL-structure is parametric in the sense of Ma & Reynolds, since the fibration $\mathbf{Relations} \to \mathbf{RelCtx}$ can play the role of $\mathbb{F} \to \mathbb{C}$, and in general the intuition of the reflexive graph of λ_2-fibrations is that the fibration $\mathbb{F} \to \mathbb{C}$ is a fibration of relations. But since this is only formulated for the closed types, we cannot use it to prove consequences of parametricity for open types. See Paper 1 for a further discussion of the relation to Ma & Reynolds definition.

5.2 Parametricity graphs

In a recent PhD dissertation Brian Dunphy [7, 8] together with his adviser Uday Reddy, has studied a class of models of polymorphism based on reflexive graphs of categories $\begin{tikzcd} \mathbf{G_e} \arrow[r, shift left=2] \arrow[r, leftarrow] \arrow[r, shift right=2] & \mathbf{G_v} \end{tikzcd}$. Under certain conditions on such a reflexive graph one can build a model of polymorphism where types with n free variables are modeled as pairs of functors making the diagram

$$\begin{tikzcd} {|\mathbf{G_e}|^n} \arrow[r] \arrow[d, shift right=2] \arrow[d, leftarrow] \arrow[d, shift left=2] & \mathbf{G_e} \arrow[d, shift right=2] \arrow[d, leftarrow] \arrow[d, shift left=2] \\ {|\mathbf{G_v}|^n} \arrow[r] & \mathbf{G_v} \end{tikzcd}$$

commute, where $|\mathbf{G_v}|$ denotes the discrete category on the objects of $\mathbf{G_v}$. Dunphy states conditions under which the category $\mathbf{G_e}$ can be considered a category of relations on $|\mathbf{G_v}|$. Reflexive graphs satisfying these conditions are called *parametricity graphs*, and correctness of the encoding of data types can be shown for these using a logic resembling a logic called System R [1] for reasoning about parametricity.

One technical issue worth mentioning is that Dunphy can only in general prove correctness of the encoding of data types for *well-pointed* parametricity graphs. Dunphy even gives an example of a non-wellpointed parametricity graphs in which the encodings are not correct. Since we give an example of a non-wellpointed parametric APL-structure, we show that parametricity is in fact useful in a setting without well-pointedness.

The main difference between Dunphy's work and this dissertation is that Dunphy does not give a general notion of parametricity for λ_2-fibrations. He only considers models given by reflexive graphs. So, for example the question of whether the standard per-model (as described in Section 2.1) is parametric does not make sense in Dunphy's setting. In this sense APL-structures may be more general than parametricity graphs. It should be mentioned that the *parametric* models considered in this dissertation all come from reflexive graphs and so are probably all parametricity graphs. But, as mentioned, some of these models are not well-pointed and so cannot be shown to satisfy consequences of parametricity using the tools of parametricity graphs, but only using the tools of APL-structures.

On the other hand, parametricity graphs model a logic that is different from Abadi & Plotkin's logic and so may incorporate some models that cannot fit into an APL-structure.

Finally, we mention that Dunphy also considers models of predicative polymorphism, which is not covered in this dissertation. It should however be easy to find a variant of the definition of APL-structures that would handle predicative polymorphism. However, most of our arguments for correctness of encoding of inductive and coinductive types use impredicativity, and so Dunphy's proofs would have to be adopted for this to work out.

In his dissertation Dunphy also considers parametricity graphs modeling PILL$_Y$-like languages.

Claudio Hermida and Robert Tennent study a related framework of parametric models in [13].

5.3 Parametricity in operational semantics

Parametric polymorphism has also been used in a more syntactic setting by Andrew Pitts in [25] and by Gavin Bierman, Andrew Pitts and Claudio Russo in [5] to prove properties of programming languages with operational semantics up to contextual equivalence. In [5] for example, the language Lily which is basically PILL$_Y$ equipped with two operational semantics: a call-by-name and a call-by-value operational semantics is considered. For each of these operational semantics a notion of contextual equivalence is defined by observing termination at types of the form $!\sigma$. Using operational methods the two notions of equivalence are shown to coincide.

Because there is a set of closed terms of Lily, one can use set theoretic relations to reason about them. In [5] a particular subset of these relations called $\top\top$-closed relations are used to reason about these terms, and it is shown that up to contextual equivalence Lily is parametric with respect to $\top\top$-closed relations. This parametricity result is then used to show correctness of an encoding of coproducts for closed types of Lily up to contextual equivalence.

It would be interesting to see if the language Lily with terms considered up to contextual equivalence gives rise to a parametric LAPL-structure. To show this, we need to check that $\top\top$-relations give a notion of admissible relations as defined in this dissertation. We do believe this is the case, and it is on the schedule for future work.

Showing that Lily gives rise to an LAPL-structure would formally prove that the encodings of inductive, coinductive and recursive types are correct. In fact we have almost done this already, as we have shown a similar result for Lily$_{\text{strict}}$ using the LAPL-structure obtained from SDT (see Section 4.5).

5.4 More related research

Ryu Hasegawa has studied a specific family of models for polymorphism and shown that for these parametricity of encoding of inductive and coinductive types is equivalent to correctness of these encodings [12]. The proofs in [12] inspired some of the proofs of the consequences of parametricity used in this dissertation. Ryu Hasegawa is also working on a model of a polymorphic linear type theory [11].

Parametric polymorphism has also been used to model local variables [22, 21]. The idea is to use parametricity to hide local variables from called procedures, the same way parametricity can be used to hide information about specific implementations of data types. In [22] models of an Algol-like language are given using reflexive graphs and it is shown how these models model hiding of local variables using parametricity. In [21] two versions of Algol are translated into a predicative version of polymorphic linear lambda-calculus (basically a predicative version of PILL$_Y$). Models of polymorphic linear lambda calculus can then give models of the Algol-like languages. The idea behind using linearity is that it can be used to rule out nonimperative behavior in the model such as functions restoring the old state after running an expression with side effects, since this requires copying the old state before running the expression. Many of the same ideas are used in [23] to construct fully abstract translations of PCF and an idealized version of Algol into a language with parametric polymorphism.

Other logics for reasoning about parametricity exist. Before Abadi & Plotkin's logic appeared a different logic had been proposed [1]. As mentioned Dunphy and Reddy [7, 8] use a variant of this logic. Izumi Takeuti has constructed a variant of Abadi & Plotkin's logic, in which one can also discuss other arities of parametricity (such as unary parametricity involving predicates instead of relations).

Ivar Rummelhoff [36] has studied the encoding of natural numbers in per-models over different PCA's, and showed that in some of these models, the encoding contains more than natural numbers. So these models cannot be parametric. Even though he does not mention it, this shows that unary parametricity is different from binary (relational) parametricity, since one can easily show that the encoding of the natural numbers in any per-model is unary parametric. Other studies of parametric polymorphism for per-models include [34, 9].

Philip Wadler [40] presents a viewpoint, where the abstraction property of [30] corresponds to the existence of a map mapping terms of second-order lambda calculus to expressions in a logic. On the other hand, a representation result of Girard's corresponds to a map going the other way.

6 Structure of the dissertation

This dissertation consists of five papers. Here follows a description of each paper.

Paper 1: L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*, 2005. To Appear (Accepted for publication).

We give a detailed description of Abadi & Plotkin's logic for parametricity, the definition of APL-structures and the interpretation of the logic in these. This is followed by proofs of soundness and completeness for the interpretation. We define parametric APL-structures and proceed to show Theorem 4.1 above. This involves proving the logical versions of these results as stated in [29]. We compare our notion of parametricity to that of Ma & Reynolds [16]. The parametric completion process is described for APL-structures and in connection with this we discuss parametricity for internal models of λ_2 in quasi-toposes.

Paper 2: L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of linear Abadi & Plotkin logic. Technical Report TR-2005-57, IT University of Copenhagen, February 2005.

In this article we describe the language PILL$_Y$ and the variant of Abadi & Plotkin's logic used for it. We show how to reason in this logic and in particular we prove correctness of encoding of inductive, coinductive and recursive data types in the logic. As in the first article, we define LAPL-structures, show how to interpret the logic in these and show that the interpretation is sound and complete. Parametric LAPL-structures are introduced, and we show how to use the logical proofs of the correctness of the encoding of data types to solve recursive domain equations in parametric LAPL-structures (Theorem 4.2). Finally we construct the parametric domain theoretic per-model, show that it fits into a natural parametric LAPL-structure and describe the interpretation of the encoding of the natural numbers in this.

Paper 3: R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005.

This paper contains mostly well-known material on models of PILL, based on in particular [3, 4, 10, 17, 18, 19]. Since none of the above mentioned present all the material needed for this dissertation, we have included an exposition of the theory. The material covered includes the 2-category of symmetric monoidal categories, linear categories, models of LNL and DILL, and a fibrational account of these concepts ending with models of PILL and PILL$_Y$.

Paper 4: R. E. Møgelberg, L. Birkedal, and G. Rosolini. Synthetic domain theory and models of linear Abadi & Plotkin logic. Technical Report TR-2005-59, IT University of Copenhagen, February 2005.

Here we present the LAPL-structure constructed from synthetic domain theory and use it to show consequences of parametricity for the operational semantics on Lily$_{\text{strict}}$. For readability we have included a full description of the setup of synthetic domain theory as presented in [35], the language Lily$_{\text{strict}}$ and a formulation of the adequacy result for the interpretation of Lily$_{\text{strict}}$ as shown by Simpson and Rosolini. The presentation of the setup of synthetic domain theory follows the presentation in [35] closely.

Paper 5: R. E. Møgelberg. Parametric completion for models of polymorphic intuitionistic / linear lambda calculus. Technical Report TR-2005-60, IT University of Copenhagen, February 2005.

The main result of this article is the description of the parametric completion process for LAPL-structures. Before this however, we review some theory of internal categories including internal fibrations and internal linear categories. We define a notion of internal PILL$_Y$-model in a quasi-topos, and show that the externalization of an internal PILL$_Y$-model gives rise to an LAPL-structure.

Dependencies are as follows. It is not necessary to read Paper 1 before Paper 2, except that Paper 2 uses a few definitions of Appendix A in Paper 1, but, for readers unfamiliar with parametricity, it may be helpful to start with Paper 1, since the proofs of consequences of parametricity given in Paper 2 are slightly more sophisticated than the ones in Paper 1 due to the use of linearity.

The material in Paper 2 depends on Paper 3, but since we think of the latter as a (long) appendix to Paper 2, we have placed it after Paper 2. Paper 3 can be read independently of all other papers in this dissertation. Paper 4 and Paper 5 can be read independently of each other, but they both depend on Paper 2.

7 Conclusion

We have introduced a notion of parametric APL structures which can be taken as definition of parametric models of second-order λ-calculus. These structures can be shown to have initial algebras and final coalgebras for a large class of fibred endofunctors, which means that parametric APL-structures give a good notion of parametric models as discussed in Section 3.

Likewise we have defined a notion of parametric LAPL-structures. These give a good notion of domain theoretic models of parametric polymorphism, since we can solve recursive domain equations in LAPL-structures, as we would expect to be able to in parametric domain theoretic models.

The definition of APL-structure ask for quite a lot of structure — besides the λ_2- fibration in question we ask for another fibration with a fibration on top, etc. But in the concrete case providing such extra structure to show that a λ_2-fibration is parametric just corresponds to answering the question "with respect to which logic is the model parametric".

This becomes even more apparent in the case of LAPL-structures. Concrete models considered in the literature, have often been parametric with respect to some logic, and a relational interpretation of types defined only on a subset of the relations of the logic: the ones we call admissible. Providing a full parametric LAPL-structure to a model corresponds to answering the question "with respect to which logic and which set of admissible relations is the model parametric?".

In both cases the APL- and LAPL-structures provide a check-list for what kind of structure is needed to reason about parametricity. In particular, for the LAPL-structures, we have a set of axioms that a notion of admissible relations should satisfy for it to be strong enough for reasoning about parametricity.

We have shown that parametric APL- and LAPL-structures provide a general and usable framework by showing that very different parametric models known from the literature are of this form. These involve parametric versions of per-models, and a family of models constructed using synthetic domain theory. We even have a very general way of constructing these models, namely using parametric completion processes.

Of the models presented in this dissertation, most were known as models of polymorphism, but for most of them, the correctness of the encodings of data types had not been shown formally. These proofs are presented in all details in this dissertation.

Another contribution of this dissertation is to sort out the details of the PILL$_Y$- version of Abadi & Plotkin's logic. In fact, for both versions of the logic considered here, we have worked out the details of models for them, thereby showing them to be consistent.

This dissertation has also provided detailed proofs of theorems that have been known to the community for long, but whose proofs have never appeared in print. These proofs are the proofs of correctness of encoding of initial algebras, final coalgebras and recursive types. These proofs are non-trivial, and it is my hope that making the details available will contribute to the accessibility of parametricity as a research area.

7.1 Future work

As said, we have provided a couple of very different parametric LAPL-structures showing that the notion is quite general. It would be interesting to see if Lily with terms identified up to contextual equivalence and $\top\top$-closed relations as admissible relations gives rise to a parametric LAPL-structure. This would imply that the correctness of the encodings of inductive and coinductive data types as sketched in [5] would be consequences of the same results for parametric LAPL-structures in a more direct way than the results proved in this dissertation using the SDT-model. This work is already under way.

We have shown how parametric polymorphism allows us to encode certain types with the right category theoretic properties. Parametricity also gives us reasoning principles for these types, but it is unclear whether these are the principles one will want to use in practice for reasoning about the language. In particular, for the LAPL-structures the reasoning principles only apply to admissible relations, which may not be a sufficiently large class of relations.

This dissertation is an abstract study of parametricity, and it would be interesting to show that these results can be used in the theory of programming languages in general. In this dissertation we have only once applied the abstract theory to show results about a programming language with an operational semantics, namely for the parametricity results for $\text{Lily}_{\text{strict}}$ up to operational equivalence. Can we use these models to show for example data abstraction results for real programming languages? How does our work relate to that of O'Hearn, Reynolds and Tennent [22, 21, 23] as briefly mentioned in Section 5.4.

The second-order lambda-calculus is a programming language (or an equational theory) suitable for studying parametricity, since it has few constructions. The language PILL_Y having fixed points is closer to a "real life" programming language. To be able to apply the theory of parametric polymorphism to programming languages used in practice, it needs to be studied in connection with effects.

Finally I do not think that the concept of parametricity is fully understood at this point. Parametric models contain less "junk" than other models at polymorphic types, so parametricity seems to provide a way of constructing better models. But how good are these models, and what are the connections to other good properties of models such as adequacy, universality and full abstraction? Not much work has been done in that area, [38] is an exception.

References

[1] Martín Abadi, Luca Cardelli, and Pierre-Louis Curien. Formal parametric polymorphism. *Theoretical Computer Science*, 121(1–2):9–58, December 1993. 5.2, 5.4

[2] E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70:35–64, 1990. 4.2

[3] A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, Edinburgh University, 1997. 2.2, 6

[4] P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical report, University of Cambridge, 1995. 6

[5] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal*, volume 41 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 2000. 3.1, 4.5, 5.3, 7.1

[6] Lars Birkedal and Jaap van Oosten. Relative and modified relative realizability. *Ann. Pure Appl. Logic*, 118(1-2):115–132, 2002. 4.2

[7] B.P. Dunphy. *Parametricity as a notion of uniformity in reflexive graphs*. PhD thesis, 2004. 3.1, 5.2, 5.4

[8] Brian Dunphy and Uday S. Reddy. Parametric limits. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS-04))*, pages 242–251, 2004. 5.2, 5.4

[9] P.J. Freyd, E.P. Robinson, and G. Rosolini. Dinaturality for free. In M. P. Fourman, P.T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 107–118. Cambridge University Press, 1991. 5.4

[10] Masahito Hasegawa. Categorical glueing and logical predicates for models of linear logic. 1999. 6

[11] R. Hasegawa. The theory of twiners and linear parametricity. 5.4

[12] R. Hasegawa. Categorical data types in parametric polymorphism. *Mathematical Structures in Computer Science*, 4:71–109, 1994. 4.1, 5.4

[13] C. Hermida and R.D. Tennent. A fibrational framework for possible-world semantics of algol-like languages. 2004. 5.2

[14] H. Huwig and A. Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73:101–112, 1990. 2.2

[15] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 2.1

[16] Q. Ma and J.C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40. Springer-Verlag, 1992. 5.1, 6

[17] Maria E Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. Technical Report CSR-01-7, University of Birmingham, School of Computer Science, August 2001. 2.2, 6

[18] Paola Maneggia. *Models of Linear Polymorphism*. PhD thesis, University of Birmingham, Feb. 2004. 6

[19] Paul-André Melliès. Categorical models of linear logic revisited. *Theoretical Computer Science*. To appear. 6

[20] J.C. Mitchell and G.D. Plotkin. Abstract types have existential types. *ACM Transactions on Programming Languages and Systems*, 10(3):470–502, July 1988. 1.2

[21] P. W. O'Hearn and J. C. Reynolds. From algol to polymorphic linear lambda-calculus. *Jrnl. A.C.M.*, 47(1):167–223, January 2000. 5.4, 7.1

[22] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *Journal of the ACM*, 42(3):658–709, May 1995. 5.4, 7.1

[23] P.W. O'Hearn and J.G. Riecke. Fully abstract translations and parametric polymorphism. In D. Sanella, editor, *Programming Languages and Systems, ESOP'94*, volume 788 of *Lecture Notes in Computer Science*, pages 454–468. Springer, 1994. 5.4, 7.1

[24] A. M. Pitts. Non-trivial power types can't be subtypes of polymorphic types. In *4th Annual Symposium on Logic in Computer Science*, pages 6–13. IEEE Computer Society Press, Washington, 1989. 2.1

[25] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000. 3.1, 5.3

[26] A.M. Pitts. Polymorphism is set theoretic, constructively. In D. H. Pitt, A. Poigné, and D. E. Rydeheard, editors, *Category Theory and Computer Science, Proc. Edinburgh 1987*, volume 283 of *Lecture Notes in Computer Science*, pages 12–39. Springer-Verlag, 1987. 2.1

[27] G. D. Plotkin. Type theory and recursion (extended abstract). In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, page 374, Montreal, Canada, 19–23 June 1993. IEEE Computer Society Press. 2.2

[28] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 2.2, 4.3, 4.3

[29] Gordon Plotkin and Martín Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, volume 664 of *Lecture Notes in Comput. Sci.*, pages 361–375. Springer, Berlin, 1993. 1.3, 4.1, 4.2, 6

[30] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983. 1.3, 2.1, 5.1, 5.4

[31] J.C. Reynolds. Polymorphism is not set-theoretic. In G. Kahn, D. B. MacQueen, and G. D. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 145–156. Springer-Verlag, 1984. 2.1

[32] J.C. Reynolds and G.D. Plotkin. On functors expressible in the polymorphic typed lambda calculus. In Gérard Huet, editor, *Logical Foundations of Functional Programming*, chapter 7, pages 127–151. Addison-Wesley, 1990. 2.1

[33] E.P. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In S. Abramsky, editor, *Proc. 9th Symposium in Logic in Computer Science*, pages 364–371, Paris, 1994. I.E.E.E. Computer Society. 3.1, 4.4

[34] J.M.E. Hyland E.M. Robinson and G. Rosolini. Algebraic types in PER models. In M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics. 5th Interational Conference*, volume 442 of *Lecture Notes in Computer Science*, pages 333–350, Tulane University, New Orleans, Louisiana, USA, March/April 1989. Spinger-Verlag. 5.4

[35] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. 3.1, 4.5, 6

[36] I. Rummelhoff. Polynat in PER-models. *Theoretical Computer Science*, 316(1–3):215–224, May 2004. 5.4

[37] C. Strachey. Fundamental concepts in programming languages. Lecture Notes, International Summer School in Computer Programming, Copenhagen, August 1967. 1

[38] T. Streicher. A relational characterisation of syntactic definability in models of system f. Unpublished Manuscript, 1998. 7.1

[39] P. Wadler. Theorems for free! In *4'th Symposium on Functional Programming Languages and Computer Architecture, ACM, London*, September 1989. 4.1

[40] P. Wadler. The Girard-Reynolds isomorphism (second edition). Manuscript, March 2004. 5.4

Categorical Models for Abadi-Plotkin's Logic for Parametricity

Lars Birkedal
Rasmus Ejlers Møgelberg

Abstract

We propose a new category-theoretic formulation of relational parametricity based on a logic for reasoning about parametricity given by Abadi and Plotkin [12]. The logic can be used to reason about parametric models, such that we may prove consequences of parametricity that to our knowledge have not been proved before for existing category-theoretic notions of relational parametricity. We provide examples of parametric models and we describe a way of constructing parametric models from given models of the second-order lambda calculus.

Contents

1 Introduction

The notion of parametricity for models of polymorphic type theories intuitively states that a function of polymorphic type behaves the same way on all type instances. Reynolds [13] discovered that parametricity is central for modeling data abstraction and proving representation independence results. The idea is that a client of an abstract data type is modeled as a polymorphic function; parametricity then guarantees that the client cannot distinguish between different implementations of the abstract data type. Reynolds also observed that parametricity can be used for encoding (inductive and coinductive) data types. See [20, 8] for expository introductions.

In 1983 Reynolds gave a precise formulation of parametricity called relational parametricity for set-theoretic models [13]. It basically states that a term of polymorphic type preserves relations between types: if term u has type $\prod \alpha \colon \mathsf{Type}.\, \sigma$ and $R \colon \mathsf{Rel}(\tau, \tau')$ is a relation between τ and τ', then

$$u(\tau)(\sigma[R])u(\tau'),$$

where $\sigma[R]$ is a relational interpretation of the type σ defined inductively over the structure of σ. Equivalently, parametricity could be defined as the identity extension property: for all terms u, v of type $\sigma(\vec{\alpha})$,

$$u(\sigma[\vec{eq_\alpha}])v \iff u = v.$$

However, Reynolds himself later proved that set-theoretic models do not exist [14] in classical set-theory (it was later discovered that set theoretic models do exist in some models of intuitionistic set theory [10, 9]). In 1992 Ma and Reynolds [6] then gave a new formulation of parametricity phrased in terms of more general models (PL-categories of Seely [18]). One may formulate Ma and Reynolds' notion in the language of λ_2-fibrations[1] as follows. The fibration $E \to B$ is parametric with respect to a given logic on E if there exist a reflexive graph of λ_2-fibrations, whose restriction to the fibres over the terminal object is the reflexive graph

$$\begin{tikzcd} E_1 \arrow[r, shift left=1.5ex, leftarrow] \arrow[r] \arrow[r, shift right=1.5ex, leftarrow] & \mathbf{LR}(E_1) \end{tikzcd}$$

of logical relations with domain, codomain maps and the middle map mapping a type to the identity on that type. (See [6, 5] for more details.)

In recent work by Birkedal and Rosolini on parametric domain-theoretic models it became clear that this is not the right categorical formulation of parametricity: it appears that the definition does not allow one to prove the expected consequences of parametricity such as data abstraction and the encoding of data types. Indeed, these consequences have only been proved for specific models, see, e.g., [20, 3], using specific properties of the models.

In this article we propose a new category-theoretic formulation of parametricity, called a *parametric APL-structure*, which *does* allow one to prove the expected properties of parametricity in general. We build upon a logic for reasoning about parametricity given by Abadi and Plotkin [12]. In this logic one can formulate parametricity as a schema and prove the expected consequences of parametricity. An APL-structure is a category-theoretic model of Abadi and Plotkin's logic, for which we prove soundness and completeness, thereby answering a question posed in [12, Page 5]. Each APL-structure contains a model of the second-order lambda calculus, which we may reason about using the logic.

We also provide a completion process that given an internal model of λ_2 (see [4, 15]) produces a parametric APL-structure. In special cases, the λ_2-fibration of this APL-structure is the one obtained in [15] and thus

[1] A λ_2-fibration is a fibration with enough properties to model second-order lambda calculus, see, e.g., [5].

we prove that the models obtained in [15] in fact satisfy the consequences of parametricity (as expected, but not shown in the literature before).

The consequences of parametricity proved earlier for specific models [3, 20, 1] all seem to use well-pointedness, i.e., the property, that morphisms $f : A \to B$ are determined by their values on global elements $a : 1 \to A$. For parametric APL-structures, we do not need to use well-pointedness to prove the expected consequences of parametricity. Loosely speaking, the point is that our notion of parametric APL-structure includes an appropriate extensional logic to reason with. In *loc. cit.*, the ambient world of set theory is used as the logic and thus extensionality there amounts to asking for well-pointedness. We provide a family of concrete parametric APL-structures, including non-well pointed ones. Thus parametricity *is* useful for proving consequences also for non-well-pointed models.

In subsequent papers we will show how to modify the parametric completion process to produce domain-theoretic parametric models and how to extend the notion of APL-structure to include models of polymorphic *linear* lambda calculus [11].

The remainder of the paper is organized as follows. In Section 2, we recall Abadi and Plotkin's logic. The reader is warned that our version of the logic is slightly different from the one described in [12]. In Section 3 we define the notion of an APL-structure. We prove soundness and completeness with respect to Abadi and Plotkin's logic in sections 3.1 and 3.2. Section 4 defines the internal language of an APL-structure and we define the notion of a *parametric* APL-structure. We also demonstrate in Section 5 how to use the internal language to show consequences of parametricity in parametric APL-structures. Section 5 mainly contain proofs of well-known results in Abadi & Plotkin's logic. However, since these proofs are by no means trivial, and to our knowledge do not appear in the literature, and since we think they are of general interest, we include them here.

Section 6 contains a definition of a concrete parametric APL-structure, and we also mention a non-well-pointed parametric APL-structure. Section 7 contains a comparison of our notion of parametricity with the one defined by Ma & Reynolds [6]. The parametric completion process is described in Section 8. Since an internal model of λ_2 in a quasitopos has ambient logic corresponding to most of the constructions in Abadi & Plotkin's logic, there exists a natural APL-structure incorporating it, so we may formulate the question if this model is parametric. This is done in Section 9.

Appendix A contains definitions and theory concerning composable fibrations, i.e., pairs of fibrations such that the codomain of the first is the domain of the second. In particular, we study the case of fibrations $\mathbb{F} \to \mathbb{E} \to \mathbb{B}$ where $\mathbb{F} \to \mathbb{E}$ is a logic fibration, and we study what is needed for it to model quantification along vertical maps in $\mathbb{E}$ and quantification along maps in $\mathbb{B}$. The definitions of this appendix are used in the definition of an APL-structure.

Acknowledgments. We would like to acknowledge helpful discussions with Alex Simpson and Martin Hyland and the constructive comments of the two anonymous referees.

2 Abadi & Plotkin's logic

We first recall Abadi & Plotkin's logic for reasoning about parametricity, originally defined in [12]. We will use a slightly modified version of the logic.

Abadi & Plotkin's logic is basically a second-order logic on the second-order λ-calculus (λ_2). Thus we begin by calling to mind the second order λ-calculus (a more formal presentation can be found in e.g. [5]).

2.1 Second-order λ-calculus

Well-formed type expressions in second-order λ-calculus are expressions of the form:

$$\alpha_1\colon \mathsf{Type}, \ldots, \alpha_n\colon \mathsf{Type} \vdash \sigma\colon \mathsf{Type}$$

where σ is built up from the α_i's using products (1, $\sigma \times \tau$), arrows ($\sigma \to \tau$) and quantification over types. The latter means that if we have a type

$$\alpha_1\colon \mathsf{Type}, \ldots, \alpha_n\colon \mathsf{Type} \vdash \sigma\colon \mathsf{Type},$$

then we may form the type

$$\alpha_1\colon \mathsf{Type}, \ldots, \alpha_{i-1}\colon \mathsf{Type}, \alpha_{i+1}\colon \mathsf{Type}, \ldots, \alpha_n\colon \mathsf{Type} \vdash \prod \alpha_i\colon \mathsf{Type}.\, \sigma\colon \mathsf{Type}$$

We do not allow repetitions in the list of α's, and we call this list the kind context. It is often denoted simply Ξ or $\vec{\alpha}$. We use σ, τ, ω to range over the set of types.

The terms in λ_2 are of the form:

$$\Xi \mid x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n \vdash t\colon \tau$$

where the σ_i and τ are well-formed types in the kind context Ξ. The list of x's is called the type context and is often denoted Γ. As for kind contexts we do not accept repetition in type contexts.

The grammar for raw terms is:

$$t ::= x \mid \lambda x\colon \sigma.t \mid t(t) \mid \star \mid \langle t, t\rangle \mid \pi t \mid \pi' t \mid \Lambda\alpha\colon \mathsf{Type}.\, t \mid t(\sigma)$$

corresponding to variables, λ-abstraction, function applications, an element of unit type, pairing and projections on product types and second-order λ-abstractions and type applications. We use s, t, u to range over the set of terms, and as usual we consider α-equivalent terms equal. Most of the formation rules are well known from the simply-typed λ-calculus; here we just recall the two additional rules for type abstraction and type application:

$$\frac{\Xi, \alpha\colon \mathsf{Type} \mid \Gamma \vdash t\colon \sigma}{\Xi \mid \Gamma \vdash \Lambda\alpha\colon \mathsf{Type}.\, t\colon \prod \alpha\colon \mathsf{Type}.\, \sigma}\ \Xi \mid \Gamma \text{ is well-formed}$$

$$\frac{\Xi \mid \Gamma \vdash t\colon \prod \alpha\colon \mathsf{Type}.\, \sigma \qquad \Xi \vdash \tau\colon \mathsf{Type}}{\Xi \mid \Gamma \vdash t(\tau)\colon \sigma[\tau/\alpha]}$$

What we have described above is called the *pure* second-order λ-calculus. In general we will consider second-order λ-calculi based on polymorphic signatures [5, 8.1.1]. Informally one may think of such a calculus as the pure second-order λ-calculus with added type-constants and term-constants. For instance one may have a constant type for integers or a constant type for lists $\alpha \vdash \mathit{lists}(\alpha)\colon \mathsf{Type}$. We will be particularly interested in the internal language of a λ_2-fibration (see Section 3) which in general will be a non-pure calculus.

2.1.1 Equality

We consider an equality theory on second-order λ-calculus called *external* equality. It is the least equivalence relation given by the rules in Figure 1.

$$\frac{\Xi \mid \Gamma, x\colon \sigma \vdash t\colon \tau \qquad \Xi \mid \Gamma \vdash u\colon \sigma}{\Xi \mid \Gamma \vdash (\lambda x\colon \sigma.\, t)u = t[u/x]}\ \beta\text{-reduction}$$

$$\frac{\Xi, \alpha \mid \Gamma \vdash t\colon \tau \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma \text{well-formed}}{\Xi \mid \Gamma \vdash (\Lambda\alpha\colon \mathsf{Type}.\, t)\sigma = t[\sigma/\alpha]}\ \beta\text{-reduction}$$

$$\frac{\Xi \mid \Gamma \vdash t\colon \sigma \to \tau}{\Xi \mid \Gamma \vdash \lambda x\colon \sigma.\, (tx) = t}\ \eta\text{-reduction}$$

$$\frac{\Xi \mid \Gamma \vdash t\colon \prod \alpha\colon \mathsf{Type}.\, \sigma}{\Xi \mid \Gamma \vdash \Lambda\alpha\colon \mathsf{Type}.\, (t\alpha) = t}\ \eta\text{-reduction}$$

$$\frac{\Xi \mid \Gamma \vdash t\colon \sigma \qquad \Xi \mid \Gamma \vdash u\colon \tau}{\Xi \mid \Gamma \vdash \pi\langle t, u\rangle = t} \quad \frac{\Xi \mid \Gamma \vdash t\colon \sigma \qquad \Xi \mid \Gamma \vdash u\colon \tau}{\Xi \mid \Gamma \vdash \pi'\langle t, u\rangle = u}$$

$$\frac{\Xi \mid \Gamma \vdash t\colon \sigma \times \tau}{\Xi \mid \Gamma \vdash \langle \pi t, \pi' t\rangle = t} \quad \frac{\Xi \mid \Gamma \vdash t\colon 1}{\Xi \mid \Gamma \vdash t = \star}$$

$$\frac{\Xi \mid \Gamma \vdash t = t'\colon \sigma \qquad \Xi \mid \Gamma, x\colon \sigma \vdash u\colon \tau}{\Xi \mid \Gamma \vdash u[t/x] = u[t'/x]}\ \text{replacement}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma \vdash t = s\colon \tau}{\Xi \mid \Gamma \vdash \lambda x\colon \sigma.\, t = \lambda x\colon \sigma.\, s} \quad \frac{\Xi, \alpha \mid \Gamma \vdash t = s \quad \Xi \mid \Gamma \text{ well-formed}}{\Xi \mid \Gamma \vdash \Lambda\alpha.\, t = \Lambda\alpha.\, s}$$

Figure 1: Rules for external equality

2.2 The logic

Abadi & Plotkin's logic can be built on top of any second-order lambda calculus (based on any polymorphic signature), so in the following we will assume that we are given one such.

Formulas of Abadi & Plotkin's logic live in contexts of elements of λ_2 and relations on types of λ_2. The contexts look like

$$\Xi \mid \Gamma \mid R_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, R_n\colon \mathsf{Rel}(\tau_n, \tau_n'),$$

where $\Xi \mid \Gamma$ is a context of second-order λ-calculus and the τ_i and τ_i' are well-formed types in context Ξ, for all i. The list of R's is called the relational context and is often denoted Θ. In this context as in the other contexts we do not accept repetitions of variable names. It is important to notice that the relational and type contexts are independent of each other in the sense that one does not affect whether the other is well-formed.

Formulas are given by the syntax:

$$\begin{aligned}\phi ::= \ & (t =_\sigma u) \mid \rho(t, u) \mid \ \phi \supset \psi \mid \bot \mid \top \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall \alpha\colon \mathsf{Type}.\, \phi \mid \\ & \forall x\colon \sigma.\, \phi \mid \forall R\colon \mathsf{Rel}(\sigma, \tau).\, \phi \mid \exists \alpha\colon \mathsf{Type}.\, \phi \mid \exists x\colon \sigma.\, \phi \mid \exists R\colon \mathsf{Rel}(\sigma, \tau).\, \phi,\end{aligned}$$

where ρ is a definable relation (to be discussed below).

In the following we give formation rules for the above. First we have internal equality

$$\frac{\Xi \mid \Gamma \vdash t\colon \sigma \qquad \Xi \mid \Gamma \vdash u\colon \sigma}{\Xi \mid \Gamma \mid \Theta \vdash (t =_\sigma u)\colon \mathsf{Prop}}$$

Notice here the notational difference between $t = u$ and $t =_\sigma u$. The former denotes *external* equality and the latter is a formula in the logic. The rules for $\supset$, $\vee$ and $\wedge$ are the usual ones. $\top$, $\perp$ are formulas in any context.

We have the formation rules for universal quantification:

$$\frac{\Xi \mid \Gamma, x\colon \sigma, \Gamma' \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma, \Gamma' \mid \Theta \vdash \forall x\colon \sigma.\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau), \Theta' \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta, \Theta' \vdash \forall R\colon \mathsf{Rel}(\sigma,\tau).\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi, \alpha, \Xi' \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi, \Xi' \mid \Gamma \mid \Theta \vdash \forall \alpha\colon \mathsf{Type}.\, \phi\colon \mathsf{Prop}}\ \Xi, \Xi' \mid \Gamma \mid \Theta \text{ is well-formed}$$

The same formation rules apply to the existential quantifier.

2.3 Definable relations

Definable relations are given by the grammar:

$$\rho ::= R \mid (x\colon \sigma, y\colon \tau).\phi \mid \sigma[\vec{\rho}].$$

A definable relation ρ always has a domain and a codomain, and we write $\rho\colon \mathsf{Rel}(\sigma,\tau)$ to denote that ρ has domain σ and codomain τ. There are 3 rules for this judgement. The first two are

$$\frac{}{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau), \Theta' \vdash R\colon \mathsf{Rel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi\colon \mathsf{Rel}(\sigma,\tau).}$$

In the second rule above the variables x, y become bound in ϕ. For example, we have the equality relation *eq*$_\sigma$ defined as $(x\colon \sigma, y\colon \sigma).\, x =_\sigma y$ and the graph relation of a function $\langle f \rangle = (x\colon \sigma, y\colon \tau).\, fx =_\tau y$ if $f\colon \sigma \to \tau$.

The last rule for definable relations is

$$\frac{\alpha_1, \ldots, \alpha_n \vdash \sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, \rho_n\colon \mathsf{Rel}(\tau_n, \tau_n')}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))}.$$

The notation is a bit ambiguous, since by $\sigma[\vec{\rho}]$ we mean to substitute each ρ_i for α_i in σ, and so the order of the α's and the ρ's is important. A more precise notation would have been $\sigma[\rho_1/\alpha_1, \ldots, \rho_n/\alpha_n]$, but we choose to use the more convenient $\sigma[\vec{\rho}]$.

Observe that $\sigma[\vec{\rho}]$ is a syntactic construction and is not obtained by substitution. In [12] $\sigma[\vec{\rho}]$ is defined inductively from the structure of σ, but in our case this is not enough, since we will need to form $\sigma[\vec{\rho}]$ for type constants σ in Section 4. The inductive definition of [12] is reflected in the rules (12)-(15) below. We call $\sigma[\vec{\rho}]$ the *relational interpretation of the type* σ.

If $\rho\colon \mathsf{Rel}(\sigma,\tau)$ is a definable relation, we may apply it to terms of the right types. This gives the last formation rule for formulas

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau) \quad \Xi \mid \Gamma \vdash t\colon \sigma, u\colon \tau}{\Xi \mid \Gamma \mid \Theta \vdash \rho(t,u)\colon \mathsf{Prop}.}$$

We will also write $t\rho u$ for $\rho(t,u)$.

Lemma 2.1. *Suppose* $\Xi \mid \Gamma \vdash \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \vdash \phi\colon \mathsf{Prop}$ *and* $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)$ *are well-formed. Then*

$$\Xi \mid \Gamma \mid \Theta \vdash \phi[\rho/R]\colon \mathsf{Prop}$$

is well-formed.

Proof. Easy induction on the structure of ϕ. □

Remark 2.2. *Abadi & Plotkin's logic is designed for reasoning about binary relational parametricity. For reasoning about other arities of parametricity (such as unary parametricity), one can easily replace binary relations in the logic by relations of other arities. In the case of unary parametricity, for example, one would then have an interpretation of types as predicates. See also [19, 21]*

We introduce the short notation $\rho \equiv \rho'$ for definable relations $\rho\colon \mathsf{Rel}(\sigma,\tau)$, $\rho'\colon \mathsf{Rel}(\sigma,\tau)$ as

$$\forall x\colon \sigma, y\colon \tau.\, \rho(x,y) \supset\subset \rho'(x,y).$$

Notice that we use $\supset\subset$ for biimplication.

We can take exponents, products and universal quantification of relations. These constructions will turn out to define categorical exponents, products and quantification in a category of relations (see Lemma 3.7). For now, the reader should just consider the next three definitions as shorthand notation.

If $\rho\colon \mathsf{Rel}(\sigma,\tau)$ and $\rho'\colon \mathsf{Rel}(\sigma',\tau')$ we may define a definable relation:

$$(\rho \to \rho')\colon \mathsf{Rel}((\sigma \to \sigma'),(\tau \to \tau'))$$

as

$$\rho \to \rho' = (f\colon \sigma \to \sigma', g\colon \tau \to \tau').\, \forall x\colon \sigma.\, \forall y\colon \tau.\, (x\rho y \supset (fx)\rho'(gy))$$

We may also take the product of ρ and ρ':

$$\rho \times \rho'\colon \mathsf{Rel}((\sigma \times \sigma'),(\tau \times \tau'))$$

as

$$\rho \times \rho' = (x\colon \sigma \times \sigma', y\colon \tau \times \tau').\, (\pi x)\rho(\pi y) \wedge (\pi' x)\rho'(\pi' y)$$

If

$$\Xi, \alpha, \beta \mid \Gamma \mid \ \Theta, R\colon \mathsf{Rel}(\alpha,\beta) \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)$$

is well-formed and $\Xi \mid \Gamma \mid \Theta$ and $\Xi, \alpha \vdash \sigma\colon \mathsf{Type}$ and $\Xi, \beta \vdash \tau\colon \mathsf{Type}$ we may define:

$$\Xi \mid \Gamma \mid \Theta \vdash \forall(\alpha,\beta,R\colon \mathsf{Rel}(\alpha,\beta)).\, \rho\colon \mathsf{Rel}((\textstyle\prod \alpha\colon \mathsf{Type}.\, \sigma),(\prod \beta\colon \mathsf{Type}.\, \tau))$$

as

$$(t\colon \textstyle\prod \alpha\colon \mathsf{Type}.\, \sigma, u\colon \prod \beta\colon \mathsf{Type}.\, \tau).\, \forall \alpha,\beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{Rel}(\alpha,\beta).\, (t\alpha)\rho(u\beta).$$

$$\Xi\colon \mathsf{Ctx} \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma\colon \mathsf{Ctx}$$

$$\Xi \mid \Theta\colon \mathsf{Ctx} \qquad \Xi \mid \Gamma \vdash t\colon \sigma \qquad \Xi \mid \Gamma \vdash t = u$$

$$\Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop} \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \mid \phi_1,\ldots,\phi_n \vdash \psi$$

Figure 2: Types of judgements

2.4 The axioms

Figure 2 sums up the types of judgements we have in the logic. The last judgement in the figure says that in the given context, the conjunction of the formulas $\phi_1,\ldots,\phi_n$ implies ψ.

Having specified the language of Abadi & Plotkin's logic, it is time to specify the axioms and the rules of the logic. We have all the axioms of propositional logic plus the rules specified below.

We have rules for $\forall$-quantification:

$$\frac{\Xi,\alpha \mid \Gamma \mid \Theta \mid \Phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall\alpha\colon \mathsf{Type}.\psi}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi \tag{1}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \Phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall x\colon \sigma.\psi}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi \tag{2}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau,\tau') \mid \Phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall R\colon \mathsf{Rel}(\tau,\tau').\psi}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi \tag{3}$$

The double bars mean that these are double rules, i.e., the condition on the bottom implies the one on top and vice versa.

Rules for $\exists$-quantification:

$$\frac{\Xi,\alpha \mid \Gamma \mid \Theta \mid \phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \exists\alpha\colon \mathsf{Type}.\phi \vdash \psi}\ \Xi \mid \Gamma \mid \Theta \vdash \psi \tag{4}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \exists x\colon \sigma.\phi \vdash \psi}\ \Xi \mid \Gamma \mid \Theta \vdash \psi \tag{5}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau,\tau') \mid \phi \vdash \psi}{\Xi \mid \Gamma \mid \Theta \mid \exists R\colon \mathsf{Rel}(\tau,\tau').\phi \vdash \psi}\ \Xi \mid \Gamma \mid \Theta \vdash \psi \tag{6}$$

We have substitution rules

$$\frac{\Xi,\alpha \mid \Gamma \mid \Theta \mid \Psi \vdash \phi \qquad \Xi \vdash \sigma\colon \mathsf{Type}}{\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \mid \Psi[\sigma/\alpha] \vdash \phi[\sigma/\alpha]} \tag{7}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \Psi \vdash \phi \qquad \Xi \mid \Gamma \vdash t\colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \Psi[t/x] \vdash \phi[t/x]} \tag{8}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \mid \Psi \vdash \phi \qquad \Xi \mid \ \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)}{\Xi \mid \Gamma \mid \Theta \mid \ \Psi[\rho/R] \vdash \phi[\rho/R]} \tag{9}$$

The *substitution* axiom:

$$\begin{aligned}\Xi \mid \Gamma \mid \Theta \mid \top \vdash \forall \alpha, \beta \colon \mathsf{Type}. \forall x, x' \colon \alpha. \forall y, y' \colon \beta. \forall R \colon \mathsf{Rel}(\alpha, \beta). \\ R(x, y) \wedge x =_\alpha x' \wedge y =_\beta y' \supset R(x', y')\end{aligned} \tag{10}$$

External equality implies internal equality:

$$\frac{\Xi \mid \Gamma \vdash t = u \colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash t =_\sigma u} \tag{11}$$

We omit the obvious rules stating that internal equality is an equivalence relation. The following rules concern the interpretation of types as relations.

$$\frac{}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \forall x, y \colon 1.\, x 1 y} \tag{12}$$

$$\frac{\vec{\alpha} \vdash \alpha_i \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho} \colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \alpha_i[\vec{\rho}] \equiv \rho_i} \tag{13}$$

$$\frac{\vec{\alpha} \vdash \sigma \to \sigma' \qquad \Xi \mid \Theta \vdash \vec{\rho} \colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \to \sigma')[\vec{\rho}] \equiv (\sigma[\vec{\rho}] \to \sigma'[\vec{\rho}])} \tag{14}$$

$$\frac{\vec{\alpha} \vdash \prod \beta.\, \sigma(\vec{\alpha}, \beta) \qquad \Xi \mid \Theta \vdash \vec{\rho} \colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\forall \beta.\, \sigma(\vec{\alpha}, \beta))[\vec{\rho}] \equiv \forall (\beta, \beta', R \colon \mathsf{Rel}(\beta, \beta')).\, \sigma[\vec{\rho}, R])} \tag{15}$$

Finally we have

$$\frac{\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \phi \colon \mathsf{Prop} \quad \Xi \mid \Gamma \vdash t \colon \sigma, u \colon \tau}{\Xi \mid \Gamma \vdash ((x \colon \sigma, y \colon \tau).\, \phi)(t, u) \supset\subset \phi[t, u / x, y].} \tag{16}$$

Using this rule, we may prove a bijective correspondence between definable relations and propositions with two free variables considered up to provable equivalence. The bijection maps a definable relation ρ to the formula $\rho(x, y)$ with free variables x, y and a formula ϕ with free variables x, y to the definable relation $(x, y).\, \phi$.

Lemma 2.3. *Suppose* $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)$ *and* $\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \phi \colon \mathsf{Prop}$. *Then*

$$\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \mid \top \vdash \phi \supset\subset ((x \colon \sigma, y \colon \tau).\, \phi)(x, y)$$

and

$$\Xi \mid \Gamma \mid \Theta \mid \top \vdash \rho \equiv (x \colon \sigma, y \colon \tau).\, \rho(x, y).$$

Proof. The first statement above is just a reformulation of (16), and for the second we need to prove that

$$\forall x \colon \sigma, y \colon \tau.\, ((x \colon \sigma, y \colon \tau).\, \rho(x, y))(x, y) \ \supset\subset \rho(x, y)$$

which is also an easy consequence of (16). □

We would also like to mention the extensionality schemes:

$$(\forall x\colon \sigma.\, t\,x =_\tau u\,x) \supset t =_{\sigma\to\tau} u$$
$$(\forall \alpha\colon \mathsf{Type}.\, t\,\alpha =_\tau u\,\alpha) \supset t =_{\prod \alpha\colon \mathsf{Type}.\tau} u.$$

These are taken as axioms in [12], but we shall not take these as axioms as we would like to be able to talk about models that are not necessarily extensional.

Lemma 2.4. *The substitution axiom above implies the* replacement *rule:*

$$\frac{\Xi \mid \Gamma \mid \Theta \mid \Phi \vdash t =_\sigma t' \qquad \Xi \mid \Gamma, x\colon \sigma \vdash u\colon \tau}{\Xi \mid \Gamma \mid \Theta \mid \Phi \vdash u[t/x] =_\tau u[t'/x]}$$

Proof. Instantiate the substitution axiom with the definable relation

$$\rho = (y\colon \sigma, z\colon \sigma).\, u[y/x] =_\tau u[z/x].$$

Clearly $\Phi \vdash \rho(t,t)$, so since $t =_\sigma t'$, we have $\Phi \vdash \rho(t,t')$ as desired. □

Lemma 2.5 (Weakening, Exchange). *If $\Xi \mid \Gamma \mid \Theta \mid \Psi \vdash \phi$ is provable in the logic, and if further $\Xi' \mid \Gamma' \mid \Theta'$ is a context obtained from $\Xi \mid \Gamma \mid \Theta$ by permuting the order of the variables in the contexts, and possibly adding variables, then*

$$\Xi' \mid \Gamma' \mid \Theta' \mid \Psi \vdash \phi$$

is also provable in the logic.

3 APL-structures

In this section we define the notion of an APL-structure, which is basically a category-theoretic formulation of a model of Abadi & Plotkin's logic. We also show how to interpret the logic in an APL-structure. We use the definitions and results of Appendix A.

But first we recall the notion of a λ_2-fibration, which is basically a model of λ_2.

Definition 3.1. *A fibration $\mathbf{Type} \to \mathbf{Kind}$ is a* λ_2-fibration *if it is fibred cartesian closed, has a generic object $\Omega \in \mathbf{Kind}$, products in $\mathbf{Kind}$, and simple Ω-products, i.e., right adjoints $\prod_\pi$ to the reindexing functors π^* for projections $\pi\colon \Xi \times \Omega \to \Xi$.*

Remark 3.2. *In a λ_2 fibration, for a map $f\colon \Xi \to \Omega$ in $\mathbf{Kind}$, we will use the notation $\hat{f}$ to denote the object of $\mathbf{Type}_\Xi$ corresponding to f, and likewise for $\sigma \in \mathbf{Type}_\Xi$ we write $\hat{\sigma}\colon \Xi \to \Omega$ for the map corresponding to σ.*

Definition 3.3. *A **pre-APL-structure** consists of*

1. *Fibrations:*

$$\begin{tikzcd} & \mathbf{Prop} \arrow[d, "r"] \\ \mathbf{Type} \arrow[r, hook, "I"] \arrow[dr, "p"'] & \mathbf{Ctx} \arrow[d, "q"] \\ & \mathbf{Kind} \end{tikzcd}$$

where

- *p is a $\lambda 2$-fibration.*
- *q is a fibration with fibred products*
- *(r, q) is an indexed first-order logic fibration (Definition A.4) which has products and coproducts with respect to $\Xi \times \Omega \to \Xi$ in $\mathbf{Kind}$ (Definition A.5) where Ω is the generic object of p.*
- *I is a faithful product preserving map of fibrations.*

2. *a contravariant morphism of fibrations:*

$$\begin{tikzcd}
\mathbf{Type} \times_{\mathbf{Kind}} \mathbf{Type} \arrow[rr, "U"] \arrow[dr] & & \mathbf{Ctx} \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

3. *a family of bijections*

$$\Psi_\Xi : \mathrm{Hom}_{\mathbf{Ctx}_\Xi}(\xi, U(\sigma, \tau)) \to \mathrm{Obj}\left(\mathbf{Prop}_{\xi \times I(\sigma \times \tau)}\right)$$

for σ and τ in $\mathbf{Type}_\Xi$ and ξ in $\mathbf{Ctx}_\Xi$, which

- *is natural in the ξ, σ, τ*
- *commutes with reindexing functors; that is, if $\rho : \Xi' \to \Xi$ is a morphism in $\mathbf{Kind}$ and $u : \xi \to U(\sigma, \tau)$ is a morphism in $\mathbf{Ctx}_\Xi$, then*

$$\Psi_{\Xi'}(\rho^*(u)) = (\bar{\rho})^*(\Psi_\Xi(u))$$

where $\bar{\rho}$ is the cartesian lift of ρ.

Notice that Ψ is only defined on vertical morphisms.

By a contravariant functor of fibrations, we mean a functor of fibrations, which is contravariant in each fibre.

Remark 3.4. *Item 3 implies that $(U(1_\Xi, 1_\Xi))_{\Xi \in \mathbf{Kind}}$ is an indexed family of generic objects. If, on the other hand, we have an indexed family of generic objects $(\Sigma_\Xi)_{\Xi \in \mathbf{Kind}}$ and $\mathbf{Ctx}$ is cartesian closed, then we may define U to be $\Sigma^{- \times -}$ and thereby get items 2 and 3 for free. In general, however, $\mathbf{Ctx}$ will not be cartesian closed. In particular, in the syntactic model described below in the proof of completeness $\mathbf{Ctx}$ is not cartesian closed.*

Remark 3.5. *Below we will describe how the $U(\sigma, \tau)$ is used to model the object of relations from σ to τ. To model a version of Abadi & Plotkin's logic for unary or any other arity of parametricity as in Remark 2.2, the functor U should have corresponding arity and the domain and codomain of the bijection Ψ should be changed accordingly.*

We now explain how to interpret all of Abadi & Plotkin's logic, except for the relational interpretation of types, in a pre-APL-structure. First we recall the interpretation of λ_2 in a λ_2-fibration.

A type $\alpha_1 \ldots \alpha_n \vdash \alpha_i$ is interpreted as the object of $\mathbf{Type}$ over Ω^n corresponding to the i'th projection $\Omega^n \to \Omega$. For a type $\alpha_1 \ldots \alpha_n \vdash \sigma$, we have $[\![\prod \alpha_i.\, \sigma]\!] = \prod_\pi [\![\vec{\alpha} \vdash \sigma]\!]$, where π is the projection forgetting the i'th coordinate. Since each fibre of the λ_2-fibration is cartesian closed, we may interpret the constructions of the simply typed λ-calculus using fibrewise constructions.

If $\Xi, \alpha \mid \Gamma \vdash t\colon \tau$ is a term and $\Xi \vdash \Gamma$ is well-formed, then we may interpret the term $\Xi \mid \Gamma \vdash \Lambda\alpha.\, t\colon \prod \alpha.\, \tau$ as the morphism corresponding to $[\![\Xi, \alpha \mid \Gamma \vdash t\colon \tau]\!]$ under the adjunction $\pi^* \dashv \prod_\pi$.

To interpret $\Xi \mid \Gamma \vdash t\, \sigma$, notice that $[\![\Xi \vdash \sigma]\!]$ corresponds to a map

$$\widehat{[\![\Xi \vdash \sigma]\!]}\colon [\![\Xi]\!] \to \Omega.$$

The morphism $[\![\Xi \mid \Gamma \vdash t\colon \prod \alpha.\, \tau]\!]$ corresponds by the adjunction $\pi^* \dashv \prod_\pi$ to a morphism in the fibre over $[\![\Xi]\!] \times \Omega$. We reindex this morphism along

$$\langle id_{[\![\Xi]\!]}, \widehat{[\![\Xi \vdash \sigma]\!]}\rangle\colon [\![\Xi]\!] \to [\![\Xi]\!] \times \Omega$$

to get $[\![\Xi \mid \Gamma \vdash t\, \sigma]\!]$.

Relational contexts are interpreted in $\mathbf{Ctx}$ as:

$$[\![\Xi \mid R_1\colon \mathsf{Rel}(\sigma_1, \tau_1), \ldots, R_n\colon \mathsf{Rel}(\sigma_n, \tau_n)]\!] = U([\![\sigma_1]\!], [\![\tau_1]\!]) \times \ldots \times U([\![\sigma_n]\!], [\![\tau_n]\!]),$$

where $[\![\sigma_i]\!]$, $[\![\tau_i]\!]$ are the interpretations of the types in $\mathbf{Type}$ as described above.

We aim to define $[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!]$ as an object of $\mathbf{Prop}$ over $[\![\Xi \mid \Gamma \mid \Theta]\!]$, which we define to be

$$I([\![\Xi \mid \Gamma]\!]) \times [\![\Xi \mid \Theta]\!].$$

We proceed by induction on the structure of ϕ. We use the short notation $[\![\Xi \mid \Gamma \mid \Theta \vdash t\colon \tau]\!]$ for the composition

$$[\![\Xi \mid \Gamma \mid \Theta]\!] \xrightarrow{\pi} I([\![\Xi \mid \Gamma]\!]) \xrightarrow{I([\![\Xi \mid \Gamma \vdash t\colon \tau]\!])} I([\![\Xi \vdash \tau]\!]) \ ,$$

and we will in the following leave obvious isomorphisms involving products implicit.

If we define $\Delta_X\colon X \to X \times X$ to be the diagonal map, then

$$[\![\Xi \mid x\colon \sigma, y\colon \sigma \mid - \vdash x =_\sigma y\colon \mathsf{Prop}]\!] = \coprod\nolimits_{\Delta_{I([\![\sigma]\!])}}(\top)$$

and

$$[\![\Xi \mid \Gamma \mid \Theta \mid t =_\sigma u]\!] = \\ \langle [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash u]\!]\rangle^* [\![\Xi \mid x\colon \sigma, y\colon \sigma \mid - \vdash x =_\sigma y\colon \mathsf{Prop}]\!].$$

$\forall x\colon A.\phi$ and $\forall R\colon \mathsf{Rel}(\sigma, \tau).\phi$ are interpreted using right adjoints to reindexing functors related to the appropriate projections in $\mathbf{Ctx}$. Likewise $\exists x\colon A.\phi$ and $\exists R\colon \mathsf{Rel}(\sigma, \tau).\phi$ are interpreted using left adjoints to the same reindexing functors.

$\forall \alpha.\phi$ and $\exists \alpha.\phi$ are interpreted using respectively right and left adjoints to $\bar{\pi}^*$ where $\bar{\pi}$ is the lift of the projection $\pi : [\![\Xi, \alpha\colon \mathsf{Type}]\!] \to [\![\Xi]\!]$ in $\mathbf{Kind}$ to $\mathbf{Ctx}$. To be more precise, one may easily show that for $\Xi \mid \Gamma \mid \Theta$ wellformed $[\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] = \pi^*[\![\Xi \mid \Gamma \mid \Theta]\!]$ using the corresponding result for the interpretation of λ_2, and so the cartesian lift of π is a map:

$$\bar{\pi}\colon [\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma \mid \Theta]\!]$$

and we define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \forall \alpha.\, \phi]\!] = \prod\nolimits_{\bar{\pi}} [\![\Xi,\ \alpha \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where $\prod_{\bar{\pi}}$ is the right adjoint to $\bar{\pi}^*$.

Definable relations are interpreted as maps in $\mathbf{Ctx}$. To be more precise, a definable relation

$$\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma, \tau)$$

is interpreted as a morphism from $[\![\Xi \mid \Gamma \mid \Theta]\!]$ to $U([\![\sigma]\!], [\![\tau]\!])$. The definable relation

$$\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau), \Theta' \vdash R\colon \mathsf{Rel}(\sigma, \tau)$$

is interpreted as the projection. We define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi\colon \mathsf{Rel}(\sigma, \tau)]\!] = \Psi^{-1}[\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi]\!].$$

We define the interpretation of application of definable relations to terms as follows:

$$[\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x, y)]\!] = \Psi([\![\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma, \tau)]\!]).$$

Finally

$$\begin{gathered}[\![\Xi \mid \Gamma \mid \Theta \vdash \rho(t, u)]\!] = \\ \langle \pi, id, [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash u]\!], \pi' \rangle^* [\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x, y)]\!]\end{gathered}$$

where $\pi\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to I[\![\Xi \mid \Gamma]\!]$ and $\pi'\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid - \mid \Theta]\!]$ are the projections. As usual, we have left out some obvious isomorphisms here.

To interpret the relational interpretation of types we need a little more structure. First we consider a fibration

$$\mathbf{Relations} \to \mathbf{RelCtx},$$

that can be defined for every pre-APL-structure. $\mathbf{RelCtx}$ is defined as the pullback

$$\begin{tikzcd}
\mathbf{RelCtx} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Ctx} \arrow[d] \\
\mathbf{Kind} \times \mathbf{Kind} \arrow[r, "\times"] & \mathbf{Kind}
\end{tikzcd}$$

If Θ is an object of $\mathbf{RelCtx}$ projecting to $(\Xi, \Xi') \in \mathbf{Kind} \times \mathbf{Kind}$, we will write it as $\Xi, \Xi' \mid \Theta$. The fibre of $\mathbf{Relations}$ over $\Xi, \Xi' \mid \Theta$ is

objects Triples (σ, τ, ρ), where σ is an object in $\mathbf{Type}_{\Xi}$, τ is an object in $\mathbf{Type}_{\Xi'}$ and ρ is a map $\rho\colon \Theta \to U(\pi^*\sigma, (\pi')^*\tau)$, where π, π' are the projections out of $\Xi \times \Xi'$.

morphisms A morphism from (σ, τ, ρ) to (σ', τ', ρ') is a pair of morphisms (s, t), such that $s\colon \sigma \to \sigma'$ and $t\colon \tau \to \tau'$, and

$$\Psi(U(\pi^*t, (\pi')^*s) \circ \rho') \leq \Psi(\rho)$$

where the ordering refers to the fibrewise ordering on $\mathbf{Prop}$.

Reindexing (σ, τ, ρ) along a vertical map $\Theta' \to \Theta$ in $\mathbf{RelCtx}$ (vertical with respect to $\mathbf{Kind} \times \mathbf{Kind}$) is given by composition. Reindexing with respect to lifts of maps $(\omega, \omega')\colon (\Xi_1, \Xi_1') \to (\Xi_2, \Xi_2')$ is given by reindexing in $\mathbf{Ctx} \to \mathbf{Kind}$.

Remark 3.6. *In the internal language, objects of* $\mathbf{Relations}$ *are simply relations*

$$\Xi, \Xi' \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma(\Xi), \tau(\Xi')),$$

and a morphism from $\rho \colon \mathsf{Rel}(\sigma(\Xi), \tau(\Xi'))$ *to* $\rho' \colon \mathsf{Rel}(\sigma'(\Xi), \tau'(\Xi'))$ *is simply a pair of morphisms* $t \colon \sigma \to \sigma'$ *in* $\mathbf{Type}_\Xi$ *and* $s \colon \tau \to \tau'$ *in* $\mathbf{Type}_{\Xi'}$ *such that*

$$\forall x, y.\, \rho(x, y) \supset \rho'(t\, x, s\, y).$$

We clearly have two functors $\mathbf{RelCtx} \to \mathbf{Kind}$ defined by mapping (Ξ, Ξ', Θ) to Ξ and Ξ' respectively, and we also have two functors $\mathbf{Relations} \to \mathbf{Type}$ defined by mapping (ϕ, σ, τ) to σ and τ respectively.

Lemma 3.7. *The fibration* $\mathbf{Relations} \to \mathbf{RelCtx}$ *is a* λ_2*-fibration, and the maps mentioned above define a pair of maps of* λ_2 *fibrations*

$$\begin{tikzcd}
\mathbf{Type} \arrow[d] & \mathbf{Relations} \arrow[l, shift right, "\partial_0"'] \arrow[l, shift left, "\partial_1"] \arrow[d] \\
\mathbf{Kind} & \mathbf{RelCtx}. \arrow[l, shift right, "\partial_0"'] \arrow[l, shift left, "\partial_1"]
\end{tikzcd}$$

Proof. The category $\mathbf{RelCtx}$ has products:

$$(\Xi_1, \Xi_1', \Theta) \times (\Xi_2, \Xi_2', \Theta') = (\Xi_1 \times \Xi_2, \Xi_1' \times \Xi_2', (\pi, \pi)^*\Theta \times (\pi', \pi')^*\Theta').$$

where $(\pi, \pi) \colon (\Xi_1 \times \Xi_2, \Xi_1' \times \Xi_2') \to (\Xi_1, \Xi_1')$ is the projection, and (π', π') is the other evident projection. The fibration has a generic object $(\Omega, \Omega, U(\widehat{id_\Omega}, \widehat{id_\Omega}))$, since morphism into this from (Ξ, Ξ', Θ) in $\mathbf{RelCtx}$ consists of pairs of types $(f : \Xi \to \Omega, g : \Xi' \to \Omega)$ and vertical morphisms from Θ to $U(\hat{f}, \hat{g})$. These are exactly the objects of $\mathbf{Relations}$.

The constructions for fibred products, fibred exponents and simple Ω-products are simply the rules for products, exponents and universal quantification of relations in Abadi & Plotkin's logic formulated in the internal language of the model, which we will describe in Section 4. One can either interpret these constructions in the pre-APL-structure, and prove directly that these constructions have the desired properties, or one can use the fact that pre-APL-structures interpret these constructions soundly (Theorem 3.10) and reason in the internal logic.

Here we give the rest of the proof reasoning in the internal logic. Suppose $\rho \colon \mathsf{Rel}(\sigma, \tau)$ and $\rho' \colon \mathsf{Rel}(\sigma', \tau')$ and $\omega \colon \mathsf{Rel}(\sigma'', \tau'')$ are objects in some fibre of $\mathbf{Relations}$. Then a vertical morphism from ω to

$$\rho \times \rho' \colon \mathsf{Rel}((\sigma \times \sigma'), (\tau \times \tau')),$$

defined as

$$(x, x')\rho \times \rho'(y, y') = x\rho y \wedge x'\rho' y',$$

is a pair of maps $t : \sigma'' \to \sigma \times \sigma'$ and $u : \tau'' \to \tau \times \tau'$ such that

$$\forall x, y.\, x\omega y \supset \pi(tx)\rho\pi(uy) \wedge \pi'(tx)\rho'\pi'(uy),$$

which is the same as a pair of maps from ω into ρ and ρ' respectively.

Likewise maps from ω into

$$(\rho \to \rho') \colon \mathsf{Rel}((\sigma \to \sigma'), (\tau \to \tau')),$$

defined as

$$f(\rho \to \rho')g = \forall x\colon \sigma \forall y\colon \tau (x \rho y \supset (fx)\rho'(gy)),$$

are in one-to-one correspondence with maps from $\omega \times \rho$ to ρ'.

Given new relations $\Xi, \Xi' \mid \Theta \vdash \omega\colon \mathsf{Rel}(\sigma, \sigma')$ and

$$\Xi, \alpha; \Xi', \beta \mid \Theta, R\colon \mathsf{Rel}(\alpha, \beta) \vdash \rho\colon \mathsf{Rel}(\tau, \tau'),$$

we have defined

$$\Xi, \Xi' \mid \Theta \vdash \forall(\alpha, \beta, R\colon \mathsf{Rel}(\alpha, \beta)).\, \rho\colon \mathsf{Rel}((\textstyle\prod \alpha\colon \mathsf{Type}.\, \tau), (\prod \beta\colon \mathsf{Type}.\, \tau'))$$

as

$$(t\colon\ \textstyle\prod \alpha.\, \tau, \prod \beta.\, \tau').\, \forall \alpha, \beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{Rel}(\alpha, \beta).\, (t\alpha)\rho(u\beta).$$

We need to show that this defines a right adjoint to weakening. The idea is that the correspondence between maps will be the same as in $\mathbf{Type} \to \mathbf{Kind}$. In this fibration, the correspondence is given as follows, a map $\Xi, \alpha \mid - \vdash t\colon \sigma \to \tau$ with $\Xi \vdash \sigma\colon \mathsf{Type}$ corresponds to $\Xi \mid - \vdash \hat{t}\colon \sigma \to \prod \alpha.\, \tau$ where $\hat{t} = \lambda x\colon \sigma.\, \Lambda\alpha.\, (t\, x)$. We will show, that (t, u) preserves relations iff $(\hat{t}, \hat{u})$ does. It is clear that

$$\Xi, \alpha; \Xi', \beta \mid x : \sigma, y : \sigma' \mid \Theta, R\colon \mathsf{Rel}(\alpha, \beta) \mid x\omega y \vdash (tx)\rho(uy)$$

iff

$$\Xi, \Xi' \mid x : \sigma, y : \sigma' \mid \Theta \mid x\omega y \vdash \forall \alpha, \beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{Rel}(\alpha, \beta).\, (\hat{t}\, x\, \alpha)\rho(\hat{u}\, y\, \beta),$$

which establishes the bijective correspondence. □

Definition 3.8. *An* ***APL-structure*** *is a pre-APL-structure for which the graph of 3.7 can be extended to a reflexive graph of $\lambda 2$-fibrations*

$$\begin{tikzcd}
\mathbf{Type} \arrow[r, "J" description] \arrow[d] & \mathbf{Relations} \arrow[l, shift right=2, "\partial_0"'] \arrow[l, shift left=2, "\partial_1"] \arrow[d] \\
\mathbf{Kind} \arrow[r, "J" description] & \mathbf{RelCtx}, \arrow[l, shift right=2, "\partial_0"'] \arrow[l, shift left=2, "\partial_1"]
\end{tikzcd}$$

i.e., there exists a map J of λ_2-fibrations such that $\partial_0 J = id = \partial_1 J$.

Remark 3.9. *There is a functor from* $\mathbf{Relations}$ *to* $\mathbf{Prop}$ *mapping an object (σ, τ, ρ) to $\Psi(\rho)$. In the following we often use that functor implicitly.*

We need to show how to interpret the rule

$$\frac{\alpha_1, \ldots, \alpha_n \vdash \sigma(\vec{\alpha})\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, \rho_n\colon \mathsf{Rel}(\tau_n, \tau_n')}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))}$$

in an APL-structure.

Since J preserves products and generic objects, $J([\![\vec{\alpha} \vdash \sigma(\vec{\alpha})]\!])$ is a definable relation of the form

$$[\![\vec{\alpha}; \vec{\beta} \mid - \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha}, \vec{\beta}) \vdash J(\sigma)\colon \mathsf{Rel}(\sigma(\vec{\alpha}), \sigma(\vec{\beta}))]\!].$$

It thus makes sense to define

$$[\![\vec{\alpha}, \vec{\beta} \mid - \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\alpha}), \sigma(\vec{\beta}))]\!]$$

to be $J(\llbracket \vec{\alpha} \vdash \sigma(\vec{\alpha})\colon \mathsf{Type}\rrbracket)$, so now all we need to do is reindex this object. Given types $\Xi \vdash \vec{\tau}, \vec{\tau}'\colon \mathsf{Type}$, we define

$$\llbracket \Xi \mid - \mid \vec{R}\colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}') \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))\rrbracket$$

to be

$$\langle \widehat{\llbracket \Xi \vdash \vec{\tau}\rrbracket}, \widehat{\llbracket \Xi \vdash \vec{\tau}'\rrbracket}\rangle^* \llbracket \vec{\alpha}; \vec{\beta} \mid - \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\alpha}), \sigma(\vec{\beta}))\rrbracket.$$

Finally, given definable relations $\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}')$ we define

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))\rrbracket = \\ \llbracket \Xi \mid - \mid \vec{R}\colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}') \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))\rrbracket \circ \llbracket \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{Rel}(\vec{\tau}, \vec{\tau}')\rrbracket.$$

3.1 Soundness

We have now completed showing how to interpret all constructions of the language of Abadi and Plotkin's logic in APL-structures. We consider an implication $\Xi \mid \Gamma \mid \Theta \mid \phi_1, \dots, \phi_n \vdash \psi$ to hold in the model if

$$\bigwedge_i \llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi_i\rrbracket \vdash \llbracket \Xi \mid \Gamma \mid \Theta \vdash \psi\rrbracket,$$

where $\vdash$ above refers to the fibrewise ordering in $\mathbf{Prop}$.

Theorem 3.10 (Soundness). *In any APL-structure the interpretation defined above is sound with respect to the axioms and rules specified in Section 2.4, i.e., all axioms hold in the model, and for all rules, if the hypothesis holds in the model, then so does the conclusion. In any pre-APL structure the interpretation of the part of the logic excluding the relational interpretation of terms is sound.*

We will only prove the first part of Theorem 3.10, i.e., soundness for APL-structures. The proof of soundness for pre-APL structures is basically the same. For the proof we need the following lemmas:

Lemma 3.11. *If* $\Xi \mid \Gamma \vdash t\colon \sigma$ *then*

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi[t/x]\rrbracket = (I\langle id_{\llbracket \Xi \mid \Gamma\rrbracket}, \llbracket t\rrbracket\rangle \times id_{\llbracket \Xi \mid \Theta\rrbracket})^* \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\rrbracket$$

Proof. We will prove the statement of the lemma and the statement

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho[t/x]\colon \mathsf{Rel}(\tau, \tau')\rrbracket = \\ \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\tau, \tau')\rrbracket \circ (I\langle id_{\llbracket \Xi \mid \Gamma\rrbracket}, \llbracket t\rrbracket\rangle \times id_{\llbracket \Xi \mid \Theta\rrbracket}),$$

for all definable relations ρ, by simultaneous induction on the structure of ϕ and ρ. We only do a few cases and leave the rest to the reader.

Case $\rho = \sigma[\vec{\rho}]$:

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}][t/x]\rrbracket = \llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}[t/x]]\rrbracket = \llbracket \Xi \mid - \mid \vec{R} \vdash \sigma[\vec{R}]\rrbracket \circ \llbracket \vec{\rho}[t/x]\rrbracket$$

Since by induction $\llbracket \vec{\rho}[t/x]\rrbracket = \llbracket \vec{\rho}\rrbracket \circ (I\langle id_{\llbracket \Xi \mid \Gamma\rrbracket}, \llbracket t\rrbracket\rangle \times id_{\llbracket \Xi \mid \Theta\rrbracket}))$, we are done.

Case $\rho = (y\colon \tau, z\colon \tau').\, \phi$:

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho[t/x] \rrbracket = \Psi^{-1}(\llbracket \Xi \mid \Gamma, y\colon \tau, z\colon \tau' \mid \Theta \vdash \phi[t/x] \rrbracket),$$

which by induction is equal to

$$\Psi^{-1}(\langle \pi_{\llbracket \Gamma \rrbracket}, \llbracket t \rrbracket, \pi_{\llbracket y\colon \tau, z\colon \tau' \mid \Theta \rrbracket} \rangle^* \llbracket \Xi \mid \Gamma, x\colon \sigma, y\colon \tau, z\colon \tau' \mid \Theta \vdash \phi \rrbracket).$$

By naturality of Ψ this is equal to

$$\begin{gathered} \Psi^{-1}(\llbracket \Xi \mid \Gamma, x\colon \sigma, y\colon \tau, z\colon \tau' \mid \Theta \vdash \phi \rrbracket) \circ \langle \pi_{\llbracket \Gamma \rrbracket}, \llbracket t \rrbracket, \pi_{\llbracket \Theta \rrbracket} \rangle = \\ \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \rho \rrbracket) \circ \langle \pi_{\llbracket \Gamma \rrbracket}, \llbracket t \rrbracket, \pi_{\llbracket \Theta \rrbracket} \rangle \end{gathered}$$

as desired.

Case $\phi = \rho(u, s)$

Using naturality of Ψ as before, one can prove that

$$\begin{gathered} \llbracket \Xi \mid \Gamma, y\colon \tau, z\colon \tau' \mid \Theta \vdash \rho(y, z)[t/x] \rrbracket = \\ (I\langle id_{\llbracket \Xi \mid \Gamma, y\colon \tau, z\colon \tau' \rrbracket}, \llbracket t \rrbracket \rangle \times id_{\llbracket \Xi \mid \Theta \rrbracket})^* \llbracket \Xi \mid \Gamma, y\colon \tau, z\colon \tau', x\colon \sigma \mid \Theta \vdash \rho(y, z) \rrbracket. \end{gathered}$$

The general case follows from the fact that in a λ_2-fibration

$$\llbracket \Xi \mid \Gamma \vdash u[t/x] \rrbracket = \llbracket \Xi \mid \Gamma \vdash u \rrbracket \circ \langle id, \llbracket \Xi \mid \Gamma \vdash t \rrbracket \rangle.$$

Case $\phi = \forall \alpha\colon \mathsf{Type}.\, \psi$:

We need to show that

$$\begin{gathered} \llbracket \Xi \mid \Gamma \mid \Theta \vdash \forall \alpha\colon \mathsf{Type}.\, \psi[t/x] \rrbracket = \\ (I\langle id_{\llbracket \Xi \mid \Gamma \rrbracket}, \llbracket t \rrbracket \rangle \times id_{\llbracket \Xi \mid \Theta \rrbracket})^* \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \forall \alpha\colon \mathsf{Type}.\, \psi \rrbracket. \end{gathered}$$

Let $\overline{\pi}$ denote the cartesian lift of the projection $\llbracket \Xi, \alpha \rrbracket \to \llbracket \Xi \rrbracket$. Then by induction we have that the left hand side of the equation is

$$\textstyle\prod_{\overline{\pi}} (I\langle id_\Gamma, \llbracket t \rrbracket \rangle \times id_\Theta)^* \llbracket \Xi, \alpha \mid \Gamma, x\colon \sigma \mid \Theta \vdash \psi \rrbracket.$$

Consider the square

$$\begin{tikzcd}
\llbracket \Xi, \alpha \mid \Gamma \mid \Theta \rrbracket \arrow[r, "\overline{\pi}"] \arrow[d, "I\langle id_\Gamma, \llbracket t \rrbracket \rangle \times id_\Theta"'] & \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket \arrow[d, "I\langle id_\Gamma, \llbracket t \rrbracket \rangle \times id_\Theta"] \\
\llbracket \Xi, \alpha \mid \Gamma, x\colon \sigma \mid \Theta \rrbracket \arrow[r, "\overline{\pi}"] & \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \rrbracket.
\end{tikzcd}$$

This square commutes since $\overline{\pi}$ is a natural transformation from π^* to id, and it is a pullback by [5, Exercise 1.4.4]. The Beck-Chevalley condition relative to this square gives the desired result.

□

Lemma 3.12. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop}$*, then*

$$\llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi \rrbracket = \pi^* \llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi \rrbracket,$$

where $\pi\colon \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \rrbracket \to \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket$ *is the projection.*

Lemma 3.13. *If* $\Xi \vdash \sigma\colon \mathsf{Type}$ *then*

$$[\![\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \vdash \phi[\sigma/\alpha]]\!] = \overline{\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle}^{*} [\![\Xi, \alpha\colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where the vertical line in $\overline{\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle}$ *denotes the cartesian lift.*

Proof. Notice first that a corresponding reindexing lemma for interpretation of λ_2 in λ_2-fibrations tells us that

$$\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle^{*} [\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] = [\![\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha]]\!].$$

The rest of the proof is by induction over the structure of ϕ, and since it resembles the proof of Lemma 3.11 closely we leave it to the reader. □

Lemma 3.14. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi$ *then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!] = \overline{\pi}^{*}_{\Xi,\alpha\to\Xi} [\![\Xi, \alpha \mid \Gamma \mid \Theta \vdash \phi]\!]$$

Proof. The proof is almost the same as for Lemma 3.13. □

Lemma 3.15. *If* $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation, then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi[\rho/R]]\!] = (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\rho]\!]\rangle)^{*} [\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau, \tau') \vdash \phi]\!]$$

Proof. The lemma should be proved simultaneously with the statement

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \rho'[\rho/R]]\!] = [\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau, \tau') \vdash \rho']\!] \circ (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\rho]\!]\rangle)$$

for all definable relations ρ', by structural induction on ϕ and ρ'. We leave the proof to the reader, as it closely resembles the proof of (3.11).

□

Lemma 3.16. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop}$*, then*

$$[\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau) \vdash \phi]\!] = \pi^{*} [\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where $\pi\colon [\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau)]\!] \to [\![\Xi \mid \Gamma \mid \Theta]\!]$ *is the projection.*

We are now ready to prove soundness.

Proof of Theorem 3.10. The rules for quantification (1)- (6) follow directly from the fact that the interpretation of $\forall$ and $\exists$ are given by right, respectively left adjoints to weakening functors. The substitution rules (7) - (9) are sound by Lemmas 3.11, 3.13 and 3.15.

For the *substitution* axiom (10) we will only prove

$$[\![\alpha, \beta \mid x, x'\colon \alpha, y\colon \beta \mid R\colon \mathsf{Rel}(\alpha, \beta) \vdash x =_{\alpha} x']\!] \leq$$
$$[\![\alpha, \beta \mid x, x'\colon \alpha, y\colon \beta \mid R\colon \mathsf{Rel}(\alpha, \beta) \vdash R(x, y) \supset R(x', y)]\!].$$

Once this is done, the rest of the proof amounts to doing the same thing in the second variable. We will for readability write simply $[\![\alpha]\!], [\![\beta]\!], [\![R]\!]$ for $[\![\alpha, \beta \vdash \alpha]\!]$, $[\![\alpha, \beta \vdash \beta]\!]$, $[\![\alpha, \beta \mid - \mid R\colon \mathsf{Rel}(\alpha, \beta)]\!]$.

If we let $\pi_1, \pi_2, \pi_3, \pi_4$ denote the projections out of

$$\llbracket \alpha, \beta \mid x, x' \colon \alpha, y \colon \beta \mid R \colon \mathsf{Rel}(\alpha, \beta) \rrbracket = \\ \llbracket \alpha, \beta \vdash \alpha \rrbracket^2 \times \llbracket \alpha, \beta \vdash \beta \rrbracket \times U(\llbracket \alpha, \beta \vdash \alpha \rrbracket, \llbracket \alpha, \beta \vdash \beta \rrbracket)$$

we can formulate what we aim to prove as

$$\langle \pi_1, \pi_2 \rangle^* (\textstyle\coprod_{\Delta_{\llbracket \alpha \rrbracket}} (\top)) \leq \langle \pi_1, \pi_3 \rangle^* \Psi(id_{\llbracket R \rrbracket}) \supset \langle \pi_2, \pi_3 \rangle^* \Psi(id_{\llbracket R \rrbracket}),$$

where Δ denotes the diagonal map.

Using the Beck-Chevalley condition on the square

$$\begin{tikzcd}
\llbracket \alpha \rrbracket \times \llbracket \beta \rrbracket \times \llbracket R \rrbracket \arrow[r, "\Delta_{\llbracket \alpha \rrbracket} \times id"] \arrow[d, "\pi_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & \llbracket \alpha \rrbracket^2 \times \llbracket \beta \rrbracket \times \llbracket R \rrbracket \arrow[d, "\langle \pi_1, \pi_2 \rangle"] \\
\llbracket \alpha \rrbracket \arrow[r, "\Delta_{\llbracket \alpha \rrbracket}"] & \llbracket \alpha \rrbracket^2
\end{tikzcd}$$

we get

$$\langle \pi_1, \pi_2 \rangle^* (\textstyle\coprod_{\Delta_{\llbracket \alpha \rrbracket}} (\top)) = \textstyle\coprod_{\Delta_{\llbracket \alpha \rrbracket} \times id_{\llbracket \beta \rrbracket \times \llbracket R \rrbracket}} (\top).$$

Now the result follows from using the adjunction and the fact that

$$\langle \pi_1, \pi_3 \rangle \circ (\Delta_{\llbracket \alpha \rrbracket} \times id_{\llbracket \beta \rrbracket \times \llbracket R \rrbracket}) = \langle \pi_2, \pi_3 \rangle \circ (\Delta_{\llbracket \alpha \rrbracket} \times id_{\llbracket \beta \rrbracket \times \llbracket R \rrbracket}).$$

External equality implies internal equality (11) since the model of λ_2 included in the model is sound. Internal equality is clearly an equivalence relation.

The axioms concerning types as relations (12) - (15) follow from the fact that J is required to be a morphism of λ_2 fibrations and that the λ_2 structure in $\mathbf{Relations} \to \mathbf{RelCtx}$ is given by the interpretation of products and quantification of relations. For instance soundness of the (15) is proved as follows:

$$\llbracket \vec{\alpha}, \vec{\alpha}' \ \mid - \mid \vec{R} \colon \mathsf{Rel}(\vec{\alpha}, \vec{}\alpha') \vdash (\textstyle\prod \beta.\, \sigma)[\vec{R}] \rrbracket = \\ J(\llbracket \vec{\alpha} \vdash \textstyle\prod \beta.\, \sigma \rrbracket) = \\ \llbracket \vec{\alpha}, \vec{\alpha}' \mid \vec{R} \colon \mathsf{Rel}(\ \vec{\alpha}, \vec{\alpha}') \vdash (\forall \gamma, \gamma', S \colon \mathsf{Rel}(\gamma, \gamma')).\, \sigma[\vec{R}, S] \rrbracket$$

where the second equality holds since J preserves simple Ω-products.

Finally, to prove soundness of rule (16), it suffices to prove soundness of

$$\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \mid \top \vdash ((x \colon \sigma, y \colon \tau).\, \phi)(x, y) \supset\subset \phi,$$

but

$$\llbracket \Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash ((x \colon \sigma, y \colon \tau).\, \phi)(x, y) \rrbracket = \\ \Psi(\llbracket \Xi \mid \Gamma \mid \Theta \vdash (x \colon \sigma, y \colon \tau).\, \phi \rrbracket) = \\ \Psi \circ \Psi^{-1}(\llbracket \Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \phi \rrbracket) = \llbracket \Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \phi \rrbracket.$$

□

3.2 Completeness

The Soundness Theorem (3.10) allows us to reason about APL-structures using Abadi & Plotkin's logic. The Completeness Theorem below states that any formula that holds in all APL-structures, is provable in the logic. This allows us to reason about the logic using the class of APL-structures. However, since the APL-structure below is constructed from the logic, this does not say much. Instead, one should view the Completeness Theorem as stating that the class of APL-structures is not too restrictive; it completely describes the logic.

Theorem 3.17 (Completeness). *There exists an APL-structure with the property that any formula of Abadi & Plotkin's logic based on pure λ_2 that holds in the structure may be proved in the logic.*

Proof. We construct the APL-structure syntactically, giving the categories in question the same names as in the diagram of item 1 in Definition 3.3.

- The category $\mathbf{Kind}$ has sequences of the form $\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type}$ as objects, where we identify these contexts up to renaming (in other words, we may think of objects as natural numbers). A morphism from Ξ into $\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type}$ is a sequence of types $(\sigma_1, \ldots, \sigma_n)$ such that all σ_i are well-formed in context Ξ.
- Objects in the fibre of $\mathbf{Type}$ over Ξ are well-formed types in this context, where we identify types up to renaming of free type variables. Morphisms in this fibre from σ to τ are equivalence classes of terms t such that $\Xi \mid - \vdash t \colon \sigma \to \tau$ where we identify terms up to external equality. Reindexing with respect to morphisms in $\mathbf{Kind}$ is by substitution.
- The category $\mathbf{Ctx}$ has as objects in the fibre over Ξ well-formed contexts of Abadi & Plotkin's logic: $\Xi \mid \Gamma \mid \Theta$, where we again identify such contexts up to renaming of free type-variables. A vertical morphism from $\Xi \mid \Gamma \mid \Theta$ to $\Xi \mid \Gamma' \mid R_1 \colon \mathsf{Rel}(\sigma_1, \tau_1), \ldots, R_n \colon \mathsf{Rel}(\sigma_n, \tau_n)$ is a pair, consisting of a morphism $\Xi \mid \Gamma \to \Xi \mid \Gamma'$ in the sense of morphisms in $\mathbf{Type}$ and a sequence of definable relations $(\rho_1, \ldots, \rho_n)$ such that $\Xi \mid \Gamma \mid \Theta \vdash \rho_i \colon \mathsf{Rel}(\sigma_i, \tau_i)$. We identify two such morphisms represented by the same type morphism and the definable relations $(\rho_1, \ldots, \rho_n)$ and $(\rho'_1, \ldots, \rho'_n)$ if, for each i, $\rho_i \equiv \rho'_i$ is provable in the logic. one. Reindexing is by substitution.
- The fibre of the category $\mathbf{Prop}$ over a context $\Xi \mid \Gamma \mid \Theta$ has as objects formulas in that context, where we identify two formulas if they are provably equivalent. These are ordered by entailment in the logic. Reindexing is done by substitution, that is, reindexing with respect to lifts of morphisms from $\mathbf{Kind}$ is done by substitution in Kind-variables, whereas reindexing with respect to vertical maps in $\mathbf{Ctx}$ is by substitution in type variables and relational variables.

It is straightforward to verify that this structure satisfies item 1 of Definition 3.3. The only non-obvious thing to verify here is existence of products and coproducts in $\mathbf{Prop}$ with respect to vertical maps in $\mathbf{Ctx}$.

Suppose $(\vec{t}, \vec{\rho})$ represents a morphism from $\Xi \mid \vec{x} \colon \vec{\sigma} \mid \vec{R}$ to $\Xi \mid \vec{y} \colon \vec{\tau} \mid \vec{S}$. Then we can define the product functor in $\mathbf{Prop}$ by:

$$\textstyle\prod_{(\vec{t}, \vec{\rho})} (\Xi \mid \vec{x} \colon \vec{\sigma} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) = \\ \Xi \mid \vec{y} \colon \vec{\tau} \mid \vec{S} \vdash \forall \vec{x}. \forall \vec{R} (\vec{t}\vec{x} = \vec{y} \wedge (\vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S}) \supset \phi(\vec{x}, \vec{R})).$$

We define coproduct as:

$$\textstyle\coprod_{(\vec{t}, \vec{\rho})} (\Xi \mid \vec{x} \colon \vec{\sigma} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) = \\ \Xi \mid \vec{y} \colon \vec{\tau} \mid \vec{S} \vdash \exists \vec{x}. \exists \vec{R}. \vec{t}\vec{x} = \vec{y} \wedge \vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S} \wedge \phi(\vec{x}, \vec{R}).$$

The functor U of item 2 is defined as

$$U(\sigma,\tau) = R\colon \mathsf{Rel}(\sigma,\tau)$$

and

$$U(t\colon \sigma \to \sigma', u\colon \tau \to \tau') = \Xi \mid R\colon \mathsf{Rel}(\sigma',\tau') \vdash (x\colon \sigma, y\colon \tau).\, R(tx, uy)$$

The map Ψ maps a definable relation $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)$ to the proposition

$$\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x,y)\colon \mathsf{Prop},$$

which is a bijection by Lemma 2.3.

We have defined a pre-APL-structure. The category $\mathbf{RelCtx}$ obtained from this pre-APL structure has as objects $\vec{\alpha},\vec{\beta} \mid \Gamma \mid \Theta$. The fibre of $\mathbf{Relations}$ over an object $\vec{\alpha},\vec{\beta} \mid \Gamma \mid \Theta$ in $\mathbf{RelCtx}$ is:

Objects Equivalence classes of definable relations

$$\vec{\alpha},\vec{\beta} \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma(\vec{\alpha}),\tau(\vec{\beta})).$$

Morphisms A morphism from $\rho\colon \mathsf{Rel}(\sigma(\vec{\alpha}),\tau(\vec{\beta}))$ to $\rho'\colon \mathsf{Rel}(\sigma'(\vec{\alpha}),\tau'(\vec{\beta}))$ is a pair of morphisms $t : \sigma \to \sigma'$, $u : \tau \to \tau'$ such that it is provable that

$$\forall x\colon \sigma.\, \forall y\colon \tau.\, \rho(x,y) \supset \rho'(tx,uy).$$

In the reflexive graph of Lemma 3.7, the functor from $\mathbf{Kind}$ to $\mathbf{RelCtx}$ acts on objects as

$$\alpha_1,\dots,\alpha_n \mapsto \alpha_1,\dots,\alpha_n;\beta_1,\dots,\beta_n \mid R_1\colon \mathsf{Rel}(\alpha_1,\beta_1),\dots,R_n\colon \mathsf{Rel}(\alpha_n,\beta_n)$$

and it takes a morphism $\vec{\sigma} : \vec{\alpha} \to \vec{\alpha}'$ to the triple $(\vec{\sigma}(\alpha),\vec{\sigma}(\beta),\vec{\sigma}[\vec{R}])$. Notice that this defines a morphism since

$$\vec{\alpha},\vec{\beta} \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha},\vec{\beta}) \vdash \sigma_i[\vec{R}]\colon \mathsf{Rel}(\sigma_i(\vec{\alpha}),\sigma_i(\vec{\beta}))$$

This really defines the object part of the functor from $\mathbf{Type}$ to $\mathbf{Relations}$ since it must preserve $\lambda 2$-structure. So this functor takes a type $\vec{\alpha} \vdash \sigma$ to

$$\vec{\alpha};\vec{\beta} \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha},\vec{\beta}) \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\alpha}),\sigma(\vec{\beta})).$$

The functor maps a morphism $\vec{\alpha} \mid x\colon \sigma \vdash t\colon \tau$ to the pair $(\lambda x\colon \sigma.\, t, \lambda x\colon \sigma.\, t)$. This defines a morphism in $\mathbf{Relations}$ since the Logical Relations Lemma [12, Lemma 2] implies that

$$\vec{\alpha};\vec{\beta} \mid \vec{R}\colon \mathsf{Rel}(\vec{\alpha},\vec{\beta}) \mid x\colon \sigma(\vec{\alpha}), y\colon \sigma(\vec{\beta}) \vdash \sigma[\vec{R}](x,y) \supset \tau[\vec{R}](t, t[\beta/\alpha][y/x]).$$

One may easily verify that the functors above define a reflexive graph of $\lambda 2$-fibrations.

Now, by definition, a formula holds in this APL-structure iff it is provable in Abadi & Plotkin's logic. □

Remark 3.18. *The Completeness Theorem only states completeness for Abadi & Plotkin's logic based on the* pure λ_2*. The reason for this is that the proof uses the Logical Relations Lemma, which is proved in [12] by structural induction on terms. In the case of general calculi, one must know that the Logical Relations Lemma holds for term-constants in the language to be able to prove completeness.*

4 Parametric APL-structures

Given an APL-structure, we may consider the internal logic of the model (to be defined precisely below), and formulate parametricity as a schema in this logic. For technical reasons we will define parametric APL-structures as APL-structures not only satisfying the parametricity schema, but also extensionality and very strong equality (A.7). For parametric APL-structures, we can derive consequences of parametricity using Abadi & Plotkin's logic, as in [12]. For many of these proofs extensionality is needed, and we need very strong equality to deduce from theorems in Abadi & Plotkin's logic to category theoretic theorems, as we will see in Section 5. This is the reason why we propose parametric APL-structures as a category-theoretic definition of parametricity.

The internal language of an APL-structure is simply Abadi & Plotkin's logic on the internal language of the λ_2-fibration (see [5]), with the ordering relation in a fibre of $\mathbf{Prop}$ defined as $\phi \vdash \psi$ iff $[\![\phi]\!] \vdash [\![\psi]\!]$ holds in the model. Using the internal language we may express properties of the APL-structure, and ask whether these properties hold in the logic.

Definition 4.1. *The extensionality schemes in the internal language of an APL-structure are the schemes*

$$- \mid - \mid - \vdash \forall \alpha, \beta \colon \mathsf{Type}.\, \forall t, u \colon \alpha \to \beta.\, (\forall x \colon \alpha. tx =_{\beta} ux) \supset t =_{\alpha \to \beta} u, \tag{17}$$

$$\Xi \mid - \mid - \vdash \forall f, g \colon (\Pi \alpha \colon \mathsf{Type}.\, \sigma).\, (\forall \alpha \colon \mathsf{Type}. f\alpha =_{\sigma} g\alpha) \supset f =_{\Pi\alpha \colon \mathsf{Type}.\sigma} g, \tag{18}$$

where in (18) σ ranges over all types such that $\Xi, \alpha \vdash \sigma \colon \mathsf{Type}$.

Lemma 4.2. *For any APL-structure, very strong equality (Definition A.7) implies extensionality.*

Proof. We can formulate extensionality equivalently as the rules

$$\frac{\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash t =_{\tau} u}{\Xi \mid \Gamma \mid \Theta \vdash \lambda x \colon \sigma.\, t =_{\sigma \to \tau} \lambda x \colon \sigma.\, u}$$

$$\frac{\Xi, \alpha \colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash f =_{\sigma} g}{\Xi \mid \Gamma \mid \Theta \vdash \Lambda \alpha.\, \mathsf{Type}.\, f =_{\Pi\alpha \colon \mathsf{Type}.\sigma} \Lambda \alpha.\, \mathsf{Type}.\, g}$$

If internal equality is the same as external equality then these rules hold by the rules for external equality in Figure 1. □

Definition 4.3. *The schema*

$$\forall \vec{\alpha} \colon \mathsf{Type}.\, \forall u, v \colon \sigma.\, (u(\sigma[eq_{\vec{\alpha}}])v \supset\subset u =_{\sigma} v)$$

is called the Identity Extension Schema. *Here σ ranges over all types such that $\vec{\alpha} \vdash \sigma \colon \mathsf{Type}$.*

Definition 4.4. *A **parametric APL-structure** is an APL-structure with very strong equality – and hence extensionality – satisfying the Identity Extension Schema.*

Remark 4.5. *If we write out the interpretation of the Identity Extension Schema, we get a category-theoretical formulation of the notion of parametric APL-structure. It is an APL-structure with very strong equality, extensionality and in which for all types $\vec{\alpha} \vdash \sigma \colon \mathsf{Type}$,*

$$(id_{[\![\vec{\alpha} \vdash \sigma]\!]^2} \times [\![\vec{\alpha} \mid - \mid - \vdash \vec{eq}_{\alpha}]\!])^* J([\![\vec{\alpha} \vdash \sigma]\!]) = [\![\vec{\alpha} \mid x \colon \sigma, y \colon \sigma \mid - \vdash x =_{\sigma} y]\!].$$

Definition 4.6. *For any type $\beta, \vec{\alpha} \vdash \sigma(\beta, \vec{\alpha})$ we can form the parametricity schema:*

$$\forall \vec{\alpha} \colon \mathsf{Type}.\, \forall u \colon (\textstyle\prod \beta.\, \sigma).\, \forall \beta, \beta' \colon \mathsf{Type}.\, \forall R \colon \mathsf{Rel}(\beta, \beta').\, (u\, \beta)\sigma[R, eq_{\vec{\alpha}}](u\, \beta')$$

in the empty context.

Proposition 4.7. *The Identity Extension Schema implies the parametricity schema. Thus the parametricity schema holds in any parametric APL-structure.*

Proof. Since

$$\vec{\alpha} \mid u \colon \textstyle\prod \beta \colon \mathsf{Type}.\, \sigma(\beta, \vec{\alpha}) \mid - \vdash u =_{\prod \beta \colon \mathsf{Type}.\sigma} u$$

always holds in the model, by the Identity Extension Schema, we know that

$$\vec{\alpha} \mid u \colon \textstyle\prod \beta \colon \mathsf{Type}.\, \sigma(\beta, \vec{\alpha}) \mid - \vdash u(\prod \beta \colon \mathsf{Type}.\, \sigma)[eq_{\vec{\alpha}}]u$$

holds, but by the Axiom (15) this means that

$$\vec{\alpha} \mid u \colon \textstyle\prod \beta \colon \mathsf{Type}.\, \sigma(\beta, \vec{\alpha}) \vdash \forall \beta, \beta' \forall R \colon \mathsf{Rel}(\beta, \beta').\, (u\, \beta)(\sigma[R, eq_{\vec{\alpha}}])(u\, \beta')$$

holds as desired. □

Without assuming parametricity we can prove the logical relations lemma:

Lemma 4.8 (Logical Relations Lemma). *For any APL-structure the Logical Relations Schema*

$$- \mid - \mid - \vdash t\sigma t$$

holds, where t ranges over all closed *terms of closed type, i.e., $- \mid - \vdash t \colon \sigma$.*

Proof. The lemma is really just a restatement of the requirement that

$$J : \mathbf{Type} \to \mathbf{Relations}$$

is a functor. Let us write out the details.

A closed term t of closed type σ corresponds in the model to a map $t \colon 1 \to \sigma$ in $\mathbf{Type}_1$, and by definition of the interpretation

$$[\![- \mid x \colon \sigma, y \colon \sigma \mid - \vdash x\sigma y]\!] = J(\sigma).$$

The fact that J is required to be a functor, means exactly that the pair (t, t) should define a map in $\mathbf{Relations}$, i.e., the formula

$$- \mid - \mid - \vdash \forall x, y \colon 1.\, x1y \supset t\sigma t$$

should hold in the model. Since the relational interpretation of 1 is simply the constantly true relation, we get the statement of the lemma. □

Remark 4.9. *The Logical Relations Lemma suspiciously resembles the Identity Extension Schema. For a closed term of open type: $\vec{\alpha} \mid - \vdash t \colon \sigma$, the Logical Relations Lemma implies $(\Lambda\alpha.\, t) \prod \vec{\alpha}.\, \sigma(\Lambda\alpha.\, t)$, so that $t\sigma[eq_{\vec{\alpha}}]t$. However, since this only holds for* closed *terms t, we do not have the formula*

$$\forall t \colon \sigma.\, t\sigma[eq_{\vec{\alpha}}]t,$$

which is the formula that we will need to prove consequences of parametricity.

5 Consequences of parametricity

As mentioned in the introduction to Section 4 we may use Abadi & Plotkin's logic to derive consequences of parametricity in parametric APL-structures. In this section we exemplify how to do so. Through our examples, it should become apparent how extensionality and very strong equality play important roles in the proofs of the consequences.

The proofs of the consequences are based on theorems about Abadi & Plotkin's logic stated in [12]. For completeness, we have written out proofs of these theorems, often inspired by [3]. What is new here, is just that we show how to conclude from the logic to the APL-structures.

5.1 Dinaturality

We shall use the following definition very often.

Definition 5.1. *We say that $\vec{\alpha} \vdash \sigma\colon \mathsf{Type}$ is an inductively constructed type, if it can be constructed from free variables $\vec{\alpha}$ and closed types using the type constructors of λ_2, i.e., $\times, \to$ and $\prod \alpha.$.*

For example, if σ is a closed type then $\prod \alpha.\, \sigma \times \alpha$ is an inductively constructed type. However, some models may contain types that are not inductively constructed! For example, in syntactical models, any basic open type, such as the type $\alpha \vdash \mathit{lists}(\alpha)$ is not inductively constructed.

We define the notion of positive and negative occurrences of a type variable α in an inductively constructed type σ inductively over the structure of σ as follows. The type variable α occurs positively in α. The positive occurrences of α in $\sigma \times \tau$ are the positive occurrences of α in σ and the positive occurrences of α in τ. Likewise for negative occurrences. The positive occurrences of α in $\sigma \to \tau$ are the positive occurrences of α in τ and the negative occurrences of α in σ. The negative occurrences are the negative in τ and the positive in σ. The positive and negative occurrences of α in $\prod \beta.\, \sigma$ are the same as for σ, if $\alpha \neq \beta$. There are no positive or negative occurrences of α in $\prod \alpha.\, \sigma$ since we only consider free occurrences of a type variable.

Suppose $\sigma(\alpha, \beta)$ is an inductively constructed type with all free variables in α, β such that α occurs only negatively and β occurs only positively in σ. We may then for $f\colon \alpha \to \alpha'$ and $g\colon \beta \to \beta'$ define a morphism

$$\sigma(f, g) : \sigma(\alpha', \beta) \to \sigma(\alpha, \beta')$$

inductively over the structure of σ as in [12].

It is well-known that Dinaturality is a consequence of parametricity, but we include the proof for completeness.

Lemma 5.2 (Dinaturality). *In a parametric APL-structure, the dinaturality schema*

$$\forall \alpha, \beta.\, \forall f\colon \alpha \to \beta.\, \sigma(id_\alpha, f) \circ (\cdot)_\alpha =_{\prod \alpha.(\sigma(\alpha,\alpha)) \to \sigma(\alpha,\beta)} \sigma(f, id_\beta) \circ (\cdot)_\beta$$

holds. Here $(\cdot)_\alpha$ denotes the term $\lambda u\colon (\prod \alpha.\, \sigma(\alpha, \alpha)).\, u(\alpha)$.

Proof. Suppose $f\colon \alpha \to \beta$. By extensionality it suffices to prove that, for any $u\colon \prod \alpha.\, \sigma(\alpha, \alpha)$,

$$\sigma(id_\alpha, f) u(\alpha) =_{\sigma(\alpha,\beta)} \sigma(f, id_\beta) u(\beta).$$

Instantiating the Logical Relations Lemma with the types

$$\alpha, \beta, \gamma, \delta \vdash (\alpha \to \beta) \times (\gamma \to \delta)$$
$$\alpha, \beta, \gamma, \delta \vdash \sigma(\beta, \gamma) \to \sigma(\alpha, \delta)$$

and

$$t = \Lambda\alpha, \beta, \gamma, \delta.\, \lambda\omega\colon (\alpha \to \beta) \times (\gamma \to \delta).\, \sigma(\pi\omega, \pi'\omega)\colon$$
$$\textstyle\prod \alpha, \beta, \gamma, \delta.\, (\alpha \to \beta) \times (\gamma \to \delta) \to \sigma(\beta, \gamma) \to \sigma(\alpha, \delta)$$

we get

$$\alpha, \beta, \gamma, \delta, \alpha', \beta', \gamma', \delta' \mid x\colon (\alpha \to \beta) \times (\gamma \to \delta), y\colon (\alpha' \to \beta') \times (\gamma' \to \delta') \mid$$
$$R_1\colon \mathsf{Rel}(\alpha, \alpha'), R_2\colon \mathsf{Rel}(\beta, \beta'), R_3\colon \mathsf{Rel}(\gamma, \gamma'), R_4\colon \mathsf{Rel}(\delta, \delta') \mid$$
$$x(R_1 \to R_2) \times (R_3 \to R_4)y \vdash \sigma(\pi x, \pi' x)(\sigma[R_2, R_3] \to \sigma[R_1, R_4])\sigma(\pi y, \pi' y).$$

Recall the notation $\langle f \rangle$ for the graph of the function f defined as $(x\colon \alpha, y\colon \beta).\, f(x) =_\beta y$. If we set $\alpha, \beta, \gamma, \alpha'$ to α and set $\delta, \beta', \gamma', \delta'$ to β and let $R_1 = eq_\alpha$, $R_2 = R_3 = \langle f \rangle$ and $R_4 = eq_\beta$, then we get

$$x(eq_\alpha \to \langle f \rangle) \times (\langle f \rangle \to eq_\beta)y \vdash \sigma(\pi x, \pi' x)(\sigma[\langle f \rangle, \langle f \rangle] \to \sigma[eq_\alpha, eq_\beta])\sigma(\pi y, \pi' y).$$

If we set $x = \langle id_\alpha, f \rangle$ and $y = \langle f, id_\beta \rangle$ then since $id_\alpha(eq_\alpha \to \langle f \rangle)f$ and $f(\langle f \rangle \to eq_\beta)id_\beta$ we obtain

$$\sigma(id_\alpha, f)(\sigma[\langle f \rangle, \langle f \rangle] \to \sigma[eq_\alpha, eq_\beta])\sigma(f, id_\beta).$$

Since the parametricity schema tells us that

$$u(\alpha)\sigma[\langle f \rangle, \langle f \rangle]u(\beta),$$

it follows that

$$\sigma(id_\alpha, f)(u(\alpha))(\sigma[eq_\alpha, eq_\beta])\sigma(f, id_\beta)u(\beta),$$

but by the Identity Extension Schema this is just

$$\sigma(id_\alpha, f)(u(\alpha)) =_{\sigma(\alpha,\beta)} \sigma(f, id_\beta)u(\beta).$$

□

5.2 Products

Consider the type $T = \prod \alpha.\, \alpha \to \alpha$. The term $\Lambda\alpha.\, \lambda x\colon \alpha.\, x$ inhabits T. Thus

Proposition 5.3. *In any model of λ_2 the type T defines a fibred weak terminal object.*

Theorem 5.4. *In a parametric APL-structure, the proposition*

$$\forall u\colon T.\, (u =_T \Lambda\alpha.\, \lambda x\colon \alpha.\, x)$$

holds in the internal logic.

Proof. By extensionality it suffices to prove that

$$\alpha\colon \mathsf{Type} \mid u\colon T, x\colon \alpha \vdash (u\alpha)x =_\alpha x.$$

Consider the relation

$$\alpha\colon \mathsf{Type} \mid u\colon T, x\colon \alpha \vdash \rho = (y\colon \alpha, z\colon \alpha).\, y =_\alpha x : \mathsf{Rel}(\alpha, \alpha).$$

By parametricity we have

$$\alpha\colon \mathsf{Type} \mid u\colon T, x\colon \alpha \vdash (u\,\alpha)(\rho \to \rho)(u\,\alpha),$$

but this means that

$$\alpha\colon \mathsf{Type} \mid u\colon T, x\colon \alpha \vdash y =_\alpha x \supset (u\,\alpha)y =_\alpha x.$$

□

Theorem 5.5. *In a parametric APL-structure, T defines a fibred terminal object of* $\mathbf{Type} \to \mathbf{Kind}$.

Proof. Suppose $u\colon \sigma \to T$ is a morphism in the fibre. By the above theorem and extensionality, u is internally equal to $\lambda y\colon \sigma.\, \Lambda\alpha.\, \lambda x\colon \alpha.\, x$. By very strong equality we have external equality between u and $\lambda y\colon \sigma.\, \Lambda\alpha.\, \lambda x\colon \alpha.\, x$. So T is a terminal object. □

For two types σ and τ in the same fibre, consider

$$\sigma \hat{\times} \tau = \prod \alpha.\, ((\sigma \to \tau \to \alpha) \to \alpha).$$

We use $\hat{\times}$ to distinguish this definition from the usual fibrewise product denoted $\times$. We will show that $\hat{\times}$ defines a weak product in the fibre, and that in parametric APL-structures it defines a genuine product.

Let projections $\pi : \sigma \hat{\times} \tau \to \sigma$ and $\pi' : \sigma \hat{\times} \tau \to \tau$ be defined by

$$\pi x = x\,\sigma\,(\lambda x\colon \sigma.\, \lambda y\colon \tau.\, x)$$
$$\pi' x = x\,\tau\,(\lambda x\colon \sigma.\, \lambda y\colon \tau.\, y)$$

and let *pair* : $\sigma \to \tau \to \sigma \hat{\times} \tau$ be defined by

$$\mathit{pair}\; x\; y = \Lambda\alpha.\, \lambda f\colon \sigma \to \tau \to \alpha.\, f\; x\; y$$

If $f : \alpha \to \sigma$ and $g : \alpha \to \beta$, we will write $\langle f, g\rangle$ for $\lambda x\colon \alpha.\, \mathit{pair}\; (f\; x)\; (g\; x)$. Then

$$\pi \circ \langle f, g\rangle = \lambda x\colon \alpha.\, (\mathit{pair}\; (f\; x)\; (g\; x))\; \sigma\; (\lambda x\colon \sigma.\, \lambda y\colon \tau.\, x) = \lambda x\colon \alpha.\, f\; x = f$$

and likewise

$$\pi' \circ \langle f, g\rangle = g$$

This proves:

Proposition 5.6. *In any model of λ_2 the construction $\hat{\times}$ defines a fibrewise weak product.*

Theorem 5.7. *For any parametric APL-structure the proposition*

$$\forall \sigma, \tau.\, \langle \pi, \pi'\rangle =_{\sigma\hat{\times}\tau} id_{\sigma\hat{\times}\tau}$$

holds in the internal logic.

Proof. For any $f\colon \sigma \to \tau \to \alpha$ define $f^*\colon \sigma \hat{\times} \tau \to \alpha$ as

$$f^*\; x = x\; \alpha\; f.$$

Suppose $z\colon \sigma\hat{\times}\tau$. By parametricity, for any relation $R\colon \mathsf{Rel}(\alpha, \beta)$,

$$(z\; \alpha)((\mathit{eq}_\sigma \to \mathit{eq}_\tau \to R) \to R)(z\; \beta).$$

Now, for any $f\colon \sigma \to \tau \to \alpha$,

$$f^*(\mathit{pair}\ x\ y) = \mathit{pair}\ x\ y\ \alpha\ f = f\ x\ y,$$

i.e.,

$$\mathit{pair}(\mathit{eq}_\sigma \to \mathit{eq}_\tau \to \langle f^* \rangle) f,$$

which means that

$$(z\ \sigma \hat{\times} \tau\ \mathit{pair})\langle f^* \rangle (z\ \alpha\ f).$$

In other words,

$$f^*(z\ \sigma \hat{\times} \tau\ \mathit{pair}) =_\alpha z\ \alpha\ f.$$

Since the left hand side of this equation simply is

$$(z\ \sigma \hat{\times} \tau\ \mathit{pair})\ \alpha\ f,$$

we get by extensionality since α, f were arbitrary,

$$z\ \sigma \hat{\times} \tau\ \mathit{pair} =_{\sigma \hat{\times} \tau} z.$$

Suppose now that we are given $f\colon \sigma \to \tau \to \alpha$. We construct $g\colon \sigma \hat{\times} \tau \to \alpha$ by

$$g\ z = f\ (\pi\ z)\ (\pi'\ z)$$

Then $\mathit{pair}(\mathit{eq}_\sigma \to \mathit{eq}_\tau \to \langle g \rangle) f$ since

$$g\ (\mathit{pair}\ x\ y) = f\ (\pi \circ \mathit{pair}\ x\ y)(\pi' \circ \mathit{pair}\ x\ y) = f\ x\ y$$

Parametricity now states that for any $z\colon \sigma \hat{\times} \tau$

$$(z\ \sigma \hat{\times} \tau)((\mathit{eq}_\sigma \to \mathit{eq}_\tau \to \langle g \rangle) \to \langle g \rangle)(z\ \alpha).$$

Thus $(z\ \sigma \hat{\times} \tau\ \mathit{pair})\langle g \rangle (z\ \alpha\ f)$ and since $(z\ \sigma \hat{\times} \tau\ \mathit{pair}) =_{\sigma \hat{\times} \tau} z$ we have

$$f\ (\pi\ z)\ (\pi'\ z) = g\ z =_\alpha z\ \alpha\ f.$$

By extensionality

$$\lambda z\colon \sigma \hat{\times} \tau.\, \Lambda \alpha.\, \lambda f\colon \sigma \to \tau \to \alpha.\, f\ (\pi\ z)\ (\pi'\ z) =_{\sigma \hat{\times} \tau \to \sigma \hat{\times} \tau}$$
$$\lambda z\colon \sigma \hat{\times} \tau.\, \Lambda \alpha.\, \lambda f\colon \sigma \to \tau \to \alpha.\, z\ \alpha\ f = id_{\sigma \hat{\times} \tau}.$$

But the left hand side of this equation is just $\langle \pi, \pi' \rangle$. □

Theorem 5.8. *In any parametric APL-structure, $\hat{\times}$ defines a fibrewise product in* $\mathbf{Type} \to \mathbf{Kind}$.

Proof. Since clearly $\langle \pi \circ f, \pi' \circ f \rangle = \langle \pi, \pi' \rangle \circ f$ any map into $\sigma \hat{\times} \tau$ is uniquely determined by its composition with π and π' by Theorem 5.7 and very strong equality. □

5.3 Coproducts

For the empty sum we define

$$I = \prod \alpha.\, \alpha.$$

Proposition 5.9. *In any model of λ_2, I defines a fibred weak initial object.*

Proof. Suppose σ is a type over some $\mathbf{Kind}$ object Ξ. The interpretation of the term $x\colon I \vdash x\sigma$ is a morphism from I to σ in the fibre over Ξ. □

Theorem 5.10. *In a parametric APL-structure, the proposition*

$$\forall u\colon I.\, \bot$$

holds in the internal logic of the model.

Proof. Parametricity says

$$\forall u\colon \prod \alpha.\, \alpha.\, \forall \alpha, \beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{Rel}(\alpha, \beta).\, u(\alpha) R u(\beta)$$

Instantiate this with the definable relation

$$(x\colon 1, y\colon 1).\, \bot\colon \mathsf{Rel}(1, 1)$$

□

Theorem 5.11. *In a parametric APL-structure, I defines a fibred initial object of* $\mathbf{Type} \to \mathbf{Kind}$.

Proof. Given two morphisms $u, v\colon I \to \sigma$ we have

$$(\forall x\colon I.\, \bot) \vdash (\forall x\colon I.\, ux =_\sigma vx) \vdash (u =_{I \to \sigma} v),$$

so, by very strong equality, we have $u = v$. □

Given two types σ and τ we define

$$\sigma + \tau = \prod \alpha.\, (\sigma \to \alpha) \to (\tau \to \alpha) \to \alpha$$

and introduce combinators $\mathit{inl}_{\sigma,\tau}\colon \sigma \to \sigma + \tau$, $\mathit{inr}_{\sigma,\tau}\colon \tau \to \sigma + \tau$ and

$$\mathit{cases}_{\sigma,\tau}\colon \prod \alpha.\, ((\sigma \to \alpha) \to (\tau \to \alpha) \to (\sigma + \tau) \to \alpha)$$

by

$$\begin{aligned} \mathit{inl}_{\sigma+\tau}(a) &= \Lambda\alpha.\, \lambda f\colon \sigma \to \alpha.\, \lambda g\colon \tau \to \alpha.\, f(a), \\ \mathit{inr}_{\sigma+\tau}(a) &= \Lambda\alpha.\, \lambda f\colon \sigma \to \alpha.\, \lambda g\colon \tau \to \alpha.\, g(a), \\ \mathit{cases}_{\sigma+\tau}\, \alpha\, f\, g\, \omega &= \omega\, \alpha\, f\, g. \end{aligned}$$

Now, suppose we are given two morphisms $t : \sigma \to \alpha$ and $u : \tau \to \alpha$. Then we may define $[u, t] = \mathit{cases}_{\sigma,\tau}\, \alpha\, t\, u : \sigma + \tau \to \alpha$ and we then have

$$[u, t] \circ \mathit{inl}_{\sigma,\tau}(x) = \mathit{inl}_{\sigma,\tau}\, x\, \alpha\, t\, u = t(x)$$

and likewise

$$[u, t] \circ \mathit{inr}_{\sigma,\tau}(y) = \mathit{inr}_{\sigma,\tau}\, x\, \alpha\, t\, u = u(y)$$

so we have proved the following proposition.

Proposition 5.12. *For any model of λ_2, the operation $+$ defines a fibred weak coproduct.*

We will prove that in a parametric APL-structure, $\sigma + \tau$ is in fact a coproduct.

Theorem 5.13. *In a parametric APL-structure, the proposition*

$$\forall \alpha, \sigma, \tau \colon \mathsf{Type}.\, \forall h \colon \sigma + \tau \to \alpha.\, h =_{\sigma+\tau\to\alpha} [h \circ \mathit{inl}_{\sigma+\tau}, h \circ \mathit{inr}_{\sigma+\tau}]$$

holds.

Proof. We will first prove that

$$[\mathit{inl}_{\sigma+\tau}, \mathit{inr}_{\sigma+\tau}] =_{\sigma+\tau} id_{\sigma+\tau}.$$

Instantiating the parametricity schema for $\omega : \sigma + \tau$ with the relation $\langle f \rangle$ we get that, for any $f : \alpha \to \beta$ and all $a : \sigma \to \alpha$ and $\beta : \tau \to \alpha$,

$$f(\omega\ \alpha\ a\ b) =_\beta \omega\ \beta\ (f \circ a)\ (f \circ b).$$

Now consider any $a' : \sigma \to \alpha$ and $b' : \tau \to \alpha$ and set $f : \sigma + \tau \to \alpha$ to

$$f(u) = u\ \alpha\ a'\ b'.$$

If we set a above to *inl* and b to *inr* we get

$$(\omega\ (\sigma + \tau)\ \mathit{inl}\ \mathit{inr})\ \alpha\ a'\ b' =_\beta \omega\ \alpha\ (f \circ \mathit{inl})\ (f \circ \mathit{inr}). \tag{19}$$

Since

$$f \circ \mathit{inl}(x) = \mathit{inl}(x)\ \alpha\ a'\ b' = a'(x),$$

for all $x \colon \sigma$, and likewise $f \circ \mathit{inr}(y) = b'(y)$, for $y : \tau$, (19) reduces to

$$(\omega(\sigma + \tau)\ \mathit{inl}\ \mathit{inr})\ \alpha\ a'\ b' =_\beta \omega\ \alpha\ a'\ b'.$$

By extensionality this implies

$$(\omega(\sigma + \tau)\mathit{inl}\ \mathit{inr}) =_{\sigma+\tau} \omega,$$

and using extensionality again we obtain

$$[\mathit{inl}_{\sigma+\tau}, \mathit{inr}_{\sigma+\tau}] =_{\sigma+\tau\to\sigma+\tau} id_{\sigma+\tau}. \tag{20}$$

Finally, by the parametricity condition on *cases*, we have for any $h : \sigma + \tau \to \alpha$ that

$$h(\mathit{cases}(\sigma + \tau)\ \mathit{inl}\ \mathit{inr}\ \omega) =_\alpha \mathit{cases}\ \alpha\ (h \circ \mathit{inl})\ (h \circ \mathit{inr})\ \ \omega,$$

so by extensionality and (20),

$$h =_{\sigma+\tau\to\alpha} [h \circ \mathit{inl}, h \circ \mathit{inr}].$$

□

Theorem 5.14. *In any parametric APL-structure, $+$ defines a fibred coproduct of* $\mathbf{Type} \to \mathbf{Kind}$.

Proof. Using very strong equality, Theorem 5.13 tells us that maps out of $\sigma + \tau$ are uniquely determined by their compositions with *inl* and *inr*. □

5.4 Initial algebras

Definition 5.15. *Consider a fibred functor*

$$\begin{tikzcd} \mathbb{E} \arrow[rr, "T"] \arrow[dr] & & \mathbb{E} \arrow[dl] \\ & \mathbb{B}. & \end{tikzcd}$$

An indexed family of initial algebras for the functor T is a family

$$(\mathit{in}_\Xi \colon T(\sigma_\Xi) \to \sigma_\Xi)_{\Xi \in \mathrm{Obj}\,\mathbb{B}}$$

such that each in_Ξ is an initial algebra for the restriction of T to the fibre over Ξ and the family is closed under reindexing. If each in_Ξ is only a weak initial algebra we call it a family of weak initial algebras.

Suppose $\alpha \vdash \sigma\colon \mathsf{Type}$ is an inductively constructed type (see Definition 5.1) in which α occurs only positively. Then $\sigma(\alpha)$ can be considered a functor in each fibre [12]. Actually, in [12] Abadi & Plotkin construct a term

$$t : \textstyle\prod \alpha, \beta\colon \mathsf{Type}.\, (\alpha \to \beta) \to \sigma(\alpha) \to \sigma(\beta),$$

which internalizes the morphism part of the functor σ.

The type σ induces a fibred functor

$$\begin{tikzcd} \mathbf{Type} \arrow[rr] \arrow[dr] & & \mathbf{Type} \arrow[dl] \\ & \mathbf{Kind} & \end{tikzcd}$$

mapping $\Xi \vdash \tau$ to $\Xi \vdash \sigma(\tau)$. In this section we study families of initial algebras for such functors.

First we prove the graph lemma:

Lemma 5.16. *If $\alpha \vdash \sigma$ is an inductively constructed type in a parametric APL-structure in which α occurs only positively, interpreted as a fibred functor as in [12], then the formula*

$$\forall \alpha, \beta\colon \mathsf{Type}.\, \forall f\colon \alpha \to \beta.\, \sigma[\langle f \rangle] \equiv \langle \sigma(f) \rangle$$

holds in the internal language of the model, where, as usual, $\rho \equiv \rho'$ is short for

$$\forall x, y.\, \rho(x, y) \supset\subset \rho'(x, y).$$

Proof. Since the polymorphic strength t mentioned above is parametric, we have, for any pair of relations $\rho\colon \mathsf{Rel}(\alpha, \alpha')$ and $\rho'\colon \mathsf{Rel}(\beta, \beta')$,

$$t\; \alpha\; \beta((\rho \to \rho') \to (\sigma[\rho] \to \sigma[\rho']))t\; \alpha'\; \beta'. \tag{21}$$

If we instantiate this with $\rho = \mathit{eq}_\alpha$, $\rho' = \langle f \rangle$ for some map $f : \alpha \to \beta$, we get

$$t\; \alpha\; \alpha((\mathit{eq}_\alpha \to \langle f \rangle) \to (\mathit{eq}_{\sigma(\alpha)} \to \sigma[\langle f \rangle]))t\; \alpha\; \beta,$$

using the Identity Extension Schema. Since $id_\alpha(\mathit{eq}_\alpha \to \langle f \rangle)f$, and since $t\; \alpha\; \beta\; f = \sigma(f)$ and $t\; \alpha\; \alpha\; id_\alpha = \sigma(id_\alpha) = id_{\sigma(\alpha)}$ we get

$$id_{\sigma(\alpha)}(\mathit{eq}_{\sigma(\alpha)} \to \sigma[\langle f \rangle])\sigma(f),$$

that is,

$$\forall x\colon \sigma(\alpha).\, x(\sigma[\langle f\rangle])\sigma(f)x.$$

Thus we have proved $\langle\sigma(f)\rangle$ implies $\sigma[\langle f\rangle]$.

To prove the other direction, instantiate (21) with the relations $\rho = \langle f\rangle$ and $\rho' = \mathit{eq}_\beta$ for $f\colon \alpha \to \beta$. Since $f(\langle f\rangle \to \mathit{eq}_\beta)id_\beta$,

$$\sigma(f)(\sigma[\langle f\rangle] \to \mathit{eq}_{\sigma(\beta)})id_{\sigma(\beta)}.$$

So for any $x\colon \sigma(\alpha)$ and $y\colon \sigma(\beta)$ we have $x(\sigma[\langle f\rangle])y$ implies $\sigma(f)x = y$. In other words, $\sigma[\langle f\rangle]$ implies $\langle\sigma(f)\rangle$. □

We shall now define a family of initial algebras for the functor induced by σ. In each fibre $\mathbf{Type}_\Xi$ we may define the type

$$\mu\alpha.\,\sigma(\alpha) = \prod \alpha.\,((\sigma(\alpha) \to \alpha) \to \alpha)$$

with combinators

$$\mathit{fold}\colon \prod \alpha.\,((\sigma(\alpha) \to \alpha) \to \mu\beta.\,\sigma(\beta) \to \alpha)$$

and

$$\mathit{in} : \sigma(\mu\alpha.\,\sigma(\alpha)) \to \mu\alpha.\,\sigma(\alpha)$$

given by

$$\mathit{fold}\;\alpha\; f\; z = z\;\alpha\; f$$

and

$$\mathit{in}\; z = \Lambda\alpha.\,\lambda f\colon \sigma(\alpha) \to \alpha.\, f(\sigma(\mathit{fold}\;\alpha\; f)z).$$

Theorem 5.17. *In any model of second-order λ-calculus the family*

$$(\Xi \vdash \mathit{in}\colon \sigma(\mu\alpha.\,\sigma(\alpha)) \to \mu\alpha.\,\sigma(\alpha))_\Xi$$

is a family of weak initial algebras for σ.

Proof. Given any algebra $f\colon \sigma(\alpha) \to \alpha$ in any fibre, the diagram

$$\begin{tikzcd}
\sigma(\mu\alpha.\,\sigma(\alpha)) \arrow[r, "\mathit{in}"] \arrow[d, "\sigma(\mathit{fold}\;\alpha\; f)"'] & \mu\alpha.\,\sigma(\alpha) \arrow[d, "\mathit{fold}\;\alpha\; f"] \\
\sigma(\alpha) \arrow[r, "f"] & \alpha
\end{tikzcd}$$

is commutative since

$$(\mathit{fold}\;\alpha\; f) \circ \mathit{in}\; z = \mathit{in}\; z\; \alpha f = f(\sigma(\mathit{fold}\;\alpha\; f)\; z)$$

and

$$f \circ \sigma(\mathit{fold}\;\alpha\; f)\; z = f(\sigma(\mathit{fold}\;\alpha\; f)\; z).$$

□

We will show that in a parametric APL-structure, $(\Xi \vdash \mathit{in})_\Xi$ actually is a family of initial algebras. First we prove a lemma.

Lemma 5.18. *In a parametric APL-structure, the formula*

$$\textit{fold}\ \mu\alpha.\,\sigma(\alpha)\ \textit{in} =_{\mu\alpha.\sigma(\alpha)\to\mu\alpha.\sigma(\alpha)} id_{\mu\alpha.\sigma(\alpha)}$$

holds in the internal logic.

Proof. Consider an arbitrary element $\omega\colon \mu\alpha.\,\sigma(\alpha)$ and a map $f\colon \alpha \to \beta$. The parametricity condition then gives

$$(\omega\ \alpha)((\sigma[\langle f\rangle] \to \langle f\rangle) \to \langle f\rangle)(\omega\ \beta).$$

Since Lemma 5.16 tells us that $\sigma[\langle f\rangle] \equiv \langle\sigma(f)\rangle$, this means that, if $a\colon \sigma(\alpha) \to \alpha$ and $b\colon \sigma(\beta) \to \beta$ have the property that

$$\forall x\colon \sigma(\alpha).\, f(a\,x) =_\beta b(\sigma(f)\,x)$$

(that is, if f is a morphism of algebras), then

$$f(\omega\ \alpha\ a) =_\beta \omega\ \beta\ b.$$

Consider now an arbitrary algebra $k\colon \sigma(\alpha) \to \alpha$ and instantiate the above with the algebra morphism $\textit{fold}\alpha k$ from *in* to k, to get

$$\textit{fold}\ \alpha\ k(\omega\ \mu\alpha.\,\sigma(\alpha)\ \textit{in}) =_\alpha \omega\ \alpha\ k.$$

Since the left hand side of this equation is $(\omega\ \mu\alpha.\,\sigma(\alpha)\ \textit{in})\ \alpha\ k$, we get by extensionality that

$$\omega\ \mu\alpha.\,\sigma(\alpha)\ \textit{in} =_{\mu\alpha.\sigma(\alpha)} \omega$$

and therefore, using extensionality again,

$$\textit{fold}\ \mu\alpha.\,\sigma(\alpha)\ \textit{in} =_{\mu\alpha.\sigma(\alpha)\to\mu\alpha.\sigma(\alpha)} id_{\mu\alpha.\sigma(\alpha)},$$

as required. □

Theorem 5.19. *Suppose $g\colon \mu\alpha.\,\sigma(\alpha) \to \alpha$ induces a map between algebras from in to $f\colon \sigma(\alpha) \to \alpha$ in a parametric APL-structure. Then*

$$g =_{\mu\alpha.\sigma(\alpha)\to\alpha} \textit{fold}\ \alpha\ f$$

holds in the internal logic.

Proof. Since g is a map of algebras, the parametricity condition on an arbitrary $\omega\colon \mu\alpha.\,\sigma(\alpha)$ entails as in the proof of Lemma 5.18 that

$$g(\omega\ \mu\alpha.\,\sigma(\alpha)\ \textit{in}) =_\alpha \omega\ \alpha\ f$$

and therefore the result follows from extensionality since, by Lemma 5.18,

$$\omega\ \mu\alpha.\,\sigma(\alpha)\ \textit{in} = (\textit{fold}\ \mu\alpha.\,\sigma(\alpha)\ \textit{in})\ \omega =_{\mu\alpha.\sigma(\alpha)} \omega$$

and, moreover,

$$\omega\ \alpha\ f = (\textit{fold}\ \alpha\ f)\ \omega.$$

□

Theorem 5.20. *In a parametric APL-structure, $(\Xi \vdash \textit{in})_\Xi$ is a family of initial algebras for σ.*

Proof. Using very strong equality Thm 5.19 gives uniqueness of algebra morphisms out of *in*. □

Remark 5.21. *Consider the case of an inductively constructed type $\alpha, \beta \vdash \sigma(\alpha, \beta)$ in which α and β occur only positively. For each closed type τ we may consider the type $\alpha \vdash \sigma(\alpha, \tau)$ and the analysis above gives us a family of initial algebras for this functor. Moreover, for each morphism $f : \tau \to \tau'$ between closed types we get a morphism of algebras induced by initiality:*

$$\begin{tikzcd}
\sigma(\mu\alpha.\, \sigma(\alpha, \tau), \tau) \arrow[r, dashed] \arrow[dd, "in_\tau"] & \sigma(\mu\alpha.\, \sigma(\alpha, \tau'), \tau) \arrow[d, "\sigma(id,f)"] \\
 & \sigma(\mu\alpha.\, \sigma(\alpha, \tau'), \tau') \arrow[d, "in_{\tau'}"] \\
\mu\alpha.\, \sigma(\alpha, \tau) \arrow[r, dashed] & \mu\alpha.\, \sigma(\alpha, \tau').
\end{tikzcd}$$

For example, if we consider the type $\alpha, \beta \vdash 1 + \alpha \times \beta$, then for any τ, we get lists$(\tau) = \mu\alpha.\, (1 + \alpha \times \tau)$ and, for any $f : \tau \to \tau'$, the induced morphism is the familiar morphism map f : lists$(\tau) \to$ lists(τ'), which applies f to each element in a list.

5.5 Final coalgebras

In this section we consider the same setup as in Section 5.4, that is, $\alpha \vdash \sigma\colon$ Type is an inductively constructed type in which α occurs only positively. As before σ defines a fibred endofunctor on $\mathbf{Type} \to \mathbf{Kind}$.

Definition 5.22. *Consider a fibred functor*

$$\begin{tikzcd}
\mathbb{E} \arrow[rr, "T"] \arrow[dr] & & \mathbb{E} \arrow[dl] \\
 & \mathbb{B}. &
\end{tikzcd}$$

An indexed family of final coalgebras for the functor T is a family

$$(out_\Xi\colon \sigma_\Xi \to T(\sigma_\Xi))_{\Xi \in \mathrm{Obj}\,\mathbb{B}}$$

such that each out$_\Xi$ is a final coalgebra for the restriction of T to the fibre over Ξ and the family is closed under reindexing. If each out$_\Xi$ is only a weak final coalgebra we call it a family of weak final coalgebras.

In this section we define a family of weak final coalgebras for σ and prove that for parametric APL-structures it is in fact a family of final coalgebras. First we need to define existential quantification in each fibre as

$$\coprod \alpha.\, \sigma(\alpha) = \prod \alpha.\, (\prod \beta.\, (\sigma(\beta) \to \alpha)) \to \alpha$$

and the combinator *pack* : $\prod \alpha.\, (\sigma(\alpha) \to \coprod \beta.\, \sigma(\beta))$ by

$$pack\; \alpha\; x = \Lambda\beta\lambda f\colon \prod \alpha.\, (\sigma(\alpha) \to \beta).\, f\; \alpha\; x.$$

In each fibre we define the type

$$\nu\alpha.\, \sigma(\alpha) = \coprod \alpha.\, ((\alpha \to \sigma(\alpha)) \times \alpha) = \prod \alpha.\, (\prod \beta.\, (\beta \to \sigma(\beta)) \times \beta \to \alpha) \to \alpha$$

with combinators

$$\mathit{unfold}\colon \prod \alpha.\, ((\alpha \to \sigma(\alpha)) \to \alpha \to (\nu\alpha.\, \sigma(\alpha)))$$

and

$$\mathit{out}\colon \nu\alpha.\, \sigma(\alpha) \to \sigma(\nu\alpha.\, \sigma(\alpha))$$

defined as

$$\mathit{unfold}\, \alpha\ f\ x = \mathit{pack}\, \alpha\ \langle f, x\rangle$$

and

$$\mathit{out}(x) = x\ \sigma(\nu\alpha.\, \sigma(\alpha))\ (\Lambda\alpha\lambda\langle f, x\rangle\colon ((\alpha \to \sigma(\alpha)) \times \alpha).\, \sigma(\mathit{unfold}\, \alpha\ f)(f\ x)).$$

Theorem 5.23. *In any model of second-order λ-calculus $(\Xi \vdash \mathit{out})_\Xi$ is a family of weak final coalgebras for σ.*

Proof. Consider a coalgebra $f\colon \alpha \to \sigma(\alpha)$ in any fibre. Then

$$\begin{tikzcd}
\alpha \arrow[r, "f"] \arrow[d, "\mathit{unfold}\ \alpha\ f"'] & \sigma(\alpha) \arrow[d, "\sigma(\mathit{unfold}\ \alpha\ f)"] \\
\nu\alpha.\, \sigma(\alpha) \arrow[r, "\mathit{out}"] & \sigma(\nu\alpha.\, \sigma(\alpha))
\end{tikzcd}$$

commutes since

$$\mathit{out}(\mathit{unfold}\, \alpha\ f\ z) = \mathit{out}(\mathit{pack}\, \alpha\ \langle f, z\rangle) =$$
$$(\mathit{pack}\, \alpha\ \langle f, z\rangle)\ (\sigma(\nu\alpha.\, \sigma(\alpha)))\ (\Lambda\alpha\lambda\langle f, x\rangle\colon ((\alpha \to \sigma(\alpha)) \times \alpha).\, \sigma(\mathit{unfold}\, \alpha\ f)(f\ x)) =$$
$$\sigma(\mathit{unfold}\, \alpha\ f)(f\ z)$$

□

Lemma 5.24. *In a parametric APL-structure,*

$$\mathit{unfold}\, \nu\alpha.\, \sigma(\alpha)\ \mathit{out}$$

is internally equal to the identity on $\nu\alpha.\, \sigma(\alpha)$.

Proof. Set $h = \mathit{unfold}\, \nu\alpha.\, \sigma(\alpha)\ \mathit{out}$ in the following.
By parametricity, for any $k : \alpha \to \beta$,

$$\mathit{unfold}\, \alpha (\langle k\rangle \to \sigma[\langle k\rangle]) \to (\langle k\rangle \to \mathit{eq}_{\nu\alpha.\sigma(\alpha)})\mathit{unfold}\, \beta.$$

Hence, since $\sigma[\langle k\rangle] \equiv \langle \sigma(k)\rangle$ by Lemma 5.16, if

$$k : (f : \alpha \to \sigma(\alpha)) \to (g : \beta \to \sigma(\beta))$$

is a morphism of coalgebras, then

$$\mathit{unfold}\, \alpha\ f =_{\alpha \to \nu\alpha.\sigma(\alpha)} (\mathit{unfold}\, \beta\ g) \circ k.$$

So since h is a morphism of coalgebras from *out* to *out* we have $h = h^2$. Intuitively, all we need to prove now is that h is “surjective”.

Consider any $g : \prod \alpha. ((\alpha \to \sigma(\alpha)) \times \alpha \to \beta)$. By parametricity and Lemma 5.16, for any coalgebra map $k : (f : \alpha \to \sigma(\alpha)) \to (f' : \alpha' \to \sigma(\alpha'))$, we must have

$$\forall x\colon \alpha.\, g\, \alpha\, \langle f, x\rangle =_\beta g\, \alpha'\, \langle f', k(x)\rangle.$$

Using this on the coalgebra map *unfold* $\alpha\ f$ from f to *out* we obtain

$$\forall x\colon \alpha.\, g\, \alpha\langle f, x\rangle =_\beta g\, \nu\alpha.\, \sigma(\alpha)\langle \mathit{out}, \mathit{unfold}\ \alpha\ f\ x\rangle.$$

In other words, if we define

$$k\colon \prod \alpha. ((\alpha \to \sigma(\alpha)) \times \alpha \to \tau),$$

where $\tau = (\nu\alpha.\, \sigma(\alpha) \to \sigma(\nu\alpha.\, \sigma(\alpha))) \times \nu\alpha.\, \sigma(\alpha)$, to be

$$k = \Lambda\alpha.\, \lambda\langle f, x\rangle : (\alpha \to \sigma(\alpha)) \times \alpha.\, \langle \mathit{out}, \mathit{unfold}\ \alpha\ f\ x\rangle,$$

then

$$\forall\alpha.\, g\ \alpha =_{(\alpha\to\sigma(\alpha))\times\alpha\to\beta} (g\ \nu\alpha.\, \sigma(\alpha)) \circ (k\ \alpha). \tag{22}$$

Now, suppose we are given $\alpha, \alpha', R\colon \mathsf{Rel}(\alpha, \alpha')$ and terms f, f' such that

$$f((R \to \sigma[R]) \times R \to \beta)f'.$$

Then, by (22) and parametricity of g

$$g\ \alpha\ f =_\beta g\ \alpha'\ f' =_\beta (g\ \nu\alpha.\, \sigma(\alpha))(k\ \alpha'\ f'),$$

from which we conclude

$$g(\forall(\alpha, \beta, R\colon \mathsf{Rel}(\alpha, \beta)).\, ((R \to \sigma[R]) \times R \to \langle g\ \nu\alpha.\, \sigma(\alpha)\rangle))k.$$

This implies that for any $x\colon \nu\alpha.\, \sigma(\alpha)$ by parametricity we have

$$x\ \beta\ g =_\beta g\ \nu\alpha.\, \sigma(\alpha)\ (x\ \tau\ k).$$

Thus, since g was arbitrary, we may apply the above to $g = k$ and get

$$x\ \tau\ k =_\tau k\ \nu\alpha.\, \sigma(\alpha)\ (x\ \tau\ k) = \langle \mathit{out}, \mathit{unfold}\ \nu\alpha.\, \sigma(\alpha)\ \pi(x\ \tau\ k)\ \pi'(x\ \tau\ k)\rangle.$$

If we write

$$l = \lambda x\colon \nu\alpha.\, \sigma(\alpha).\, \mathit{unfold}\ \nu\alpha.\, \sigma(\alpha)\ \pi(x\ \tau\ k)\ \pi'(x\ \tau\ k),$$

then since k is a closed term, so is l, and from the above calculations we conclude that we have

$$\forall\beta.\, \forall g : \prod \alpha. (\alpha \to \sigma(\alpha)) \times \alpha \to \beta.\, x\ \beta\ g =_\beta g\ \nu\alpha.\, \sigma(\alpha)\ \langle \mathit{out}, l\ x\rangle.$$

Now, finally

$$\begin{gathered} h(l\ x) = \mathit{unfold}\ \nu\alpha.\, \sigma(\alpha)\ \mathit{out}\ (l\ x) = \\ \mathit{pack}\ \nu\alpha.\, \sigma(\alpha)\ \langle \mathit{out}, l\ x\rangle = \\ \Lambda\beta.\, \lambda g : \prod \alpha. ((\alpha \to \sigma(\alpha)) \times \alpha \to \beta).\, g\ \nu\alpha.\, \sigma(\alpha)\ \langle \mathit{out}, l\ x\rangle =_{\nu\alpha.\sigma(\alpha)} \\ \Lambda\beta.\, \lambda g : \prod \alpha. ((\alpha \to \sigma(\alpha)) \times \alpha \to \beta).\, x\ \beta\ g = x \end{gathered}$$

where we have used extensionality. Thus l is a right inverse to h, and we conclude

$$h\ x =_{\nu\alpha.\sigma(\alpha)} h^2(l\ x) =_{\nu\alpha.\sigma(\alpha)} h(l\ x) =_{\nu\alpha.\sigma(\alpha)} x.$$

□

Theorem 5.25. *In a parametric APL-structure,* $(\Xi \vdash out)_\Xi$ *is a family of final coalgebras for* σ*.*

Proof. Consider a map of coalgebras into *out*:

$$\begin{tikzcd}
\alpha \arrow[r, "f"] \arrow[d, "g"] & \sigma(\alpha) \arrow[d, "\sigma(g)"] \\
\nu\alpha.\,\sigma(\alpha) \arrow[r, "out"] & \sigma(\nu\alpha.\,\sigma(\alpha)).
\end{tikzcd}$$

By parametricity of *unfold* we have

$$\textit{unfold}\ \alpha\ f =_{\alpha\to\nu\alpha.\sigma(\alpha)} (\textit{unfold}\ \nu\alpha.\,\sigma(\alpha)\ \textit{out}) \circ g =_{\alpha\to\nu\alpha.\sigma(\alpha)} g.$$

Very strong equality then implies uniqueness of coalgebra morphisms into *out* as desired. □

5.6 Generalizing to strong fibred functors

In this section, our aim is to generalize the results of Sections 5.4 and 5.5 to initial algebras and final coalgebras for a more general class of fibred functors, than the ones defined by inductively constructed types. These functor are called strong fibred functors.

Definition 5.26. *An endofunctor* $T : B \to B$ *on a cartesian closed category is called* ***strong*** *if there exists a natural transformation* $t_{\sigma,\tau} : \tau^\sigma \to T\tau^{T\sigma}$ *preserving identity and composition:*

$$\begin{tikzcd}
1 \arrow[r, "\widehat{id_\sigma}"] \arrow[dr, "\widehat{id_{T\sigma}}"'] & \sigma^\sigma \arrow[d, "t_{\sigma,\sigma}"] \\
 & T\sigma^{T\sigma}
\end{tikzcd}
\qquad
\begin{tikzcd}
\sigma_2^{\sigma_1} \times \sigma_3^{\sigma_2} \arrow[r, "comp"] \arrow[d, "t\times t"] & \sigma_3^{\sigma_1} \arrow[d, "t"] \\
T\sigma_2^{T\sigma_1} \times T\sigma_3^{T\sigma_2} \arrow[r, "comp"] & T\sigma_3^{T\sigma_1}.
\end{tikzcd}$$

The natural transformation t *is called the* ***strength*** *of the functor* T*.*

One should note that t in the definition above represents the morphism part of the functor T in the sense that it makes the diagram

$$\begin{tikzcd}
1 \arrow[r, "\widehat{f}"] \arrow[dr, "\widehat{Tf}"'] & \tau^\sigma \arrow[d, "t_{\sigma,\tau}"] \\
 & T\tau^{T\sigma}
\end{tikzcd}$$

commute, for any morphism $f \colon \sigma \to \tau$. This follows from the commutative diagram

$$\begin{tikzcd}
1 \arrow[dr, "\widehat{id}"'] \arrow[drr, bend left, "\widehat{id}"] \arrow[ddr, bend right, "\widehat{f}"'] & & \\
 & \sigma^\sigma \arrow[r, "t"] \arrow[d, "f^\sigma"] & T\sigma^{T\sigma} \arrow[d, "Tf^{T\sigma}"] \\
 & \tau^\sigma \arrow[r, "t"] & T\tau^{T\sigma}.
\end{tikzcd}$$

Definition 5.27. *A* ***strong fibred functor*** *is a fibred endofunctor*

$$\begin{tikzcd}\mathbb{E} \arrow[rr, "T"] \arrow[dr] & & \mathbb{E} \arrow[dl] \\ & \mathbb{B} & \end{tikzcd}$$

on a fibred ccc, for which there exists a fibred natural transformation t from the fibred functor $(-)^{(+)}$ to $T(-)^{T(+)}$ satisfying commutativity of the two diagrams of Definition 5.26 in each fibre. The natural transformation t is called the ***strength*** *of the functor T.*

In this definition, one should of course check that the two functors $(-)^{(+)}$ and $T(-)^{T(+)}$ — a priori only defined on the fibres — in fact define fibred functors

$$\begin{tikzcd}\mathbb{E}^{\mathrm{op}} \times_{\mathbb{B}} \mathbb{E} \arrow[rr] \arrow[dr] & & \mathbb{E} \arrow[dl] \\ & \mathbb{B}. & \end{tikzcd}$$

But this is easily seen. Notice also that T is not required to preserve the fibred ccc-structure and that the components of t are preserved under reindexing since t is a fibred natural transformation.

Example 5.28. *An inductively constructed type with one free variable $\alpha \vdash \sigma\colon \mathsf{Type}$, where α occurs only positively, defines a strong fibred functor: see Section 5.4.*

But in many situations one may want to reason about other strong fibred functors. For example, if the λ_2-fibration of the APL-structure models other type constructions than the ones from λ_2 for which there are natural functorial interpretations, one may want to prove existence of initial algebras for functors induced by types in this extended language.

All fibred endofunctors on λ_2-fibrations are in a sense given by types.

Lemma 5.29. *For any strong fibred functor*

$$\begin{tikzcd}\mathbf{Type} \arrow[rr, "F"] \arrow[dr] & & \mathbf{Type} \arrow[dl] \\ & \mathbf{Kind} & \end{tikzcd}$$

on a λ_2-fibration there exists, in the internal language of $\mathbf{Type} \to \mathbf{Kind}$ a type $\alpha \vdash \sigma$ and a term

$$- \vdash s\colon \textstyle\prod \alpha, \beta.\, (\alpha \to \beta) \to \sigma(\alpha) \to \sigma(\beta)$$

inducing F.

Proof. Denote by $T \in \mathbf{Type}_{\Omega}$ the generic object of the λ_2-fibration and for any type $\tau \in \mathbf{Type}_{\Xi}$ denote by $\hat{\tau}\colon \Xi \to \Omega$ the map satisfying $\tau = \hat{\tau}^*(T)$. Set $\sigma = F(T)$. Then for any type $\tau\colon \mathbf{Type}_{\Xi}$,

$$F(\tau) = F(\hat{\tau}^* T) = \hat{\tau}^* \sigma$$

which is the interpretation of $\sigma(\tau)$ in the internal language.

Now suppose the fibred natural transformation t is a strength for F. Consider the component $(t_{\Omega^2})_{[\![\alpha,\beta\vdash\alpha]\!],[\![\alpha,\beta\vdash\beta]\!]}$. This is a map in $\mathbf{Type}_{\Omega^2}$ from $[\![\alpha, \beta \vdash \alpha \to \beta]\!]$ to $[\![\alpha, \beta \vdash \sigma(\alpha) \to \sigma(\beta)]\!]$, i.e. a term $\alpha, \beta \vdash t' \colon (\alpha \to \beta) \to (\sigma(\alpha) \to \sigma(\beta))$ in the internal language. Set $s = \Lambda\alpha. \Lambda\beta. t'$.

To check that σ, s induce the functor F we only need to check that for any pair of types $\tau, \tau' \in \mathbf{Type}_\Xi$, $\Xi \vdash s\ \tau\ \tau'$ is interpreted as $(t_\Xi)_{\tau,\tau'}$. But $[\![\Xi \vdash s\ \tau\ \tau']\!] = \langle\tau, \tau'\rangle^*(t') = (t_\Xi)_{\tau,\tau'}$, since t is preserved by reindexing. □

Lemma 5.29 tells us that we can reason about strong fibred functors in the internal language. For instance, denoting the strong fibred functor by σ we may write

$$\alpha, \beta \mid f : \alpha \to \beta \vdash \sigma(f) \colon \sigma(\alpha) \to \sigma(\beta)$$

for $s\ \alpha\ \beta\ f$ where s is the polymorphic term inducing σ's action on morphisms.

Furthermore, since the morphism part of the functor is represented by a *polymorphic* term, we can use parametricity to reason about it. For instance, we may prove the following generalization of Lemma 5.16.

Lemma 5.30 (Graph Lemma). *For any parametric APL-structure, if σ is a strong fibred endofunctor $\mathbf{Type} \to \mathbf{Type}$, then the formula*

$$\forall\alpha, \beta \colon \mathsf{Type}. \forall f \colon \alpha \to \beta. \sigma[\langle f\rangle] \equiv \langle\sigma(f)\rangle$$

holds in the internal language of the APL-structure, where $\rho \equiv \rho'$ is short for

$$\forall x, y. \rho(x, y) \supset\subset \rho'(x, y).$$

The proof of this lemma is the same as the proof of Lemma 5.16.

Corollary 5.31. *For any parametric APL-structure, the morphism part of a strong fibred endofunctor σ is uniquely determined by the object part.*

Proof. By Lemma 5.30, $y = \sigma(f)(x)$ iff $x\sigma[\langle f\rangle]y$. □

Theorem 5.32. *In a parametric APL-structure, any strong fibred functor $F : \mathbf{Type} \to \mathbf{Type}$ has*

- *A family of initial algebras defined as in Section 5.4*
- *A family of final coalgebras defined as in Section 5.5*

Proof. The proofs work exactly as in Sections 5.4 and 5.5 since we may express the functor F in the internal language, as described above.

The fact that these initial algebras and final coalgebras are preserved by reindexing follows from the fact that the strengths t are preserved. □

6 Concrete APL-structures

In this section we define a concrete parametric APL-structure based on a well-known variant of the per-model (see, for instance, [5, Section 8.4]).

The diagram of Definition 3.3 in the concrete model is:

$$\begin{tikzcd}
 & \mathbf{UFam}(\mathbf{RegSub}(\mathbf{Asm})) \arrow[d, "r"] \\
\mathbf{PFam}(\mathbf{Per}) \arrow[r, hook, "I"] \arrow[dr, "p"] & \mathbf{UFam}(\mathbf{Asm}) \arrow[d, "q"] \\
 & \mathbf{PPer}
\end{tikzcd} \tag{23}$$

The fibration p is the fibration of [5, Def. 8.4.9]; we repeat the definition here. In the following, $\mathbf{Per}$ and $\mathbf{Asm}$, will denote the sets of partial equivalence relations and assemblies respectively on the natural numbers (see [5]).

The category $\mathbf{PPer}$ is defined as

Objects Natural numbers.

Morphisms A morphism $f : n \to 1$ is a pair (f^p, f^r) where $f^p : \mathbf{Per}^n \to \mathbf{Per}$ is any map and

$$f^r \in \prod_{\vec{R},\vec{S}\in\mathbf{Per}^n} \left[\prod_{i\leq n} P(\mathbb{N}/R_i \times \mathbb{N}/S_i) \to P(\mathbb{N}/f^p(\vec{R}) \times \mathbb{N}/f^p(\vec{S})\right]$$

is a map that satisfies *the identity extension condition*: $f^r(\overrightarrow{Eq}) = Eq$. A morphism from n to m is an m-vector of morphism from n to 1.

We can now define $\mathbf{PFam}(\mathbf{Per})$ as the indexed category with fibre over n defined as

Objects morphisms, $n \to 1$ of $\mathbf{PPer}$.

Morphisms a morphism from f to g is an indexed family of maps $(\alpha_{\vec{R}})_{\vec{R}\in\mathbf{Per}^n}$ where

$$\alpha_{\vec{R}} : \mathbb{N}/f^p(\vec{R}) \to \mathbb{N}/g^p(\vec{R})$$

are tracked uniformly, i.e., there exists a code e such that, for all $\vec{R}$ and $[n] \in \mathbb{N}/f^p(\vec{R})$, $\alpha_{\vec{R}}([n]) = [e \cdot n]$. Further, the morphism α should respect relations, that is, if $A_i \subset \mathbb{N}/R_i \times \mathbb{N}/S_i$ and $(a, b) \in f^r(\vec{A})$ then $(\alpha_{\vec{R}}(a), \alpha_{\vec{S}}(b)) \in g^r(\vec{A})$.

Reindexing is by composition.

Next we define the fibration q. The fibre category $\mathbf{UFam}(\mathbf{Asm})_n$ is defined as

Objects all maps $f : \mathbf{Per}^n \to \mathbf{Asm}$.

Morphisms a morphism from f to g is an indexed family of maps $(\alpha_{\vec{R}})_{\vec{R}\in\mathbf{Per}^n}$ where

$$\alpha_{\vec{R}} : f(\vec{R}) \to g(\vec{R})$$

are maps between the underlying sets of the assemblies that are tracked uniformly, i.e. there exists a code e such that for all $\vec{R}$ and all $i \in f(\vec{R})$ and all $a \in E_{f(\vec{R})}(i)$ we have $e \cdot a \in E_{g(\vec{R})}(\alpha_{\vec{R}}(i))$.

Reindexing is again by composition.

Finally we can define the category $\mathbf{UFam}(\mathbf{RegSub}(\mathbf{Asm}))$ as

Objects An object over f is any family of subsets $(A_{\vec{R}} \subseteq f(\vec{R}))_{\vec{R}}$, where by subset we mean subset of the underlying set of the assembly.

Morphisms In each fibre the morphisms are just subset inclusions.

Reindexing is defined as follows: Suppose $\phi : f \to g$ is a morphism in $\mathbf{UFam}(\mathbf{Asm})$ projecting to $q\phi : n \to m$ in $\mathbf{PPer}$. By definition this is a map in the fibre of $\mathbf{UFam}(\mathbf{Asm})$ over n from f to $(q\phi)^*(g)$. Such morphisms are given by indexed families of maps

$$\phi_{\vec{R}} : f(\vec{R}) \to g \circ (q\phi)^p(\vec{R})$$

ranging over $\vec{R} \in \mathbf{Per}^n$ so we can define

$$\phi^*(A_{\vec{S}} \subset g(\vec{S}))_{\vec{S} \in \mathbf{Per}^m} = (\phi_{\vec{R}}^{-1}(A_{g \circ (q\phi)^p(\vec{R})}))_{\vec{R} \in \mathbf{Per}^n}$$

The inclusion I is obtained by projecting (f^p, f^r) to f^p using the inclusion of $\mathbf{Per}$ into $\mathbf{Asm}$.

Lemma 6.1. *p is a $\lambda 2$-fibration.*

Proof. This is [5, Prop. 8.4.10]. The ccc-structure is given by a pointwise construction, and 1 is clearly a generic object. For a type $f : n + 1 \to 1$ we define $\prod f : n \to 1$ as

$$\begin{array}{rcl} (\prod f)^p(\vec{R}) & = & \{(a, a') \mid \forall U, V \in \mathbf{Per}. \forall B \subseteq \mathbb{N}/U \times \mathbb{N}/V. \, a \in |f^p(\vec{R}, U)| \text{ and} \\ & & a' \in |f^p(\vec{R}, V)| \text{ and } ([a], [a']) \in f^r_{(\vec{R},U),(\vec{R},V)}(\vec{E}q_{\vec{R}}, B)\} \end{array}$$

and

$$\begin{array}{rcl} (\prod f)^r_{\vec{R} \times \vec{S}}(\vec{A}) & = & \{([a]_{\prod(f)^p(\vec{R})}, [a']_{\prod(f)^p(\vec{S})}) \mid \forall U, V \in \mathbf{Per}. \forall B \subseteq \mathbb{N}/U \times \mathbb{N}/V \\ & & ([a]_{f^p(\vec{R},U)}, [a']_{f^p(\vec{S},V)}) \in f^r_{(\vec{R},U),(\vec{S},V)}(\vec{A}, B)\} \end{array}$$

for $\vec{A} \subseteq \vec{R} \times \vec{S}$. □

Theorem 6.2. *The diagram (23) defines a parametric APL-structure.*

We do not prove Lemma 6.2 directly. Instead, we will show in Remark 8.27 that (23) is a special case of the parametric completion process of Section 8.

Remark 6.3. *In the above model we use nothing special about the PCA $\mathbb{N}$ so the same construction applies to pers and assemblies over any PCA. All the lemmas above generalize, so that in the general case we also obtain a parametric APL-structure.*

6.1 A parametric non-well-pointed APL-structure

We may generalize the construction above even further to the case of relative realizability. Suppose we are given a PCA A and a sub-PCA $A_\sharp$. We can then define the APL-structure as above with pers and assemblies over A, with the only exception that morphisms in $\mathbf{PFam}(\mathbf{Per})$ and $\mathbf{UFam}(\mathbf{Asm})$ should be uniformly tracked by codes in $A_\sharp$. All the proofs of section 6 generalize so that we obtain:

Proposition 6.4. *For any PCA A and sub-PCA $A_\sharp$ the diagram*

$$\begin{tikzcd}
& \mathbf{UFam}(\mathbf{RegSub}(\mathbf{Asm}(\mathbf{A},\mathbf{A}_\sharp))) \arrow[d, "r"] \\
\mathbf{PFam}(\mathbf{Per}(\mathbf{A},\mathbf{A}_\sharp)) \arrow[r, hook, "I"] \arrow[dr, "p"] & \mathbf{UFam}(\mathbf{Asm}(\mathbf{A},\mathbf{A}_\sharp)) \arrow[d, "q"] \\
& \mathbf{PPer}(\mathbf{A},\mathbf{A}_\sharp)
\end{tikzcd}$$

defines a parametric APL-structure.

However, one may also prove:

Proposition 6.5. *The fibre $\mathbf{PFam}(\mathbf{Per}(\mathbf{A},\mathbf{A}_\sharp))_0$ is in general not well-pointed.*

Proof. Consider a per of the form $\{(a,a)\}$, for $a \in A \setminus A_\sharp$. There may be several maps out of this per, but it does not have any global points. □

Proposition 6.4 tells us that all the theorems of Section 5 apply, such that the λ_2-fibration

$$\mathbf{PFam}(\mathbf{Per}(\mathbf{A},\mathbf{A}_\sharp)) \to \mathbf{PPer}(\mathbf{A},\mathbf{A}_\sharp)$$

has all the properties that we consider consequences of parametricity. This should be compared to [1] in which a family of parametric models is presented (with another definition of "parametric model") and the consequences of parametricity are proved only for the *well-pointed* parametric models.

7 Comparing with Ma & Reynolds notion of parametricity

In this section we compare the notion of parametricity presented above with Ma & Reynolds' notion of parametricity [6] (see also [5]). This latter notion was the first proposal for a general category theoretic formulation of parametricity and is perhaps the most well-known.

To define parametricity in the sense Ma & Reynolds, consider first a situation where we are given a λ_2-fibration $\begin{tikzcd} E \arrow[r] & B \end{tikzcd}$ and a logic on the types given by an indexed first-order logic fibration

$$\begin{tikzcd} D \arrow[r] & E \arrow[r] & B \end{tikzcd} .$$

Consider the category of relations on closed types $LR(E_1)$ defined as

$$\begin{tikzcd}
LR(E_1) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & D_1 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & D \arrow[d] \\
E_1 \times E_1 \arrow[r, "\times"] & E_1 \arrow[r, hook] & E
\end{tikzcd}$$

where by 1 we mean the terminal object of B. In this case we have a reflexive graph of categories

$$\begin{tikzcd}
E_1 \arrow[r, shift left=0] & LR(E_1) \arrow[l, shift right=2] \arrow[l, shift left=2]
\end{tikzcd} ,$$

where the functor going left to right maps a type to the identity on that type. By reflexive graph we mean that the two compositions starting and ending in E_1 are identities.

Definition 7.1. *The λ_2-fibration*

$$\begin{tikzcd} E \arrow[d] \\ B \end{tikzcd}$$

is parametric in the sense of Ma & Reynolds with respect to $D \to E$ *if there exists a λ_2-fibration $F \to C$ and a reflexive graph of λ_2 fibrations*

$$\begin{tikzcd} \begin{pmatrix} E \\ \big\downarrow \\ B \end{pmatrix} \arrow[r, leftarrow, shift left=2] \arrow[r] \arrow[r, leftarrow, shift right=2] & \begin{pmatrix} F \\ \big\downarrow \\ C \end{pmatrix} \end{tikzcd}$$

such that the restriction to the fibres over the terminal objects becomes

$$\begin{tikzcd} E_1 \arrow[r, leftarrow, shift left=1.5] \arrow[r] \arrow[r, leftarrow, shift right=1.5] & LR(E_1) \end{tikzcd} .$$

Given an APL-structure, we have a logic over types given by the pullback of $\mathbf{Prop}$ along I. We also have a reflexive graph giving the relational interpretation of all types. It is natural to ask what kind of parametricity we obtain by requiring that the reflexive graph giving the relational interpretation of types satisfies the requirements of Definition 7.1.

First we notice that $\mathbf{Relations}_1 = LR(E_1)$, and that the two maps going from $\mathbf{Relations}$ to E_1 are in fact the domain and codomain maps, as required, so the requirements of Definition 7.1 only effect the nature of the map J.

The last requirement of Definition 7.1 says exactly that, for all closed types σ,

$$J(\llbracket\sigma\rrbracket) = \llbracket eq_\sigma\rrbracket.$$

Consider now an open type $\vec{\alpha} \vdash \sigma\colon \mathsf{Type}$ and a vector of closed types $\vec{\tau}$. Then, since J is a map of fibrations, we have

$$J(\llbracket\sigma(\vec{\tau})\rrbracket) = J(\llbracket\vec{\tau}\rrbracket^*\llbracket\vec{\alpha} \vdash \sigma\rrbracket) = J(\llbracket\vec{\alpha} \vdash \sigma\rrbracket) \circ \llbracket eq_{\vec{\tau}}\rrbracket = \llbracket\sigma[eq_{\vec{\tau}}]\rrbracket.$$

In other words, the model satisfies a weak form of Identity Extension Schema:

Definition 7.2. *The schema*

$$\forall u, v\colon \sigma(\vec{\tau}).\, (u\sigma[eq_{\vec{\tau}}]v) \supset\subset u =_{\sigma(\vec{\tau})} v$$

where $\vec{\alpha} \vdash \sigma$ ranges over all types and $\vec{\tau}$ ranges over all closed types is called the weak identity extension schema.

We will briefly mention which of the consequences of parametricity mentioned in Section 5 that hold under assumption of the weak Identity Extension Schema.

First we notice that the weak Identity Extension Schema implies the parametricity schema

$$\forall u\colon (\textstyle\prod \beta\colon \mathsf{Type}.\, \sigma(\beta, \tau_2, \ldots, \tau_n)).\, u(\forall\beta.\, \sigma[\beta, eq_{\tau_2}, \ldots, eq_{\tau_n}])u$$

in the case where the τ_i are closed types.

Using only this weak version of the parametricity schema, we can still prove existence of terminal and initial types, since in these cases we only need to use parametricity on the closed types T and I.

The proofs of existence of products and coproducts, however, fail when σ and τ are open types, since we need to use the parametricity condition on the open types $\sigma \hat{\times} \tau$ and $\sigma + \tau$.

The case of initial algebras goes through, since the proof only uses parametricity of $\mu\alpha.\,\sigma(\alpha)$, which is a closed type. The proof of Lemma 5.24, however, uses parametricity of the type $\prod \alpha.\,((\alpha \to \sigma(\alpha))\times\alpha \to \beta)$ where β is a type variable, so this proof does not go through with only the weak parametricity schema. In other words, in the setting of reflexive graphs as in Definition 7.1, we do not have a proof of existence of final coalgebras.

See also [15] for a related discussion.

8 A parametric completion process

In this section we give a description of a parametric completion process that given a model of λ_2 internal to some category satisfying certain requirements produces a parametric APL-structure. The construction is related to the parametric completion process of [15] in the sense that the process that constructs the λ_2-fibration contained in the APL-structure generated by our completion process is basically the parametric completion process of [15] (only the setup varies slightly). This means that if the ambient category is a topos, then the parametric completion process of [15] produces models parametric in our new sense which then satisfies the consequences of parametricity of Section 5. This fact is no surprise, but, to our knowledge, it has not been proved in the literature.

The concrete model of Section 6 is a result of the parametric completion process described in this section. Before describing the completion process we recall the theory of internal models of λ_2.

8.1 Internal models for λ_2

Suppose we are given a locally cartesian closed category $\mathbb{E}$. Given a full internal category $\mathbf{D}$ of $\mathbb{E}$ we may consider the externalization $\mathbf{D}$

$$\begin{tikzcd}\mathrm{Fam}(\mathbf{D}) \arrow[d] \\ \mathbb{E}\end{tikzcd} .$$

We shall denote by $\mathbf{D}_0$ the object of objects, and by $\mathbf{D}_1$ the object of morphisms of $\mathbf{D}$. The fibre over $\Xi \in \mathbb{E}$ is the internal functor category from Ξ considered as a discrete category to $\mathbf{D}$, i.e., objects are morphisms $\Xi \to \mathbf{D}_0$ and morphism are morphisms of $\mathbb{E}$: $\Xi \to \mathbf{D}_1$.

Proposition 8.1. *Suppose $\mathbf{D}$ is a full internally cartesian closed category that has right Kan extensions for internal functors $F : \Xi \to \mathbf{D}$ along projections $\Xi \times \mathbf{D}_0 \to \Xi$. Then the externalization of $\mathbf{D}$ is a λ_2-fibration.*

Proof. Since $\mathbf{D}$ is internally cartesian closed, its externalization has cartesian closed fibres preserved under reindexing [5, Corollary 7.3.9]. Clearly $\mathbf{D}_0$ is a generic object for the fibration.

Polymorphism is modeled using the Kan extensions, since for any type $\sigma : \Xi \times \mathbf{D}_0 \to \mathbf{D}$ the right Kan extension of σ along $\pi : \Xi \times \mathbf{D}_0 \to \Xi$ is the functor $\prod \alpha.\, \sigma$ in the diagram

$$\begin{tikzcd}
\Xi \times \mathbf{D}_0 \arrow[r, "\sigma"] \arrow[d, "\pi"] & \mathbf{D} \\
\Xi. \arrow[ur, dashed, "\prod \alpha.\sigma"'] &
\end{tikzcd}$$

The universality condition for the right Kan extension then gives the bijective correspondence

$$\mathrm{Nat}(\tau \circ \pi, \sigma) \cong \mathrm{Nat}(\tau, \textstyle\prod \alpha.\, \sigma)$$

between the sets of natural transformations. Since $\pi^*\tau = \tau \circ \pi$, for $\tau : \Xi \to \mathbf{D}$, this states exactly that the right Kan extension provides the right adjoint to π^*, as required.

To show that the Beck-Chevalley condition is satisfied, we need to show that for $u\colon \Xi' \to \Xi$ we have

$$u^*(\textstyle\prod \alpha.\, \sigma) \cong \prod \alpha.\, ((u \times id)^*\sigma),$$

that is,

$$(\textstyle\prod \alpha.\, \sigma) \circ u \cong \prod \alpha.\, (\sigma \circ (u \times id)).$$

By Lemma 8.2 below, we may write out the values of these two functors on objects $A \in \Xi'$ as limits:

$$((\textstyle\prod \alpha.\, \sigma) \circ u)(A) \quad = \quad \varprojlim_{u(A) \to \pi(A')} \sigma(A') \qquad (24)$$

$$(\textstyle\prod \alpha.\, (\sigma \circ u \times id))(A) \quad = \quad \varprojlim_{A \to \pi A''} \sigma(u \times id(A'')). \qquad (25)$$

In (24) we take the limit over all maps $f\colon u(A) \to \pi(A')$ in the discrete category Ξ. But since this is a discrete category, such maps only exist in the case $\pi(A') = u(A)$, so (24) can be rewritten as

$$\textstyle\prod_{D' \in \mathbf{D}_0} \sigma(u(A), D').$$

Likewise (25) can be rewritten as

$$\textstyle\prod_{D'' \in \mathbf{D}_0} \sigma(u(A), D''),$$

proving that the Beck-Chevalley condition is satisfied. □

Lemma 8.2. *Suppose the Kan extension* $\mathrm{RK}_H(F)$ *in the diagram*

$$\begin{tikzcd}
\mathbb{L} \arrow[r, "H"] \arrow[d, "F"'] & \mathbb{H} \arrow[dl, "\mathrm{RK}_H(F)"] \\
\mathbb{F} &
\end{tikzcd}$$

exists. If $\mathbb{L}$, $\mathbb{H}$ *are discrete, then* $\mathrm{RK}_H(F)$ *is given as a pointwise limit construction (as in [7, Theorem 1, p.237]).*

Proposition 8.1 justifies the following definition.

Definition 8.3. *An internal category* $\mathbf{D}$ *of a locally cartesian closed category* $\mathbb{E}$ *is called an* internal model *of* λ_2 *if it satisfies the assumptions of Proposition 8.1.*

8.2 Input for the parametric completion process

The parametric completion process takes the following ingredients as input:

1. A quasitopos $\mathbb{E}$
2. An internal model $\mathbf{D}$ of λ_2 in $\mathbb{E}$.

We will further assume that the inclusion

$$\begin{tikzcd}
\mathrm{Fam}(\mathbf{D}) \arrow[rr] \arrow[dr] & & \mathbb{E}^{\rightarrow} \arrow[dl, "\mathrm{cod}"] \\
& \mathbb{E} &
\end{tikzcd}$$

which we have already assumed is full and faithful, preserves products and is closed under regular subobjects. The latter means that for each object $E \in \mathbb{E}$, the fibre category $\mathrm{Fam}(\mathbf{D})_E$ is closed under regular subobjects as a subcategory of $\mathbb{E}/E$.

The logic $\mathrm{RegSub}_{\mathbb{E}} \to \mathbb{E}$ of regular subobjects induces a logic on $\mathbb{E}^{\rightarrow}$ by

$$\begin{tikzcd}
\mathbb{Q} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{RegSub}_{\mathbb{E}} \arrow[d] \\
\mathbb{E}^{\rightarrow} \arrow[r, "\mathrm{dom}"] & \mathbb{E},
\end{tikzcd}$$

which, by Lemma A.8, makes the composable fibration

$$\begin{tikzcd}
\mathbb{Q} \arrow[r] & \mathbb{E}^{\rightarrow} \arrow[r, "\mathrm{cod}"] & \mathbb{E}
\end{tikzcd} ,$$

an indexed first-order logic fibration with an indexed family of generic objects, simple products and simple coproducts.

Let Σ be the regular subobject classifier of $\mathbb{E}$. We can now form an internal fibration[2] by using the Grothendieck construction on the functor $(d \in \mathbf{D}) \mapsto \Sigma^d$, with Σ^d ordered pointwise. We think of this fibration as the internalization of $\mathrm{RegSub}_{\mathbb{E}} \to \mathbb{E}$ restricted to $\mathbf{D}$ and write it as $a\colon \mathbf{Q} \to \mathbf{D}$. Notice that since $\mathbf{D}$ is closed under regular subobjects, $\mathbf{Q} \to \mathbf{D}$ is a subfibration of the subobject fibration on $\mathbf{D}$, and since its externalization is simply the restriction of $\mathbb{Q} \to \mathbb{E}^{\rightarrow}$, it is closed under the logical operations $\top, \wedge, \supset, \forall, =$ from the regular subobject fibration.

Associated to the model given by $\mathbf{D}$ there is a canonical pre-APL- structure

$$\begin{tikzcd}
& \mathbb{Q} \arrow[d] \\
\mathrm{Fam}(\mathbf{D}) \arrow[r] \arrow[dr] & \mathbb{E}^{\rightarrow} \arrow[d] \\
& \mathbb{E}
\end{tikzcd} \tag{26}$$

To this we can associate, as usual, the fibration of relations denoted by $\mathbf{Relations_D} \to \mathbf{RelCtx_D}$.

[2] By internal fibration, we mean an internal functor, whose externalization is a fibration. By an internal fibration having structure such as $\wedge, \supset, \forall, =$ we mean that the externalization has the same (indexed) structure

8.3 The completion process

We define the category $\mathbf{LR}(\mathbf{D})$ to have as objects logical relations of $\mathbf{D}$ in the logic of $\mathbf{Q}$ and as morphisms pairs of morphisms in $\mathbf{D}$ that preserve relations.

Lemma 8.4. *The category $\mathbf{LR}(\mathbf{D})$ is an internal cartesian closed category of $\mathbb{E}$.*

Proof. We set

$$\mathbf{LR}(\mathbf{D})_0 = \{(X, Y, \phi) \in \mathbf{D}_0 \times \mathbf{D}_0 \times \mathbf{Q}_0 \mid a(\phi) = X \times Y\}$$

and

$$\begin{array}{rcl} \mathbf{LR}(\mathbf{D})_1 & = & \coprod_{(X,Y,\phi),(X',Y',\phi')\in\mathbf{LR}(\mathbf{D})_0} \{(f, g) \in \mathbf{D}_1 \times \mathbf{D}_1 \mid \\ & & f : X \to X' \wedge g : Y \to Y' \wedge \phi \le (f \times g)^* \phi'\}. \end{array}$$

For the cartesian closed structure we define:

$$(X, Y, \phi) \times (X', Y', \phi') = (X \times X', Y \times Y', \phi \times \phi'),$$

where $\phi \times \phi'((x, x'), (y, y')) = \phi(x, y) \wedge \phi'(x', y')$, and

$$(X, Y, \phi) \to (X', Y', \phi') = (X \to X', Y \to Y', \phi \to \phi'),$$

where

$$\phi \to \phi'(f, g) = \forall x \in X \forall y \in Y (\phi(x, y) \supset \phi'(f(x), g(y))).$$

□

Let

$$\begin{tikzcd} G = \cdot \arrow[r, shift right=2] & \cdot \arrow[l, shift right=2] \arrow[l, shift left=2] \end{tikzcd}$$

be the generic reflexive graph category, and consider the functor category $\mathbb{E}^G$. Since it is well known that $\mathbf{Cat}(\mathbb{E}^G) \cong \mathbf{Cat}(\mathbb{E})^G$ and $\mathbf{CCCat}(\mathbb{E}^G) \cong \mathbf{CCCat}(\mathbb{E})^G$ it follows that

Lemma 8.5. $\mathbf{D} \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{LR}(\mathbf{D})$ *is an internal cartesian closed category of $\mathbb{E}^G$.*

We now aim to prove that $\mathbf{D} \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{LR}(\mathbf{D})$ is an internal model of λ_2. By the lemma, all that remains is to prove that there are right Kan extensions for internal functors from $\Xi \times \mathbf{D}_0 \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \Xi' \times \mathbf{LR}(\mathbf{D}_0)$ to $\mathbf{D} \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{LR}(\mathbf{D})$ along projections to $\Xi \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \Xi'$. This is the same a saying that the fibration

$$\mathrm{Fam}(\ \mathbf{D}^n \underset{\longleftarrow}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{LR}(\mathbf{D})^n\ ) \to \mathbb{E}^G$$

has right adjoints to reindexing functors along projections.

We first consider the simpler case with spans in stead of reflexive graphs. Let $\mathbf{R}(\mathbf{D})$ denote the internal category

$$\begin{tikzcd} & \mathbf{LR}(\mathbf{D}) \arrow[dl, "\partial_0"'] \arrow[dr, "\partial_1"] & \\ \mathbf{D} & & \mathbf{D} \end{tikzcd}$$

inside $\mathbb{E}^\Lambda$, where Λ is the obvious category.

An object of $\mathrm{Fam}(\mathbf{R}(\mathbf{D}))$ is a triple of maps (f, g, ρ) such that

$$\begin{tikzcd}
 & \Theta \arrow[rrr, "\rho"] \arrow[dl] \arrow[dr] & & & \mathbf{LR}(\mathbf{D})_0 \arrow[dl] \arrow[dr] & \\
\Xi_0 \arrow[rrr, "f"] & & \Xi_1 \arrow[rrr, "g"] & \mathbf{D}_0 & & \mathbf{D}_0
\end{tikzcd} \tag{27}$$

commutes. Since $\mathbf{LR}(\mathbf{D})_0$ is the object of all relations on objects of $\mathbf{D}$, the idea is that we can consider such a triple as a definable relation

$$[\![\Xi_0, \Xi_1 \mid \Theta \vdash \rho\colon \mathsf{Rel}(f(\Xi_0), g(\Xi_1))]\!],$$

i.e., an object of $\mathbf{Relations_D}$. We will make this intuition precise in Lemma 8.6.

A vertical morphism in the category $\mathrm{Fam}(\mathbf{R}(\mathbf{D}))$ from (f, g, ρ) to (f', g', ρ') is by definition a triple consisting of a morphism from f to f', a morphism from g to g' and a morphism from ρ to ρ'. But since morphisms in $\mathbf{LR}(\mathbf{D})$ are pairs of morphisms preserving relations, and since the triple of morphisms is required to make the obvious diagram commute, we can consider such a morphism as a pair $(s\colon f \to f', t\colon g \to g')$ such that

$$\forall A \in \Theta. \forall x\colon f(\partial_0(A)), y\colon g(\partial_1(A)). \, \rho(x, y) \supset \rho'(s_{\partial_0(A)}(x), t_{\partial_1(A)}(y)),$$

as interpreted in the internal language of the quasi-topos, where $\supset$ refers to the internal ordering in $\mathbf{Q}$.

Lemma 8.6. *There is an isomorphism of fibrations*

$$\left(\begin{array}{c} \mathrm{Fam}(\mathbf{R}(\mathbf{D})) \\ \downarrow \\ \mathbb{E}^\Lambda \end{array}\right) \xrightarrow{\cong} \left(\begin{array}{c} \mathbf{Relations_D} \\ \downarrow \\ \mathbf{RelCtx_D} \end{array}\right)$$

Proof. Unwinding the definition of $\mathbf{RelCtx_D}$, we find that the objects are triples (Ξ_0, Ξ_1, Ξ) together with maps $\Xi \to \Xi_0 \times \Xi_1$ in $\mathbb{E}$. A map from $\Xi \to \Xi_0 \times \Xi_1$ to $\Xi' \to \Xi_0' \times \Xi_1'$ is a triple

$$\rho\colon \Xi \to \Xi', \quad f\colon \Xi_0 \to \Xi_0', \quad g\colon \Xi_1 \to \Xi_1'$$

making the obvious diagram commute. Thus $\mathbf{RelCtx_D} \cong \mathbb{E}^\Lambda$.

Objects in $\mathbf{Relations_D}$ are given as morphism in $\mathbf{RelCtx_D}$ into the interpretation of $\alpha, \beta \mid R\colon \mathsf{Rel}(\alpha, \beta)$ in (26). But the interpretation of this is easily seen to be

$$\textstyle\coprod_{\alpha,\beta \in \mathbf{D}_0} \Sigma^{\alpha \times \beta} \to \mathbf{D}_0 \times \mathbf{D}_0,$$

and since $\mathbf{LR}(\mathbf{D})_0 = \coprod_{\alpha,\beta \in \mathbf{D}_0} \Sigma^{\alpha \times \beta}$ we get a bijective correspondence between objects of $\mathbf{Relations_D}$ and objects of $\mathrm{Fam}(\mathbf{R}(\mathbf{D}))$. For morphisms, a vertical morphism in $\mathrm{Fam}(\mathbf{R}(\mathbf{D}))$ from (f, g, ρ) to (f', g', ρ') is by the above discussion a pair of morphisms $t\colon f \to f', s\colon g \to g'$ satisfying $\rho \supset (t \times s)^* \rho'$, which is exactly the same as a vertical morphism in $\mathbf{Relations_D}$. □

Lemma 8.7. *All internal functors* $\begin{array}{c}\Xi\\ \swarrow\ \searrow\\ \Xi_0 \quad \Xi_1\end{array} \times \mathbf{R}(\mathbf{D})_0 \to \mathbf{R}(\mathbf{D})$ *have right Kan extensions along the projection to* $\begin{array}{c}\Xi\\ \swarrow\ \searrow\\ \Xi_0 \quad \Xi_1\end{array}$

Proof. The statement to be proved is equivalent to the statement that the fibration on the left hand side of the isomorphism of Lemma 8.6 has simple products. Since we know that the fibration on the right of the isomorphism has simple products, we are done. □

Let us now consider the case that we are really interested in. We shall assume that we are given a functor (f^t, f^r) in $\mathbb{E}^G$:

$$\begin{tikzcd}
\Xi' \times \mathbf{LR}(\mathbf{D})_0 \arrow[r, "\pi"] \arrow[d, shift right=3, "\partial_0"'] \arrow[d, shift left=3, "\partial_1"] \arrow[ddr, "f^r"] & \Xi' \arrow[d, shift right=3, "\partial_0"'] \arrow[d, shift left=3, "\partial_1"] \\
\Xi \times \mathbf{D}_0 \arrow[r, "\pi"] \arrow[u, "I" description] \arrow[ddr, "f^t"'] & \Xi \arrow[u, "I" description] \\
& \mathbf{LR}(\mathbf{D}) \arrow[d, shift right=3, "\partial_0"'] \arrow[d, shift left=3, "\partial_1"] \\
& \mathbf{D}, \arrow[u, "I" description]
\end{tikzcd} \tag{28}$$

and we would like to find a right Kan extension of (f^t, f^r) along (π, π) (notice that we have used the notation $\partial_0, \partial_1, I$ for the structure maps of all objects of $\mathbb{E}^G$ - this should not cause any confusion, since it will be clear from the context which map is referred to). Let us call this extension $(\prod_{par} f^t, \prod_{par} f^r)$. An obvious idea is to try the pair $(\prod f^t, \prod f^r)$ provided by Lemma 8.7. However, $\prod_{par} f^r$ should commute with I, and we cannot know that $\prod f^r$ will do that. Consider $\prod f^r(I(A))$ for some $A \in \Xi$:

$$\begin{tikzcd}
\bar{\prod} f^r(I(A)) \arrow[d, tail] \\
\prod f^t(A) \times \prod f^t(A).
\end{tikzcd}$$

If we pull this relation back along the diagonal on $\prod f^t(A)$ we get a subobject

$$|\prod f^r(I(A))| \rightarrowtail \prod f^t(A)$$

(called the *field* of $\prod f^r(I(A))$). Logically, $|\prod f^r(I(A))|$ is the set $\{x \in \prod f^t(A) \mid (x, x) \in \bar{\prod} f^r(I(A))\}$, so if we restrict $\prod f^r(I(A))$ to this subobject, we get a relation relation containing the identity relation. The other inclusion will be easy to prove. Thus the idea is to let $\prod_{par} f^t$ be the map that maps A to $|\prod f^r(I(A))|$, and let $\prod_{par} f^r(R)$ be the relation obtained by restricting $\prod f^r(R)$ to $\prod_{par} f^t(\partial_0(R)) \times \prod_{par} f^t(\partial_1(R))$.

Theorem 8.8. *For (f^t, f^r), (π, π) as in (28), the right Kan extension of (f^t, f^r) along (π, π) exists.*

Proof. We will define $\prod_{par} f^t(A)$ as the pullback

$$\begin{tikzcd}
(\prod_{par} f^t)(A) \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "\lrcorner", very near start] & (\prod f^r)(I(A)) \arrow[d, tail] \\
\prod f^t(A) \arrow[r, "\Delta"] & \prod f^t(A) \times \prod f^t(A)
\end{tikzcd}$$

where Δ is the diagonal map. We define $\prod_{par} f^r(R)$ for $R \in \Xi'$, to be the pullback

$$\begin{tikzcd}
(\prod_{par} f^r)(R) \arrow[rr] \arrow[d, tail] \arrow[dr, phantom, "\lrcorner", very near start] & & (\prod f^r)(R) \arrow[d, tail] \\
\prod_{par} f^t(\partial_0 R) \times \prod_{par} f^t(\partial_1 R) \arrow[rr, tail] & & \prod f^t(\partial_0 R) \times \prod f^t(\partial_1 R).
\end{tikzcd}$$

First we will show that $\prod_{par} f^r(I(A)) = I(\prod_{par} f^t(A))$ for all A. Logically

$$\prod_{par} f^r(I(A)) = \{(x,y) \in \prod f^r(I(A)) \mid (y,y),(x,x) \in \prod f^r(I(A))\} \supseteq \\ \{(x,x) \mid x \in |\prod f^r(I(A))|\} = I(\prod_{par} f^t(A))$$

To prove the other inclusion suppose $(x,y) \in \prod_{par} f^r(I(A)) \subseteq \prod f^r(I(A))$. Then for any $\sigma_{n+1} \in D_0$,

$$(x,y) \in \pi^*(\prod f^r)(I(A), I(\sigma_{n+1})).$$

Let $\epsilon_{A,\sigma_{n+1}}$ denote the appropriate component of the counit for $\pi^* \dashv \prod$. Then

$$(\epsilon_{A,\sigma_{n+1}} x, \epsilon_{A,\sigma_{n+1}} y) \in \pi^*(\prod f^r)(I(A), I(\sigma_{n+1})) = I(f^t(A, \sigma_{n+1})),$$

so $\epsilon_{A,\sigma_{n+1}} x = \epsilon_{A,\sigma_{n+1}} y$. Since $\prod f^t(A)$ is the product of $f^t(A, \sigma_{n+1})$ over σ_{n+1} in $\mathbf{D}_0$, and $\epsilon_{A,\sigma_{n+1}}$ is simply the projection onto the σ_{n+1}-component, $\epsilon_{A,\sigma_{n+1}} x = \epsilon_{A,\sigma_{n+1}} y$ for all σ_{n+1} implies $x = y$ as desired.

Finally we will show that $\prod_{par}$ provides the desired right adjoint. Recall that a morphism from (g^t, g^r) to (h^t, h^r), where

$$\begin{tikzcd}
\Xi' \arrow[rr, "g^r"] \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] & & \mathbf{LR}(\mathbf{D})_0 \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] \\
\Xi \arrow[u, "I" description] \arrow[rr, "g^t"] & & \mathbf{D}_0 \arrow[u, "I" description]
\end{tikzcd}$$

and likewise (h^t, h^r) is a morphism $s\colon g^t \to h^t$ preserving relations. In the internal language this means that for each $A \in \Xi$ we have a map $s_A\colon g^t(A) \to h^t(A)$ such that for R with $\partial_0(R) = A, \partial_1(R) = B$, $(x,y) \in g^r(R)$ implies $(s_A(x), s_B(y)) \in h^r(R)$.

Now, from Lemma 8.7 we easily derive a one-to-one correspondence between maps $(g^t, g^r) \to (\prod f^t, \prod f^r)$ and maps $(g^t \circ \pi, g^r \circ \pi) \to (f^t, f^r)$. Since $\prod_{par} f^t(A) \subseteq \prod f^t(A)$, for this correspondence to carry over, we only need to check that if s denotes a map from $(g^t \circ \pi, g^r \circ \pi)$ to (f^t, f^r), and $\tilde{s}$ the adjoint correspondent to s, then $\tilde{s}$ preserves relations, and if $x \in g^t(A)$, then $\tilde{s}(x) \in \prod_{par} f^t(A)$. But since $(x,x) \in g^r(I(A)) = I(g^t(A))$, we must have $(\tilde{s}(x), \tilde{s}(x)) \in \prod f^r(I(A))$, so $\tilde{s}(x) \in \prod_{par} f^t(A)$ as desired. For the preservation of relations, suppose $(x,y) \in g^r(R)$. Then

$$(\tilde{s}(x), \tilde{s}(y)) \in \prod f^r(R) \cap \prod_{par} f^t(\partial_0 R) \times \prod_{par} f^t(\partial_1 R) = \prod_{par} f^r(R).$$

□

Corollary 8.9. *The fibration* $\mathrm{Fam}(\ \mathbf{LR}(\mathbf{D}) \substack{\longrightarrow \\ \longleftarrow \\ \longrightarrow} \mathbf{D}\ ) \to \mathbb{E}^G$ *is a* λ_2*-fibration.*

Remark 8.10. *If* $\mathbb{E}$ *is a topos then* $\mathbf{Q}$ *is the subobject fibration on* $\mathbf{D}$*, and* $\mathbb{T} \to \mathbb{K}$ *is in fact the model of* λ_2 *that Robinson and Rosolini prove to be parametric in the sense of reflexive graphs (Definition 7.1) in [15]. One interesting difference however, is that [15] considered only models of* λ_2 *that satisfied a "suitability for polymorphism" condition stating that the model is closed under* $\mathbf{LR}(\mathbf{D})_0$*-products. In our setup, this condition is replaced by the condition that the regular subobject fibration models* $\forall$*, and that the internal category* $\mathbf{D}$ *is closed under regular subobjects.*

Remark 8.11. *Consider a morphism ξ between types f and g in the model $\mathbb{T} \to \mathbb{K}$. At first sight, such a morphism is a pair of morphism (ξ_0, ξ_1) with $\xi_i : f_i \to g_i$. But morphisms in $\mathbf{LR}(\mathbf{D})$ are given by pairs of maps in $\mathbf{D}$, and commutativity of*

$$\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0^n \arrow[r, "\xi_1"] \arrow[d, "\partial_i"'] & \mathbf{LR}(\mathbf{D})_1 \arrow[d, "\partial_i"] \\
\mathbf{D}_0^n \arrow[r, "\xi_0"] & \mathbf{D}_1
\end{tikzcd}$$

tells us that ξ_1 must be given by (ξ_0, ξ_0). Thus morphisms between types are morphisms between the usual interpretations of types preserving the relational interpretations.

8.4 The APL-structure

In this section we embed the λ_2 fibration of Corollary 8.9 into a full parametric APL-structure.

Consider the functor $(\cdot)_0 : \mathbb{E}^G \to \mathbb{E}$ that maps a diagram $X_0 \overset{\longleftarrow}{\underset{\longleftarrow}{\longrightarrow}} X_1$ to X_0, and consider the pullback of (26) along $(\cdot)_0$:

$$\begin{tikzcd}
& \mathbb{P} \arrow[d] \\
\mathbb{T} \arrow[r, hook] \arrow[dr] & \mathbb{C} \arrow[d] \\
& \mathbb{E}^G.
\end{tikzcd} \tag{29}$$

Lemma 8.12. *The functor $(\cdot)_0$ extends to a morphism of fibrations:*

$$\begin{tikzcd}
\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D}) \\ \downarrow\uparrow\downarrow \\ \mathbf{D}\end{pmatrix} \arrow[r, "(\cdot)_0"] \arrow[d] & \mathrm{Fam}(\mathbf{D}) \arrow[d] \\
\mathbb{E}^G \arrow[r, "(\cdot)_0"] & \mathbb{E}.
\end{tikzcd}$$

Proof. The required map maps an object

$$\begin{pmatrix}X_1 \\ \downarrow\uparrow\downarrow \\ X_0\end{pmatrix} \longrightarrow \begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}$$

of $\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}$ to the object $X_0 \longrightarrow \mathbf{D}_0$ of $\mathrm{Fam}(\mathbf{D})$. Likewise for morphisms. □

As a consequence of Lemma 8.12 we can extend (29) to

$$\begin{tikzcd}
& & \mathbb{P} \arrow[d] \\
\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix} \arrow[r] \arrow[drr] & \mathbb{T} \arrow[r, hook] \arrow[dr] & \mathbb{C} \arrow[d] \\
& & \mathbb{E}^G.
\end{tikzcd} \tag{30}$$

If we erase $\mathbb{T}$ from (30) we obtain the diagram

$$\begin{tikzcd}
 & \mathbb{P} \arrow[d] \\
\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D}) \\ \downarrow\uparrow\downarrow \\ \mathbf{D}\end{pmatrix} \arrow[r, hook, "I"] \arrow[dr] & \mathbb{C} \arrow[d] \\
 & \mathbb{E}^G.
\end{tikzcd} \tag{31}$$

Theorem 8.13. *The diagram (31) defines a parametric APL-structure.*

We will prove Theorem 8.13 in a series of lemmas.

Corollary 8.14. *If $\mathbf{D}$ is an internal model of λ_2 in a topos, which is closed under subobjects, then the parametric completion process of [15] provides a λ_2-fibration that satisfies the consequences of parametricity provable in Abadi & Plotkin's logic.*

Proof. This follows from Remark 8.10. □

Remark 8.15. *The types (the objects of* $\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D}) \\ \downarrow\uparrow\downarrow \\ \mathbf{D}\end{pmatrix}$*) in the APL-structure (31) are morphisms*

$$\begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}^n \to \begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}$$

in $\mathbb{E}^G$. Thus types contain both the usual interpretation (the map $f_0 : \mathbf{D}_0^n \to \mathbf{D}_0$) and a relational interpretation (the map $f_1 : \mathbf{LR}(\mathbf{D})_0^n \to \mathbf{LR}(\mathbf{D})_0$). But since the map $\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix} \to \mathbb{T}$ *forgets the relational interpretation, the logic on types, given by $\mathbb{P}$, is given only by the logic on the usual interpretation of the types. To be more precise, a logical relation in the model of (31) between types f and g is a relation in the sense of the logic $\mathbb{Q}$ between $\coprod_{\vec{d}\in\mathbf{D}_0^n} f_0(\vec{d}) \to \mathbf{D}_0^n$ and $\coprod_{\vec{d}\in\mathbf{D}_0^n} g_0(\vec{d}) \to \mathbf{D}_0^n$.*

Notice also that the relational interpretation of a type (given by f_1) is in a sense parametric since the diagram

$$\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0^n \arrow[r, "f_1"] & \mathbf{LR}(\mathbf{D})_0 \\
\mathbf{D}_0^n \arrow[u, "i"] \arrow[r, "f_0"] & \mathbf{D}_0 \arrow[u, "i"']
\end{tikzcd}$$

is required to commute. This is basically the reason why the APL-structure is parametric.

Remark 8.16. *One may restrict the APL-structure of (31) to the full subcategory of $\mathbb{E}^G$ on powers of the generic object. This way one obtains a λ_2-fibration in which* Type *is the only kind. To prove that this defines a parametric APL-structure, one will need to change the proof presented here slightly to obtain the reflexive graph.*

Lemma 8.17. $\mathbb{C} \to \mathbb{K}$ *is fibred cartesian closed and I is a faithful product-preserving functor.*

Proof. The first statement follows from the fact that $\mathbb{E}^{\to} \to \mathbb{E}$ is a fibred cartesian closed fibration.

I is a restriction of the composition

$$\begin{tikzcd}
\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D}) \\ \downarrow\uparrow\downarrow \\ \mathbf{D}\end{pmatrix} \arrow[r] \arrow[drr] & \mathbb{T} \arrow[r, hook] \arrow[dr] & \mathbb{C}' \arrow[d] & . \\
& & \mathbb{E}^G &
\end{tikzcd}$$

The map $\mathbb{T} \to \mathbb{C}'$ is the pullback of the inclusion of the externalization of a full internal cartesian closed category into $\mathbb{E}^{\to}$. This is faithful and product preserving by assumption.

The map $\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D}) \\ \downarrow\uparrow\downarrow \\ \mathbf{D}\end{pmatrix} \to \mathbb{T}$ is the map that maps

$$f : \begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}^n \to \begin{pmatrix}\mathbf{LR}(\mathbf{D})_i \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_i\end{pmatrix}$$

to $f_0 : \mathbf{D}_0^n \to \mathbf{D}_i$ (for $i = 0, 1$ denoting objects and morphisms respectively). Since product structure of internal categories of graph categories is given pointwise, this map clearly preserves fibred products.

As mentioned in Remark 8.11, a morphism from f to g with

$$f, g : \begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}^n \to \begin{pmatrix}\mathbf{LR}(\mathbf{D})_0 \\ \downarrow\uparrow\downarrow \\ \mathbf{D}_0\end{pmatrix}$$

is just a map from f_0 to g_0 preserving relations. Thus the first map is also faithful. □

Lemma 8.18. *The composable fibration $\mathbb{P} \to \mathbb{C} \to \mathbb{K}$ is an indexed first-order logic fibration with an indexed family of generic objects. Moreover, the composable fibration has simple products, simple coproducts and very strong equality.*

Proof. The composable fibration $\mathbb{P} \to \mathbb{C} \to \mathbb{K}$ is a pullback of $\mathbb{Q} \to \mathbb{E}^{\to} \to \mathbb{E}$ which has the desired properties according to Lemma A.8. All of this structure is always preserved under pullback, except simple products and coproducts. These are preserved since the map $\mathbb{K} \to \mathbb{E}$ preserves products. □

As in Remark 3.4 we can now construct the functor U as needed in Definition 3.3. Thus we have:

Proposition 8.19. *The diagram (31) defines a pre-APL-structure with very strong equality.*

Consider the graph W:

$$\begin{tikzcd}
\cdot \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] & & \cdot \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] \\
& \cdot \arrow[dl] \arrow[dr] & \\
\cdot & & \cdot
\end{tikzcd}$$

where we assume that the two graphs included are reflexive graphs. The graph $\mathbf{W}$:

$$\begin{tikzcd}
\mathbf{LR}(\mathbf{D}) \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] & & \mathbf{LR}(\mathbf{D}) \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] \\
& \mathbf{LR}(\mathbf{D}) \arrow[dl] \arrow[dr] & \\
\mathbf{D} & & \mathbf{D}
\end{tikzcd}$$

defines an internal category in $\mathbb{E}^W$.

An object of $\mathrm{Fam}(\mathbf{W})$ can be denoted by a triple (f, g, ρ), where f and g are types in the same fibre (that is, objects of $\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D})\\ \downarrow\uparrow\downarrow\\ \mathbf{D}\end{pmatrix}$ in the same fibre) and ρ is a morphism $\mathbf{LR}(\mathbf{D})_0^n \to \mathbf{LR}(\mathbf{D})_0$ such that the diagram

$$\begin{tikzcd}
& \mathbf{LR}(\mathbf{D})_0^n \arrow[rr, "\rho"] \arrow[ddl] \arrow[ddr] & & \mathbf{LR}(\mathbf{D})_0 \arrow[dl] \arrow[dr] & \\
& & \mathbf{D}_0 & & \mathbf{D}_0 \\
\mathbf{D}_0^n \arrow[urr, "f_0"] & & \mathbf{D}_0^n \arrow[urr, "g_0"] & &
\end{tikzcd} \tag{32}$$

commutes.

Now, as noted in Remark 8.15 types in the pre-APL structure (31) are given by both an ordinary interpretation of types and a relational interpretation of types, but relations between types are just given by relations between the ordinary interpretation of types. Thus we may think of such triples as objects of the form

$$[\![\vec{\alpha}, \vec{\beta} \mid \vec{R} \colon \mathsf{Rel}(\vec{\alpha}, \vec{\beta}) \vdash \phi(R) \colon \mathsf{Rel}(f(\vec{\alpha}), g(\vec{\beta}))]\!]$$

in the category $\mathbf{Relations}$ as formed from the pre-APL structure (31), in the same way as in Lemma 8.6.

Note that since we have proved that the diagram (31) defines a pre-APL-structure, we can reason about it using the parts of Abadi & Plotkin's logic not involving the relational interpretation of types. In the following we shall use this to work in the internal language of the pre-APL-structure.

Proposition 8.20. *There is an isomorphism of fibrations:*

$$\begin{pmatrix}\mathrm{Fam}(\mathbf{W})\\ \Big\downarrow\\ \mathbb{E}^W\end{pmatrix} \xrightarrow{\cong} \begin{pmatrix}\mathbf{Relations}\\ \Big\downarrow\\ \mathbf{RelCtx}\end{pmatrix}$$

Proof. The argument is essentially the same as the proof of Lemma 8.6. □

Lemma 8.21. *The graph* $\mathbf{W}$ *is an internal model of* λ_2 *in* $\mathbb{E}^W$*.*

Proof. This is a consequence of Proposition 8.20. □

Proposition 8.22. *There is a reflexive graph of* λ_2*-fibrations*

$$\begin{pmatrix}\mathrm{Fam}\begin{pmatrix}\mathbf{LR}(\mathbf{D})\\ \downarrow\uparrow\downarrow\\ \mathbf{D}\end{pmatrix}\\ \Big\downarrow\\ \mathbb{E}^G\end{pmatrix} \begin{matrix}\longleftarrow\\ \longrightarrow\\ \longleftarrow\end{matrix} \begin{pmatrix}\mathrm{Fam}(\mathbf{W})\\ \Big\downarrow\\ \mathbb{E}^W\end{pmatrix}$$

Remark 8.23. *The reflexive graph in [15] arises this way, although the setup of [15] is slightly different.*

Proof. An object of $\mathrm{Fam}(\mathbf{W})$ is a map in $\mathbb{E}^W$

$$\left(\begin{tikzcd}
\Xi_1 \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \Xi_4 \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
 & \Xi_3 \arrow[dl] \arrow[dr] & \\
\Xi_2 \arrow[uu] & & \Xi_5 \arrow[uu]
\end{tikzcd}\right) \to \left(\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \mathbf{LR}(\mathbf{D})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
 & \mathbf{LR}(\mathbf{D})_0 \arrow[dl] \arrow[dr] & \\
\mathbf{D}_0 \arrow[uu] & & \mathbf{D}_0 \arrow[uu]
\end{tikzcd}\right).$$

Let us denote such objects as triples (f, g, ρ) where $f\colon \left(\begin{tikzcd}\Xi_1 \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \Xi_2 \arrow[u]\end{tikzcd}\right) \to \left(\begin{tikzcd}\mathbf{LR}(\mathbf{D})_0 \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \mathbf{D}_0 \arrow[u]\end{tikzcd}\right)$, $g : \left(\begin{tikzcd}\Xi_4 \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \Xi_5 \arrow[u]\end{tikzcd}\right) \to \left(\begin{tikzcd}\mathbf{LR}(\mathbf{D})_0 \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \mathbf{D}_0 \arrow[u]\end{tikzcd}\right)$ and $\rho : \Xi_3 \to \mathbf{LR}(\mathbf{D})_0$. The domain and codomain maps of the postulated reflexive graph map (f, g, ρ) to f and g respectively, and the last map maps f to (f, f, f_1).

The domain and codomain map preserve simple products since from the viewpoint of Proposition 8.22 these are just the domain and codomain map of Lemma 3.7. The middle map component of the simple products in $\mathrm{Fam}(\mathbf{W}) \to \mathbb{E}^W$ is computed by computing the simple products as in Lemma 8.7 and then restricting the the right domain and codomain. Since this is the same as the computation of the relational part of the simple products of $\mathrm{Fam}\left(\begin{tikzcd}\mathbf{LR}(\mathbf{D}) \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \mathbf{D} \arrow[u]\end{tikzcd}\right)$, the last map of the reflexive graph also commutes with simple products. □

Proposition 8.24. *The pre-APL-structure (31) has a full APL-structure.*

Proof. This follows from Proposition 8.22 and Proposition 8.20. □

Lemma 8.25. *The APL-structure (31) satisfies extensionality.*

Proof. The model has very strong equality, which implies extensionality (4.2). □

Lemma 8.26. *The APL-structure (31) satisfies the identity extension axiom.*

Proof. Consider a type f with n free variables. We need to show that

$$\langle id_{\Omega^n}, id_{\Omega^n}\rangle^* J(f) \circ [\![\vec{\alpha} \mid - \mid - \vdash eq_{\vec{\alpha}}]\!] = [\![\vec{\alpha} \vdash eq_{f(\vec{\alpha})}]\!].$$

The map J is defined as the composition of two maps. The first map maps f to (f, f, f_1) :

$$\left(\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0^n \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \mathbf{LR}(\mathbf{D})_0^n \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
 & \mathbf{LR}(\mathbf{D})_0^n \arrow[dl] \arrow[dr] & \\
\mathbf{D}_0^n \arrow[uu] & & \mathbf{D}_0^n \arrow[uu]
\end{tikzcd}\right) \to \left(\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \mathbf{LR}(\mathbf{D})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
 & \mathbf{LR}(\mathbf{D})_0 \arrow[dl] \arrow[dr] & \\
\mathbf{D}_0 \arrow[uu] & & \mathbf{D}_0 \arrow[uu]
\end{tikzcd}\right).$$

Since f makes the diagram

$$\begin{tikzcd}
\mathbf{LR}(\mathbf{D})_0^n \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR}(\mathbf{D})_0 \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\mathbf{D}_0^n \arrow[r, "f_0"] \arrow[u] & \mathbf{D}_0 \arrow[u]
\end{tikzcd}$$

commute we know that $f_1(eq_{\vec{\alpha}}) = eq_{f_0(\vec{\alpha})}$. □

Theorem 8.13 is now the collected statement of 8.19 8.24, 8.25 and 8.26.

Remark 8.27. *As mentioned in the introduction to this section, the concrete APL-structure of Section 6 can be considered as a result of the parametric completion process. If we consider the internal category* $\mathbf{Per}$ *in the category* $\mathbf{Asm}$ *of assemblies, then using the parametric completion process on this data we obtain the APL-structure of Section 6. To see this, we need to use the fact that there exists an isomorphism of fibrations*

$$\begin{tikzcd}
\mathbf{UFam}(\mathbf{Asm}) \arrow[rr, "\cong"] \arrow[dr] & & \mathbf{Asm}^{\rightarrow} \arrow[dl] \\
& \mathbf{Asm}. &
\end{tikzcd}$$

This proves Theorem 6.2.

9 Parametric Internal Models

The definition of APL-structure admittedly asks for a substantial amount of structure. In this section we sketch how much of that structure may be derived in the case of internal models of λ_2.

Let $\mathbb{E}$ be a quasi-topos and let j be a local operator (also known as closure operator or Lawvere-Tierney topology) on $\mathbb{E}$. We write $\mathbb{E}_j$ for the full subcategory of j-sheaves, $\mathbf{a}$ for the associated sheaf functor, I for the inclusion of j-sheaves, and η for the natural transformation $Id \to I\,\mathbf{a}$.

Let $\mathbf{C}$ be an internal model of λ_2 $\mathbb{E}$. Then $\mathbf{a}\,\mathbf{C}$ is an internal category in $\mathbb{E}_j$ and $\eta : \mathbf{C} \to \mathbf{a}\,\mathbf{C}$ is an internal functor.

Consider the following diagram:

$$\begin{tikzcd}
& \mathbb{P} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{S} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{RegSub}_{\mathbb{E}_j} \arrow[d] \\
\mathbf{Fam}(\mathbf{C}) \arrow[r, "I"] \arrow[dr] & \mathbf{Fam}(\mathbb{E}_j) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & {\mathbb{E}_j}^{\rightarrow} \arrow[r, "\mathrm{dom}"] \arrow[d] & \mathbb{E}_j \\
& \mathbb{E} \arrow[r, "\mathbf{a}"'] & \mathbb{E}_j &
\end{tikzcd}$$

(33)

where I is the functor induced by the composition of the internal functor $\eta : \mathbf{C} \to \mathbf{a}\,\mathbf{C}$ and the inclusion of the externalization of $\mathbf{C}$ into ${\mathbb{E}_j}^{\rightarrow}$ is faithful.

Suppose that

- the internal functor $\eta : \mathbf{C} \to \mathbf{a}\,\mathbf{C}$ is faithful,
- the internal category $\mathbf{a}\,\mathbf{C}$ is a subcategory of $\mathbb{E}_j$ (i.e., the inclusion of the externalization of $\mathbf{C}$ into ${\mathbb{E}_j}^{\rightarrow}$ is faithful).

Then the functor I in the above diagram is faithful and the leftmost part of the diagram (33) (the part going down and left from $\mathbb{P}$) is a pre-APL-structure, and we can thus define that **the internal λ_2 model $\mathbf{C}$ in $\mathbb{E}$ is parametric with respect to** j if this pre-APL-structure is a parametric APL-structure.

One should, of course, think of j as specifying the logic with respect to which the model is parametric.

The completion process presented in the previous section takes a full internal λ_2 model in a quasi-topos $\mathbb{F}$ and produces an internal model in $\mathbb{E} = \mathbb{F}^G$ with j on $\mathbb{E}$ such that $\mathbb{F} = \mathbb{E}_j$ (the associated sheaf functor

$\mathbf{a}$ takes $\begin{smallmatrix} X_1 \\ \downarrow\uparrow\downarrow \\ X_0 \end{smallmatrix}$ to X_0) and which satisfies the two items above ensuring that I is faithful. The results in the previous section then show that the internal model in $\mathbb{F}$ is parametric with respect to this j.

This description of parametric internal models allows us to state precisely the (still) open problem of whether there exists parametric models that are inherently parametric (not constructed though a completion process):

Problem 9.1. *Does there exist a full internal λ_2 model in a quasi-topos $\mathbb{E}$ that is parametric with respect to the trivial topology j (such that $\mathbb{E}_j = \mathbb{E}$) ?*

10 Conclusion

We have defined the notion of an APL-structure and proved that it provides sound and complete models for Abadi and Plotkin's logic for parametricity, thereby answering a question posed in [12, page 5]. We have also defined a notion of parametric APL-structures, for which we can prove the expected consequences of parametricity using the internal logic. The consequences proved in this document are existence of inductive and coinductive datatypes. These consequences have, to our knowledge not been proved in general for models parametric in the sense of Ma & Reynolds, but only for specific models.

We have presented a family of parametric models, some of which are not well-pointed. This means that our notion of parametricity is useful also in the absence of well-pointedness.

We have provided an extension of the parametric completion process of [15] that produces parametric APL-structures. This means that for a large class of models, we have proved that the parametric completion of Robinson and Rosolini produce models that satisfy the consequences of parametricity.

In subsequent papers we will show how to modify the parametric completion process to produce domain-theoretic parametric models and how to extend the notion of APL-structure to include models of polymorphic linear lambda calculus [11].

A Composable Fibrations

This appendix is concerned with the theory of composable fibrations, by which we simply mean pairs of fibrations such that the codomain of the first is the domain of the second fibration. This appendix contains definitions referred to in the text.

Suppose we are given a composable fibration:

$$\begin{tikzcd} \mathbb{F} \arrow[r, "p"] & \mathbb{E} \arrow[r, "q"] & \mathbb{B} \end{tikzcd}$$

We observe that

- The composite qp is a fibration. This is easily seen from the definition.
- If p and q are cloven, we may choose a cleavage by lifting u twice to $\overline{\overline{u}}$ for each I in $\mathrm{Obj}\,\mathbb{F}$ and $u : X \to qpI$.
- If p, q are split the composite fibration will be split since $\overline{\overline{vu}} = \overline{\overline{v} \circ \overline{u}} = \overline{\overline{v}} \circ \overline{\overline{u}}$.

Thus in the case above we may consider the composable fibration as a doubly indexed category, and reindexing in $\mathbb{F}$ with respect to u in $\mathbb{B}$ is given by $\overline{u}^*$

The lemmas below refer to the fibrations p, q above.

Definition A.1. *We say that $(\Omega_A)_{A\in \mathrm{Obj}\,\mathbb{B}}$ is an indexed family of generic objects for the composable pair of fibrations (p, q) if for all A, $\Omega_A \in \mathrm{Obj}\,\mathbb{E}_A$ is a generic object for the restriction of p to $\mathbb{E}_A$ and if the family is closed under reindexing, ie., for all morphisms $u : A \to B$ in $\mathbb{B}$, $u^*(\Omega_B) \cong \Omega_A$.*

Before we define the concept of an indexed first-order logic fibration, we recall the definition of first-order logic fibration from [5] .

Definition A.2. *A fibration $p : \mathbb{F} \to \mathbb{E}$ is called a* first-order logic fibration *if*

- *p is a fibred preorder that is fibred bicartesian closed.*
- *$\mathbb{E}$ has products.*
- *p has simple products and coproducts, i.e., right, respectively left adjoints to reindexing functors induced by projections, and these satisfy the Beck-Chevalley condition.*
- *p has fibred equality, i.e., left adjoints to reindexing functors induced by $id \times \Delta : I \times J \to I \times J \times J$, satisfying the Beck-Chevalley condition.*

Readers worried about the Frobenius condition should note that this comes for free in fibred cartesian closed categories.

Definition A.3. *We say that (p, q) has indexed (simple) products/coproducts/equality if each restriction of p to a fibre of q has the same satisfying the Beck-Chevalley condition, and these commute with reindexing, i.e., if u is a map in $\mathbb{B}$ then there is a natural isomorphism $\bar{u}^* \prod_f \cong \prod_{u^*f} \bar{u}^*$ or $\bar{u}^* \coprod_f \cong \coprod_{u^*f} \bar{u}^*$ (this can also be viewed as a Beck-Chevalley condition).*

Definition A.4. *We say that (p, q) is an* indexed first order logic fibration *if p is a fibrewise bicartesian closed preorder, and (p, q) has indexed simple products, indexed simple coproducts and indexed equality.*

We can also talk about composable fibrations (p, q) simply having products, coproducts, etc. This should be the case if the composite qp has (co-)products, but we should also require the right Beck-Chevalley conditions to hold. Notice that since u^* in qp is the same as $\bar{u}^*$ in p we can write the product as either $\prod_u$ in qp or $\prod_{\bar{u}}$ in p.

Definition A.5. *We say that the composable fibration (p, q) has products / coproducts if for each map $u \colon I \to J$ in $\mathbb{B}$, and each object $X \in \mathbb{E}_J$ the reindexing functor $\bar{u}^* \colon \mathbb{F}_X \to \mathbb{F}_{u^*X}$ has a right / left adjoint. Moreover, these (co)-products must satisfy the Beck-Chevalley condition for two sorts of diagram corresponding to reindexing in $\mathbb{B}$ and $\mathbb{E}$ respectively. First if*

$$\begin{tikzcd}
H \arrow[r, "v"] \arrow[d, "a"'] \arrow[dr, phantom, "\lrcorner", very near start] & K \arrow[d, "b"] \\
I \arrow[r, "u"] & J
\end{tikzcd}$$

is a pullback diagram in $\mathbb{B}$, then by [5, Exercise 1.4.4]

$$\begin{tikzcd}
a^*u^*X \arrow[r, "\bar{v}"] \arrow[d, "\bar{a}"'] & b^*X \arrow[d, "\bar{b}"] \\
u^*X \arrow[r, "\bar{u}"] & X
\end{tikzcd}$$

is a pullback diagram in $\mathbb{E}$, and we require that the Beck-Chevalley condition is satisfied with respect to this diagram. Second, if $f\colon Y \to X$ is a vertical map in $\mathbb{E}$, then the Beck-Chevalley condition should be satisfied with respect to the diagram

$$\begin{tikzcd} u^*Y \arrow[r, "\bar{u}"] \arrow[d, "u^*f"'] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d, "f"] \\ u^*X \arrow[r, "\bar{u}"] & X \end{tikzcd} \tag{34}$$

which by the way is a pullback by [5, Exercise 1.4.4].

The composable fibration (p, q) has simple (co-)products if it has (co-)products with respect to projections as defined above.

In the case of the APL-structures, the logical content of the Beck-Chevalley condition for diagrams of the form (34) will be that

$$(\forall \alpha\colon \mathsf{Type}.\,\phi)[t/x] = \forall \alpha\colon \mathsf{Type}.\,(\phi[t/x]).$$

Definition A.6. *We say that a first-order logic fibration has* very strong equality *if internal equality in the fibration implies external equality.*

Definition A.7. *We say that the indexed first order logic fibration (p, q) has* very strong equality *if each restriction of p to a fibre of q has.*

The next lemma gives a way of obtaining indexed first-order logic fibrations.

Lemma A.8. *Suppose $\mathbb{Q}' \to \mathbb{E}$ is a first-order logic fibration with a generic object on a locally cartesian closed category $\mathbb{E}$. Suppose further, that $\mathbb{Q}' \to \mathbb{E}$ has products and coproducts with respect to maps $A \times_B A' \to A$ from pullback diagrams*

$$\begin{tikzcd} A \times_B A' \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d] \\ A' \arrow[r] & B, \end{tikzcd}$$

and coproducts with respect to maps

$$id_C \times_B \Delta_A \colon C \times_B \times A \to C \times_B A \times_B A,$$

all satisfying the Beck-Chevalley condition. Then the composable fibration

$$\begin{tikzcd} \mathbb{Q} \arrow[r] & \mathbb{E}^{\to} \arrow[r, "\mathrm{cod}"] & \mathbb{E}\ , \end{tikzcd}$$

where $\mathbb{Q} \to \mathbb{E}^{\to}$ is the pullback

$$\begin{tikzcd} \mathbb{Q} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{Q}' \arrow[d] \\ \mathbb{E}^{\to} \arrow[r, "\mathrm{dom}"] & \mathbb{E}, \end{tikzcd}$$

is an indexed first-order logic fibration with an indexed family of generic objects, simple products and simple coproducts. Moreover, if $\mathbb{Q}' \to \mathbb{E}$ has very strong equality, so does the composable fibration.

Proof. The fibred bicartesian structure exists since the fibres of $\mathbb{Q} \to \mathbb{E}^{\to}$ are the fibres of $\mathbb{Q}' \to \mathbb{E}$. This structure is clearly preserved by reindexing.

The fibrewise product of $A \to B$ and $A' \to B$ in $\mathbb{E}^{\to}$ is $A \times_B A' \to B$ with projection

$$\begin{tikzcd}
A \times_B A' \arrow[rr, "\pi"] \arrow[dr] & & A . \arrow[dl] \\
& B &
\end{tikzcd}$$

The indexed (co-)product along this map in $\mathbb{Q} \to \mathbb{E}^{\to}$ is the (co-)product along π in $\mathbb{E}$, which exists by assumption. For the Beck-Chevalley condition for vertical pullbacks, recall that the domain functor $\mathbb{E}^{\to} \to \mathbb{E}$ preserves pullbacks, so for a vertical map

$$\begin{tikzcd}
A'' \arrow[rr, "f"] \arrow[dr] & & A \arrow[dl] \\
& B &
\end{tikzcd}$$

taking the pullback of π along f in the category $\mathbb{E}^{\to}$, and then applying the domain functor gives the pullback

$$\begin{tikzcd}
A'' \times_B A' \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & A \times_B A' \arrow[d] \\
A'' \arrow[r, "f"] & A
\end{tikzcd}$$

in $\mathbb{E}$, so that the Beck-Chevalley condition in this case reduces to Beck-Chevalley for the fibration $\mathbb{Q}' \to \mathbb{E}$.

To prove that these indexed simple (co-)products commute with reindexing, consider a map $u \colon B' \to B$ in $\mathbb{E}$. We need to prove that for the diagram

$$\begin{tikzcd}
& u^*(A) \times_{B'} u^*(A') \arrow[rr, "\bar{u}"] \arrow[dl, "\pi"] \arrow[dd] & & A \times_B A' \arrow[dl, "\pi"'] \arrow[dd] \\
u^*A \arrow[rr, "\bar{u}"] \arrow[dr] & & A \arrow[dr] & \\
& B' \arrow[rr, "u"] & & B,
\end{tikzcd}$$

we have, for products $\bar{u}^* \prod_\pi \cong \prod_\pi \bar{u}^*$ and for coproducts $\bar{u}^* \coprod_\pi \cong \coprod_\pi \bar{u}^*$. But this follows from the Beck-Chevalley condition in $\mathbb{Q}' \to \mathbb{E}$.

Indexed fibred equality is given by coproduct along maps

$$id_C \times_B \Delta_A \colon C \times_B A \to C \times_B A \times_B A,$$

which are required to exists. As with indexed (co-)products, the Beck-Chevalley conditions reduce to the Beck-Chevalley conditions for $\mathbb{Q}' \to \mathbb{E}$.

We define the family of generic objects to be the projections $(\Sigma \times B \to B)_{B \in \mathbb{E}}$ in $\mathbb{E}^{\to}$ where Σ is the generic object of $\mathbb{Q} \to \mathbb{E}$. This family is clearly closed under reindexing, and maps

$$\begin{tikzcd}
A \arrow[rr, "h"] \arrow[dr, "f"'] & & \Sigma \times B \arrow[dl, "\pi"] \\
& B &
\end{tikzcd}$$

correspond to maps $A \to \Sigma$ in $\mathbb{E}$, which correspond to objects of $\mathbb{Q}'_A \cong \mathbb{Q}_f$.

We shall prove that we have simple products; simple coproducts are proved similarly. Suppose $\pi : D \times D' \to D$ is a projection in $\mathbb{E}$. For $f : A \to D$ in $\mathbb{E}^{\to}$, $\bar{\pi}$ is the map

$$\begin{tikzcd}
A \times D' \arrow[r, "\pi"] \arrow[d, "f \times id"'] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d, "f"] \\
D \times D' \arrow[r, "\pi"] & D.
\end{tikzcd}$$

Reindexing along this map in $\mathbb{Q}$ corresponds to reindexing in $\mathbb{Q}'$ along $\pi : A \times D' \to A$, so by existence of simple products in $\mathbb{Q}' \to \mathbb{E}$ we have a right adjoint $\pi^* \dashv \prod_\pi$.

We need to prove Beck-Chevalley first for pullbacks in $\mathbb{E}$. In this case a pullback in $\mathbb{E}$

$$\begin{tikzcd}
D \times D'' \arrow[r, "id_D \times u"] \arrow[d, "\pi'"] \arrow[dr, phantom, "\lrcorner", very near start] & D \times D' \arrow[d, "\pi'"] \\
D'' \arrow[r, "u"] & D'
\end{tikzcd}$$

lifts to the pullback

$$\begin{tikzcd}
& D \times u^*A \arrow[rr, "id \times \bar{u}"] \arrow[dl, "\pi'"'] \arrow[dd] & & D \times A \arrow[dl, "\pi'"] \arrow[dd, "id \times f"] \\
u^*A \arrow[rr, "\bar{u}"] \arrow[dd] & & A \arrow[dd, "f"] & \\
& D \times D'' \arrow[rr, "id \times u"] \arrow[dl, "\pi'"'] & & D \times D' \arrow[dl, "\pi'"] \\
D'' \arrow[rr, "u"] & & D' &
\end{tikzcd}$$

in $\mathbb{E}^{\to}$. The Beck-Chevalley condition for this pullback reduces to the Beck-Chevalley condition for the upper square in $\mathbb{Q}' \to \mathbb{E}$ which is known to hold.

We should also check that the Beck-Chevalley condition holds in the case of the pullback.

$$\begin{tikzcd}
& A' \times D' \arrow[rr, "\bar{\pi}"] \arrow[dl, "h \times id"'] \arrow[dd] & & A' \arrow[dl, "h"] \arrow[dd] \\
A \times D' \arrow[rr, "\bar{\pi}"] \arrow[dr] & & A \arrow[dr] & \\
& D \times D' \arrow[rr, "\pi"] & & D
\end{tikzcd}$$

But again this reduces to the Beck-Chevalley condition for $\mathbb{Q}' \to \mathbb{E}$ because $\bar{\pi}$ is a projection.

Very strong equality is clearly preserved. □

References

[1] B.P. Dunphy. *Parametricity as a notion of uniformity in reflexive graphs*. PhD thesis, 2004. 1, 6.1

[2] Brian Dunphy and Uday S. Reddy. Parametric limits. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS-04))*, pages 242–251, 2004.

[3] R. Hasegawa. Categorical data types in parametric polymorphism. *Mathematical Structures in Computer Science*, 4:71–109, 1994. 1, 5

[4] J.M.E. Hyland, E.P. Robinson, and G. Rosolini. The discrete objects in the effective topos. *Proc. London Math. Soc.*, 3(60):1–36, 1990. 1

[5] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 1, 2, 2.1, 3.1, 4, 6, 6, 6, 7, 8.1, A, A.5, A

[6] Q. Ma and J.C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40. Springer-Verlag, 1992. 1, 1, 7

[7] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971. 8.2

[8] J.C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996. 1

[9] A. M. Pitts. Non-trivial power types can't be subtypes of polymorphic types. In *4th Annual Symposium on Logic in Computer Science*, pages 6–13. IEEE Computer Society Press, Washington, 1989. 1

[10] A.M. Pitts. Polymorphism is set theoretic, constructively. In D. H. Pitt, A. Poigné, and D. E. Rydeheard, editors, *Category Theory and Computer Science, Proc. Edinburgh 1987*, volume 283 of *Lecture Notes in Computer Science*, pages 12–39. Springer-Verlag, 1987. 1

[11] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1, 10

[12] Gordon Plotkin and Martín Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, volume 664 of *Lecture Notes in Comput. Sci.*, pages 361–375. Springer, Berlin, 1993. (document), 1, 2, 2.3, 2.4, 3.2, 3.18, 4, 5, 5.1, 5.4, 5.16, 10

[13] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983. 1

[14] J.C. Reynolds. Polymorphism is not set-theoretic. In G. Kahn, D. B. MacQueen, and G. D. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 145–156. Springer-Verlag, 1984. 1

[15] E.P. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In S. Abramsky, editor, *Proc. 9th Symposium in Logic in Computer Science*, pages 364–371, Paris, 1994. I.E.E.E. Computer Society. 1, 7, 8, 8.10, 8.14, 8.23, 10

[16] G. Rosolini. Notes on synthetic domain theory. Draft, 1995.

[17] I. Rummelhoff. Polynat in PER-models. *Theoretical Computer Science*, 316(1–3):215–224, May 2004.

[18] R.A.G. Seely. Categorical semantics of higher-order polymorphic lambda calculus. *The Journal of Symbolic Logic*, 52(4):969–989, December 1987. 1

[19] Izumi Takeuti. An axiomatic system of parametricity. *Fund. Inform.*, 33(4):397–432, 1998. Typed lambda-calculi and applications (Nancy, 1997). 2.2

[20] P. Wadler. Theorems for free! In *4'th Symposium on Functional Programming Languages and Computer Architecture, ACM, London*, September 1989. 1, 1

[21] P. Wadler. The Girard-Reynolds isomorphism (second edition). Manuscript, March 2004. 2.2

Parametric Domain-theoretic models of Linear Abadi & Plotkin Logic

Lars Birkedal
Rasmus Ejlers Møgelberg
Rasmus Lerchedahl Petersen

Abstract

We present a formalization of a linear version of Abadi and Plotkin's logic for parametricity for a polymorphic dual intuitionistic / linear type theory with fixed points, and show, following Plotkin's suggestions, that it can be used to define a wide collection of types, including solutions to recursive domain equations. We further define a notion of parametric LAPL-structure and prove that it provides a sound and complete class of models for the logic. Finally, we present a concrete parametric parametric LAPL-structure based on suitable categories of partial equivalence relations over a universal model of the untyped lambda calculus.

Contents

1 Introduction

In this paper we show how to define parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. The work is motivated by two different observations, due to Reynolds and Plotkin.

In 1983 Reynolds argued that parametric models of the second-order lambda calculus are very useful for modeling data abstraction in programming [25] (see also [20] for a recent textbook description). For real programming, one is of course not just interested in a strongly terminating calculus such as the second-order lambda calculus, but also in a language with full recursion. Thus in *loc. cit.* Reynolds also asked for a parametric *domain-theoretic* model of polymorphism. Informally, what is meant [26] by this is a model of an extension of the polymorphic lambda calculus [24, 10], with a polymorphic fixed-point operator $Y : \forall\alpha.\, (\alpha \to \alpha) \to \alpha$ such that

1. types are modeled as domains, the sublanguage without polymorphism is modeled in the standard way and $Y\sigma$ is the least fixed-point operator for the domain σ;
2. the logical relations theorem (also known as the abstraction theorem) is satisfied when the logical relations are admissible, i.e., strict and closed under limits of chains;
3. every value in the domain representing some polymorphic type is parametric in the sense that it satisfies the logical relations theorem (even if it is not the interpretation of any expression of that type).

Of course, this informal description leaves room for different formalizations of the problem. Even so, it has proved to be a non-trivial problem. Unpublished work of Plotkin [22] indicates one way to solve the problem model-theoretically by using strict, admissible partial equivalence relations over a domain model of the untyped lambda calculus but, as far as we know, the details of this relationally parametric model have not been worked out in detail before. (We do that here.) In *loc. cit.* Plotkin also suggested that one should consider parametric domain-theoretic models not only of polymorphic lambda calculus but of polymorphic intuitionistic / linear lambda calculus, since this would give a way to distinguish, in the calculus, between strict and possibly non-strict continuous functions, and since some type constructions, e.g., coproducts, should not be modeled in a cartesian closed category with fixed points [11]. Indeed Plotkin argued that such a calculus could serve as a very powerful metalanguage for domain theory in which one could also encode recursive types, using parametricity. To prove such consequences of parametricity, Plotkin suggested to use a linear version of Abadi and Plotkin's logic for parametricity [23] with fixed points.

Thus parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus are of import both from a programming language perspective (for modeling data abstraction) and from a purely domain-theoretic perspective.

Recently, Pitts and coworkers [21, 4] have presented a syntactic approach to Reynolds' challenge, where the notion of domain is essentially taken to be equivalence classes of terms modulo a particular notion of contextual equivalence derived from an operational semantics for a language called Lily, which is essentially polymorphic intuitionistic / linear lambda calculus endowed with an operational semantics.

In parallel with the work presented here, Rosolini and Simpson [27] have shown how to construct parametric domain-theoretic models using synthetic domain-theory in intuitionistic set-theory. Moreover, they have shown how to give a computationally adequate denotational semantics of Lily.

In the present paper we make the following contributions to the study of parametric domain-theoretic models of intuitionistic / linear lambda calculus:

- We present a formalization of Linear Abadi-Plotkin Logic with fixed points (LAPL). The term language, called PILL_Y for polymorphic intuitionistic / linear logic, is a simple extension of Barber and Plotkin's calculus for dual intuitionistic / linear lambda calculus (DILL) with polymorphism and fixed points and the logic is an extension of Abadi-Plotkin's logic for parametricity with rules for forming admissible relations. The logic allows for intuitionistic reasoning over PILL_Y terms; *i.e.*, the terms can be linear but the reasoning about terms is always done intuitionistically.

- We give detailed proofs in LAPL of consequences of parametricity, including the solution of recursive domain equations; these results and proofs have not been presented formally in the literature before.

- We give a definition of a *parametric LAPL-structure*, which is a categorical notion of a parametric model of LAPL, with associated soundness and completeness theorems.

- We show how to solve recursive domain equations in parametric LAPL-structures by a simple use of the internal language and the earlier proofs in LAPL.

- We present a detailed definition of a concrete parametric LAPL-structure based on suitable categories of partial equivalence relations over a universal model of the untyped lambda calculus, thus confirming the folklore idea that one should be able to get a parametric domain-theoretic model using partial equivalence relations over a universal model of the untyped lambda calculus.

We remark that one can see our notion of parametric LAPL-structure as a suitable categorical axiomatization of a good category of domains. In Axiomatic Domain Theory much of the earlier work has focused on axiomatizing the adjunction between the category of predomains and continuous functions and the category of predomains and partial continuous functions [6, Page 7] – here we axiomatize the adjunction between the category of domains and strict functions and the category of domains and all continuous functions and extend it with parametric polymorphism, which then suffices to also model recursive types.

In the technical development, we make use of a notion of admissible relations, which we axiomatize, since admissible may mean different things in different models. We believe our axiomatization is reasonable in that it accommodates several different kinds of models, such as the classical one described here and models based on synthetic domain theory [18].

The work presented here builds upon our previous work on categorical models of Abadi-Plotkin's logic for parametricity [5], which includes detailed proofs of consequences of parametricity for polymorphic lamdba calculus and also includes a description of a parametric completion process that given an internal model of polymorphic lambda calculus produces a parametric model. It is not necessary to be familiar with the details of [5] to read the present paper (except for Appendix A of [5], which contains some definitions and theory concerning composable fibrations), but, for readers unfamiliar with parametricity, it may be helpful to start with [5], since the proofs of consequences of parametricity given here are slightly more sophisticated than the ones in [5] because of the use of linearity.

In subsequent papers we intend to show how one can define a computationally adequate model of Lily and how to produce parametric LAPL-structures from Rosolini and Simpson's models based on intuitionistic set theory [27] (this has been worked out at the time of writing [18]) and from Pitts and coworkers operational models [4] (we conjecture that this is possible, but have not checked all the details at the time of writing). As a corollary one then has that the encodings of recursive types mentioned in [27] and [4] really do work out (these properties were not formally proved in *loc. cit.*). We will also extend the parametric completion process of [5] to produce a parametric LAPL-structure given a model of polymorphic intuitionistic / linear lambda calculus, see [16].

1.1 Outline

The remainder of this paper is organized as follows. In Section 2 we present LAPL, the logic for reasoning about parametricity over polymorphic intuitionistic / linear lambda calculus (PILL$_Y$). In Section 3 we give detailed proofs of many consequences of parametricity, including initial algebras and final coalgebras for definable functors and recursive types of mixed variance. In Section 4 we present our definition of an LAPL-structure, and we prove soundness and completeness with respect to LAPL in Sections 4.1 and 4.2, respectively. The definition of LAPL-structure builds upon fibred versions of models of intuitionistic / linear logic [3, 14]. In our presentation we assume that the reader is familiar with models of intuitionistic / linear logic.[1] In Section 5 we present our definition of a *parametric* LAPL-structure and prove that one may solve recursive domains equations in such. In Section 6 we present a concrete parametric LAPL-structure based on partial equivalence relations over a universal domain model. To make it easier to understand the model, we first present a model of PILL$_Y$ (without parametricity) and then show how to make it into a parametric LAPL-structure. We also include an example of calculations in the concrete model.

2 Linear Abadi-Plotkin Logic

In this section we define a logic for reasoning about parametricity for Polymorphic Intuitionistic Linear Lambda calculus with fixed points (PILL$_Y$). The logic is based on Abadi and Plotkin's logic for parametricity [23] for the second-order lambda calculus and thus we refer to the logic as Linear Abadi-Plotkin Logic (LAPL).

The logic for parametricity is basically a higher-order logic over PILL$_Y$. Expressions of the logic are formulas in contexts of variables of PILL$_Y$ and relations among types of PILL$_Y$. Thus we start by defining PILL$_Y$.

2.1 PILL$_Y$

PILL$_Y$ is essentially Barber and Plotkin's DILL [2] extended with polymorphism and a fixed point combinator.

Well-formed type expressions in PILL$_Y$ are expressions of the form:

$$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type} \vdash \sigma \colon \mathsf{Type}$$

where σ is built using the syntax

$$\sigma ::= \alpha \mid I \mid \sigma \otimes \tau \mid \sigma \multimap \tau \mid !\sigma \mid \prod \alpha.\, \sigma.$$

and all the free variables of sigma appear on the left hand side of the turnstile. The last construction binds α, so if we have a type

$$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type} \vdash \sigma \colon \mathsf{Type},$$

then we may form the type

$$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_{i-1}\mathsf{Type}, \alpha_{i+1}\mathsf{Type} \ldots \alpha_n \colon \mathsf{Type} \vdash \prod \alpha_i.\, \sigma \colon \mathsf{Type}.$$

[1] To aid readers unfamiliar with these matters, we have written a short technical note containing detailed definitions and propositions needed here [17].

We use σ, τ, ω, σ', τ'... to range over types. The list of α's is called the kind context, and is often denoted simply by Ξ or $\vec{\alpha}$. Since there is only one kind this annotation is often omitted.

The terms of PILL$_Y$ are of the form:

$$\Xi \mid x_1 \colon \sigma_1, \ldots, x_n \colon \sigma_n; x'_1 \colon \sigma'_1, \ldots, x'_m \colon \sigma'_m \vdash t \colon \tau$$

where the σ_i, σ'_i, and τ are well-formed types in the kind context Ξ. The list of x's is called the intuitionistic type context and is often denoted Γ, and the list of x''s is called the linear type context, often denoted Δ. No repetition of variable names is allowed in any of the contexts, but permutation akin to having an exchange rule is. Note, that due to the nature of the axioms of the to-be-introduced formation rules, weakening and contraction can be derived for all but the linear context.

The grammar for terms is:

$$\begin{gathered} t ::= x \mid \star \mid Y \mid \lambda^{\circ} x \colon \sigma . t \mid t\, t \mid t \otimes t \mid !t \mid \Lambda \alpha \colon \mathsf{Type}.\, t \mid t(\sigma) \mid \\ \text{let } x \colon \sigma \otimes y \colon \tau \text{ be } t \text{ in } t \mid \text{let } !x \colon \sigma \text{ be } t \text{ in } t \mid \text{let } \star \text{ be } t \text{ in } t \end{gathered}$$

We use λ°, which bear some graphical resemblance to $\multimap$, to denote linear function abstraction. And we use s, t, u... to range over terms.

The formation rules are given in Figure 1. $\Xi \mid \Gamma; \Delta$ is considered well-formed if for all types σ appearing in Γ and Δ, $\Xi \vdash \sigma \colon \mathsf{Type}$ is a well-formed type construction. Δ and Δ' are considered disjoint if the set of variables appearing in Δ is disjoint from the set of variables appearing in Δ'. We use $-$ to denote an empty context. As the types of variables in the let-constructions and function abstractions are often apparent from the context, these will just as often be omitted. What we have described above is called *pure* PILL$_Y$. In general we will consider PILL$_Y$ over polymorphic signatures [12, 8.1.1]. Informally, one may think of such a calculus as pure PILL$_Y$ with added type-constants and term-constants. For instance, one may have a constant type for integers or a constant type for lists $\alpha \vdash \mathit{lists}(\alpha) \colon \mathsf{Type}$. We will be particularly interested in the internal language of a PILL$_Y$ model (see Section 4), which in general will be a non-pure calculus.

We will also sometimes speak of the calculus PILL. This is PILL$_Y$ without the fixed point combinator Y.

2.1.1 Equality

The *external equality* relation on PILL$_Y$ terms is the least equivalence relation given by the rules in Figure 2. The definition makes use of the notion of a *context*, which, loosely speaking, is a term with exactly one hole in it. Formally contexts are defined using the grammar:

$$\begin{array}{rcl} C[-] & ::= & - \mid \text{let } \star \text{ be } C[-] \text{ in } t \mid \text{let } \star \text{ be } t \text{ in } C[-] \mid t \otimes C[-] \mid C[-] \otimes t \mid \\ & & \text{let } x \otimes y \text{ be } C[-] \text{ in } t \mid \text{let } x \otimes y \text{ be } t \text{ in } C[-] \mid \lambda^{\circ} x \colon \sigma .\, C[-] \mid \\ & & C[-]\, t \mid t\, C[-] \mid !C[-] \mid \text{let } !x \text{ be } C[-] \text{ in } t \mid \text{let } !x \text{ be } t \text{ in } C[-] \mid \\ & & \Lambda \alpha \colon \mathsf{Type}.\, C[-] \mid C[-]\sigma \end{array}$$

A $\Xi \mid \Gamma; \Delta \vdash \sigma$ — $\Xi \mid \Gamma'; \Delta' \vdash \tau$ context is a context $C[-]$ such that for any well-formed term $\Xi \mid \Gamma; \Delta \vdash t \colon \sigma$, the term $\Xi \mid \Gamma'; \Delta' \vdash C[t] \colon \tau$ is well-formed. A context is **linear**, if it does not contain a subcontext of the form $!C[-]$.

We prove a couple of practical lemmas about external equality.

Lemma 2.1. *Suppose* $\Xi \mid \Gamma; \Delta \vdash f, g \colon !\sigma \multimap \tau$ *are terms such that*

$$\Xi \mid \Gamma, x \colon \sigma; \Delta \vdash f(!x) = g(!x).$$

Then $f = g$.

$$\frac{}{\Xi \mid \Gamma; - \vdash \star \colon I}$$

$$\frac{}{\Xi \mid \Gamma; - \vdash Y \colon \prod \alpha.\, !(!\alpha \multimap \alpha) \multimap \alpha}$$

$$\frac{}{\Xi \mid \Gamma, x \colon \sigma; - \vdash x \colon \sigma}$$

$$\frac{}{\Xi \mid \Gamma; x \colon \sigma \vdash x \colon \sigma}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t \colon \sigma \multimap \tau \quad \Xi \mid \Gamma; \Delta' \vdash u \colon \sigma}{\Xi \mid \Gamma; \Delta, \Delta' \vdash t\, u \colon \tau}\ \Delta, \Delta' \text{ disjoint}$$

$$\frac{\Xi \mid \Gamma; \Delta, x \colon \sigma \vdash u \colon \tau}{\Xi \mid \Gamma; \Delta \vdash \lambda^\circ x \colon \sigma.\, u \colon \sigma \multimap \tau}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t \colon \sigma \quad \Xi \mid \Gamma; \Delta' \vdash s \colon \tau}{\Xi \mid \Gamma; \Delta, \Delta' \vdash t \otimes s \colon \sigma \otimes \tau}\ \Delta, \Delta' \text{ disjoint}$$

$$\frac{\Xi \mid \Gamma; - \vdash t \colon \sigma}{\Xi \mid \Gamma; - \vdash !t \colon \sigma}$$

$$\frac{\Xi, \alpha \colon \mathsf{Type} \mid \Gamma; \Delta \vdash t \colon \sigma}{\Xi \mid \Gamma; \Delta \vdash \Lambda\alpha \colon \mathsf{Type}.\, t \colon \prod \alpha \colon \mathsf{Type}.\, \sigma}\ \Xi \mid \Gamma; \Delta \text{ is well-formed}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t \colon \prod \alpha \colon \mathsf{Type}.\, \sigma \qquad \Xi \vdash \tau \colon \mathsf{Type}}{\Xi \mid \Gamma; \Delta \vdash t(\tau) \colon \sigma[\tau/\alpha]}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash s \colon \sigma \otimes \sigma' \qquad \Xi \mid \Gamma; \Delta', x \colon \sigma, y \colon \sigma' \vdash t \colon \tau}{\Xi \mid \Gamma; \Delta, \Delta' \vdash \text{let } x \colon \sigma \otimes y \colon \sigma' \text{ be } s \text{ in } t \colon \tau}\ \Delta, \Delta' \text{ disjoint}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash s \colon !\sigma \qquad \Xi \mid \Gamma, x \colon \sigma; \Delta' \vdash t \colon \tau}{\Xi \mid \Gamma; \Delta, \Delta' \vdash \text{let } !x \colon !\sigma \text{ be } s \text{ in } t \colon \tau}\ \Delta, \Delta' \text{ disjoint}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t \colon I \quad \Xi \mid \Gamma; \Delta' \vdash s \colon \sigma}{\Xi \mid \Gamma; \Delta, \Delta' \vdash \text{let } \star \text{ be } t \text{ in } s \colon \sigma}$$

Figure 1: Formation rules for terms

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash (\lambda^{\circ} x\colon \sigma.\, t)u = t[u/x]}\ \beta\text{-term}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash (\Lambda \alpha\colon \mathsf{Type}.\, t)\sigma = t[\sigma/\alpha]}\ \beta\text{-type}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \lambda^{\circ} x\colon \sigma.\, (tx) = t}\ \eta\text{-term}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \Lambda \alpha\colon \mathsf{Type}.\, (t\alpha) = t}\ \eta\text{-type}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } \star \text{ in } t = t}\ \beta - \star$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } t \text{ in } \star = t}\ \eta - \star$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } s \otimes u \text{ in } t = t[s, u/x, y]}\ \beta - \otimes$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } t \text{ in } x \otimes y = t}\ \eta - \otimes$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x\colon \sigma \text{ be } !u \text{ in } t = t[u/x]}\ \beta - !$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x\colon \sigma \text{ be } t \text{ in } !x = t}\ \eta - !$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t = s\colon \sigma \quad C[-] \text{ is a } \Xi \mid \Gamma; \Delta \vdash \sigma - \Xi \mid \Gamma'; \Delta' \vdash \tau \text{ context}}{\Xi \mid \Gamma'; \Delta' \vdash C[t] = C[s]\colon \tau}$$

$$\frac{C[-] \text{ is a linear context}}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } t \text{ in } C[u] = C[\text{let } \star \text{ be } t \text{ in } u]}$$

$$\frac{C[-] \text{ is a linear context and does not bind } x, y \text{ or contain them free}}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } t \text{ in } C[u] = C[\text{let } x \otimes y \text{ be } t \text{ in } u]}$$

$$\frac{C[-] \text{ is linear and does not bind } x \text{ or contain it free}}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x \text{ be } t \text{ in } C[u] = C[\text{let } !x \text{ be } t \text{ in } u]}$$

$$\frac{\Xi \mid \Gamma; - \vdash f\colon !\sigma \multimap \sigma}{\Xi \mid \Gamma; - \vdash f\ !(Y\ \sigma\ (!f)) = Y\ \sigma\ (!f)}$$

Figure 2: Rules for external equality

$$\Xi\colon \mathsf{Ctx} \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma;\Delta\colon \mathsf{Ctx}$$

$$\Xi \mid \Gamma \mid \Theta\colon \mathsf{Ctx} \qquad \Xi \mid \Gamma;\Delta \vdash t\colon \sigma \qquad \Xi \mid \Gamma;\Delta \vdash t = u$$

$$\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau)$$

$$\Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop} \qquad \Xi \mid \Gamma \mid \Theta \mid \phi_1,\ldots,\phi_n \vdash \psi$$

Figure 3: Types of judgements

Proof. Using the rules for external equality, we conclude from the assumption that

$$\Xi \mid \Gamma;\Delta, y\colon !\sigma \vdash \text{let } !x \text{ be } y \text{ in } f(!x) = \text{let } !x \text{ be } y \text{ in } g(!x)$$

and further that

$$\Xi \mid \Gamma;\Delta, y\colon !\sigma \vdash f(\text{let } !x \text{ be } y \text{ in } !x) = g(\text{let } !x \text{ be } y \text{ in } !x).$$

Thus

$$\Xi \mid \Gamma;\Delta, y\colon !\sigma \vdash f(y) = g(y),$$

and hence $f = \lambda^\circ y\colon !\sigma.\, f(y) = \lambda^\circ y\colon !\sigma.\, g(y) = g$. □

2.1.2 Ordinary lambda abstraction

We encode ordinary lambda abstraction in the usual way by defining

$$\sigma \to \tau = !\sigma \multimap \tau$$

and

$$\lambda x\colon \sigma.\, t = \lambda^\circ y\colon !\sigma.\, \text{let } !x \text{ be } y \text{ in } t$$

where y is a fresh variable. This gives us the rule

$$\frac{\Xi \mid \Gamma, x\colon \sigma;\Delta \vdash t\colon \tau}{\Xi \mid \Gamma;\Delta \vdash \lambda x\colon \sigma.\, t\colon \sigma \to \tau}$$

For evaluation we have the rule

$$\frac{\Xi \mid \Gamma;- \vdash t\colon \sigma \quad \Xi \mid \Gamma;\Delta \vdash f\colon \sigma \to \tau}{\Xi \mid \Gamma;\Delta \vdash f\ !t\colon \tau}$$

and the equality rules give

$$(\lambda x\colon \sigma.\, t)\ !s = t[s/x].$$

Note that using this notation the constant Y can obtain the more familiar looking type

$$Y\colon \Pi\alpha.\, (\alpha \to \alpha) \to \alpha$$

2.2 The logic

As mentioned, expressions of LAPL live in contexts of variables of PILL_Y and relations among types of PILL_Y. The contexts look like this:

$$\Xi \mid \Gamma \mid R_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, R_n\colon \mathsf{Rel}(\tau_n, \tau_n'), S_1\colon \mathsf{AdmRel}(\omega_1, \omega_1'), \ldots, S_m\colon \mathsf{AdmRel}(\omega_m, \omega_m')$$

where $\Xi \mid \Gamma; -$ is a context of PILL_Y and the $\tau_i, \tau_i', \omega_i, \omega_i'$ are well-formed types in context Ξ, for all i. The list of R's and S's is called the relational context and is often denoted Θ. As for the other contexts we do not allow repetition, but permutation of variables. The R's and the S's are interchangeable.

The concept of admissible relations is taken from domain theory. Intuitively admissible relations relate $\perp$ to $\perp$ and are chain complete.

It is important to note that there is no linear component Δ in the contexts — the point is that the logic only allows for *intuitionistic* (no linearity) reasoning about terms of PILL_Y, whereas PILL_Y terms can behave linearly.

Propositions in the logic are given by the syntax:

$$\begin{aligned}\phi \quad ::= \quad & (t =_\sigma u) \mid \rho(t, u) \mid \ \phi \supset \psi \mid \perp \mid \top \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall \alpha\colon \mathsf{Type}.\, \phi \mid \\ & \forall x\colon \sigma.\, \phi \mid \forall R\colon \mathsf{Rel}(\sigma, \tau).\, \phi \mid \forall S\colon \mathsf{AdmRel}(\sigma, \tau).\, \phi \mid \\ & \exists \alpha\colon \mathsf{Type}.\, \phi \mid \exists x\colon \sigma.\, \phi \mid \exists R\colon \mathsf{Rel}(\sigma, \tau).\, \phi \mid \exists S\colon \mathsf{AdmRel}(\sigma, \tau).\, \phi\end{aligned}$$

where ρ is a definable relation (to be defined below). The judgements of the logic are presented in figure 3. In the following we give formation rules for the above.

Remark 2.2. Our Linear Abadi & Plotkin logic is designed for reasoning about binary relational parametricity. For reasoning about other arities of parametricity, one can easily replace binary relations in the logic by relations of other arities. In the case of unary parametricity, for example, one would then have an interpretation of types as predicates. See also [28, 29]

We first have the formation rule for internal equality:

$$\frac{\Xi \mid \Gamma; - \vdash t\colon \sigma \qquad \Xi \mid \Gamma; - \vdash u\colon \sigma}{\Xi \mid \Gamma \mid \Theta \vdash t =_\sigma u\colon \mathsf{Prop}}$$

Notice here the notational difference between $t = u$ and $t =_\sigma u$. The former denotes *external* equality and the latter is a proposition in the logic. The rules for $\supset$, $\vee$ and $\wedge$ are the usual ones, where $\supset$ denotes implication. $\top$, $\perp$ are propositions in any context. We use $\supset\subset$ for biimplication.

We have the following formation rules for universal quantification:

$$\frac{\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall x\colon \sigma.\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau) \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall R\colon \mathsf{Rel}(\sigma, \tau).\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\sigma, \tau) \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall S\colon \mathsf{AdmRel}(\sigma, \tau).\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi, \alpha \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall \alpha\colon \mathsf{Type}.\, \phi\colon \mathsf{Prop}}\ \Xi \mid \Gamma \mid \Theta \text{ is well-formed}$$

The side condition $\Xi \mid \Gamma \mid \Theta$ is well-formed means that all the types of variables in Γ and of relation variables in Θ are well-formed in Ξ (i.e., all the free type variables of the types occur in Ξ).

There are similar formation rules for the existential quantifier.

Before we give the formation rule for $\rho(t, u)$, we discuss definable relations.

2.2.1 Definable relations

Definable relations are given by the grammar:

$$\rho ::= R \mid (x\colon \sigma, y\colon \tau).\phi$$

Definable relations always have a domain and a codomain, just as terms always have types. The basic formation rules for definable relations are:

$$\frac{}{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \vdash R\colon \mathsf{Rel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\,\phi\colon \mathsf{Rel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau)}{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)}$$

Notice that in the second rule we can only abstract *intuitionistic* variables to obtain definable relations. In the last rule, $\rho\colon \mathsf{AdmRel}(\sigma,\tau)$ is an admissible relation, to be discussed below. The rule says that the admissible relations constitute a subset of the definable relations.

An example of a definable relation is the graph relation of a function:

$$\langle f \rangle = (x\colon \sigma, y\colon \tau).\, fx =_\tau y,$$

for $f\colon \sigma \multimap \tau$. The equality relation eq_σ is defined as the graph of the identity map.

If $\rho\colon \mathsf{Rel}(\sigma,\tau)$ is a definable relation, and we are given terms of the right types, then we may form the proposition stating that the two terms are related by the definable relation:

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau) \quad \Xi \mid \Gamma; - \vdash t\colon \sigma, s\colon \tau}{\Xi \mid \Gamma \mid \Theta \vdash \rho(t,s)\colon \mathsf{Prop}} \tag{1}$$

We shall also write $t\rho s$ for $\rho(t,s)$.

We introduce some shorthand notation for reindexing of relations. For $f\colon \sigma' \multimap \sigma, g\colon \tau' \multimap \tau$ and $\rho\colon \mathsf{Rel}(\sigma,\tau)$, we write $(f,g)^*\rho$ for the definable relation

$$(x\colon \sigma', y\colon \tau').\, \rho(f\,x, g\,y).$$

2.2.2 Constructions on definable relations

In this subsection we present some constructions on definable relations, which will be used to give a relational interpretation of the types of PILL_Y.

If $\rho\colon \mathsf{Rel}(\sigma,\tau)$ and $\rho'\colon \mathsf{Rel}(\sigma',\tau')$, then we may construct a definable relation

$$(\rho \multimap \rho')\colon \mathsf{Rel}((\sigma \multimap \sigma'),(\tau \multimap \tau')),$$

defined by

$$\rho \multimap \rho' = (f\colon \sigma \multimap \sigma', g\colon \tau \multimap \tau').\, \forall x\colon \sigma.\, \forall y\colon \tau.\, \rho(x,y) \supset \rho'(fx, gy).$$

If

$$\Xi, \alpha, \beta \mid \Gamma \mid \ \Theta, R\colon \mathsf{AdmRel}(\alpha,\beta) \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)$$

is well-formed and $\Xi \mid \Gamma \mid \Theta$ is well-formed, $\Xi, \alpha \vdash \sigma\colon \mathsf{Type}$, and $\Xi, \beta \vdash \tau\colon \mathsf{Type}$ we may define

$$\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha,\beta)).\, \rho\colon \mathsf{Rel}((\textstyle\prod \alpha\colon \mathsf{Type}.\, \sigma), (\prod \beta\colon \mathsf{Type}.\, \tau))$$

as

$$\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha,\beta)).\, \rho = \\ (t : \textstyle\prod \alpha\colon \mathsf{Type}.\, \sigma, u : \prod \beta\colon \mathsf{Type}.\, \tau).\, \forall \alpha, \beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{AdmRel}(\alpha,\beta).\, \rho(t\alpha, u\beta).$$

For $\rho\colon \mathsf{Rel}(\sigma,\tau)$, we seek to define a relation $!\rho\colon \mathsf{Rel}(!\sigma, !\tau)$. First we define for any type σ the proposition $(-)\downarrow$ on σ as

$$x \downarrow \equiv \exists f\colon \sigma \multimap I.\, f(x) =_I \star.$$

The intuition here is that types are pointed, and $x\downarrow$ is thought of as $x \neq \bot$. Since we have also fixed points, we may think of types as domains.

We further define the map $\epsilon\colon !\sigma \multimap \sigma$ as $\lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } y = \lambda x\colon \sigma.\, x$. We can now define

$$!\rho = (x\colon\ !\sigma, y\colon !\tau).\, x \downarrow \supset\subset y \downarrow \wedge (x \downarrow \supset \rho(\epsilon x, \epsilon y)).$$

Following the intuition of domains, ! is to be thought of as lifting, and ϵ the unit providing the unlifted version of an element. The intuitive reading of $!\rho$ is, that $\bot$ is related to $\bot$ (represented by the fact, that x is related to y if neither $x\downarrow$ nor $y\downarrow$) and that two !'ed elements are related if their un-!-ed versions are.

Next we define the tensor product of ρ and ρ'

$$\rho \otimes \rho'\colon \mathsf{Rel}((\sigma \otimes \sigma'),(\tau \otimes \tau')),$$

for $\rho\colon \mathsf{Rel}(\sigma,\tau)$ and $\rho'\colon \mathsf{Rel}(\sigma',\tau')$. The basic requirement for this definition is that $\otimes$ should become a left adjoint to $\multimap$ in the category of relations $\mathbf{LinAdmRelations}$ to be defined in Section 4. To give a concrete definition satisfying this requirement, we take a slightly long route. We first introduce the map

$$f : \sigma \otimes \tau \multimap \textstyle\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

defined as

$$f\,x = \text{let } x' \otimes x''\colon \sigma \otimes \tau \text{ be } x \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\, x'\, x''.$$

Then we define

$$\rho \otimes \rho' = (f, f)^*(\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha,\beta)).\, (\rho \multimap \rho' \multimap R) \multimap R),$$

or, if we write it out,

$$\begin{aligned} \rho \otimes \rho' \quad = \quad & (x\colon \sigma \otimes \sigma', y\colon \tau \otimes \tau').\, \forall \alpha, \beta, R\colon \mathsf{AdmRel}(\alpha,\beta). \\ & \forall t\colon \sigma \multimap \tau \multimap \alpha, t'\colon \sigma' \multimap \tau' \multimap \beta.\, (\rho \multimap \rho' \multimap R)(t,t') \supset \\ & R(\text{let } x' \otimes x'' \text{ be } x \text{ in } t\, x'\, x'', \text{let } y' \otimes y'' \text{ be } y \text{ in } t'\, y'\, y''). \end{aligned}$$

The reason for this at first sight fairly convoluted definition, is that we will later prove, using parametricity, that $\sigma \otimes \tau$ is isomorphic to $\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$, and we already have a relational interpretation of the latter. The idea of using this definition of $\otimes$ is due to Alex Simpson. We use the same trick to define a relation on I:

Following the same strategy as before, we define a relation $I_{Rel} \colon \mathsf{AdmRel}(I, I)$ using the map

$$f \colon I \multimap \prod \alpha.\, \alpha \multimap \alpha$$

defined as $\lambda^\circ x \colon I.\,\text{let } \star \text{ be } x \text{ in } id$, where $id = \Lambda\alpha.\, \lambda^\circ x \colon \alpha.\, x$ and define

$$I_{Rel} = (f, f)^*(\forall(\alpha, \beta, R \colon \mathsf{AdmRel}(\alpha, \beta)).\, R \multimap R),$$

which, if we write it out, is

$$(x \colon I, y \colon I).\, \forall(\alpha, \beta, R \colon \mathsf{AdmRel}(\alpha, \beta)).\, \forall z \colon \alpha, w \colon \beta.\, zRw \supset (\text{let } \star \text{ be } x \text{ in } z) R (\text{let } \star \text{ be } y \text{ in } w).$$

2.2.3 Admissible relations

The relational interpretation of a type with n free variables is a function taking n relations and returning a new relation. However, we will not require that this function is defined on all vectors of relations, but only that it is defined on vectors of “admissible relations”. On the other hand this function should also return admissible relations. Since “admissible” might mean different things in different settings, we axiomatize the concept of admissible relations.

The axioms for admissible relations are formulated in Figure 4. In the last of these rules $\rho \equiv \rho'$ is a shorthand for $\forall x, y.\, \rho(x, y) \supset\subset \rho'(x, y)$.

Proposition 2.3. *The class of admissible relations contains all graphs and is closed under the constructions of Section 2.2.2.*

Proof. Graph relations are admissible since equality relations are and admissible relations are closed under reindexing. For the constructions of Section 2.2.2, we just give the proof of $\multimap$.

We must prove that for ρ, ρ' admissible relations $\rho \multimap \rho'$ is admissible

$$\dfrac{\dfrac{\dfrac{\Xi \mid \Gamma \mid \Theta \vdash \rho' \colon \mathsf{AdmRel}(\sigma', \tau')}{\Xi \mid \Gamma, x, y \mid \Theta \vdash (f, g).\, \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}(\sigma \multimap \sigma', \tau \multimap \tau')} \qquad \dfrac{\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{AdmRel}(\sigma, \tau)}{\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \rho(x, y) \colon \mathsf{Prop}}}{\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash (f \colon \sigma \multimap \sigma', g \colon \tau \multimap \tau').\, \rho(x, y) \supset \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}((\sigma \multimap \sigma'), (\tau \multimap \tau'))}}{\Xi \mid \Gamma \mid \Theta \vdash (f \colon \sigma \multimap \sigma', g \colon \tau \multimap \tau').\, \forall x \colon \sigma, y \colon \tau.\, \rho(x, y) \supset \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}((\sigma \multimap \sigma'), (\tau \multimap \tau'))}$$

where in the top deduction on the left, we have reindexed ρ' along the evaluation maps

$$\lambda^\circ f \colon \sigma \multimap \sigma'.\, f\,x \qquad \lambda^\circ g \colon \tau \multimap \tau'.\, g\,y.$$

□

Now, finally, we may give the last formation rule for definable relations:

$$\dfrac{\alpha_1, \ldots, \alpha_n \vdash \sigma(\vec{\alpha}) \colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho_1 \colon \mathsf{AdmRel}(\tau_1, \tau_1'), \ldots, \rho_n \colon \mathsf{AdmRel}(\tau_n, \tau_n')}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}] \colon \mathsf{AdmRel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))}$$

$$\frac{}{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{AdmRel}(\sigma,\tau) \vdash R\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{}{\Xi \mid \Gamma \mid \Theta \vdash \mathit{eq}_\sigma\colon \mathsf{AdmRel}(\sigma,\sigma)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma; - \vdash t\colon \sigma' \multimap \sigma, u\colon \tau' \multimap \tau \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma', y\colon \tau').\, \rho(t\,x, u\,y)\colon \mathsf{AdmRel}(\sigma',\tau')}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho, \rho'\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \rho(x,y) \wedge \rho'(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \tau, y\colon \sigma).\, \rho(y,x)\colon \mathsf{AdmRel}(\tau,\sigma)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau)}{\Xi \mid \Gamma \mid \Theta \vdash !\rho\colon \mathsf{AdmRel}(!\sigma,!\tau)}$$

$$\frac{x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \top\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop} \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi \supset \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi, \alpha \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \vdash \tau\colon \mathsf{Type} \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall \alpha\colon \mathsf{Type}.\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma, z\colon \omega \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall z\colon \omega.\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{AdmRel}(\omega,\omega') \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall R\colon \mathsf{AdmRel}(\omega,\omega').\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\omega,\omega') \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall R\colon \mathsf{Rel}(\omega,\omega').\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau), \rho'\colon \mathsf{Rel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \mid \top \vdash \rho \equiv \rho'}{\Xi \mid \Gamma \mid \Theta \vdash \rho'\colon \mathsf{AdmRel}(\sigma,\tau)}$$

Figure 4: Rules for admissible relations

Observe that $\sigma[\vec{\rho}]$ is a syntactic construction and is not obtained by substitution as in [23]. Still the notation $\sigma[\rho_1/\alpha_1, \ldots, \rho_n/\alpha_n]$ might be more complete, but this quickly becomes overly verbose. In [23] $\sigma[\vec{\rho}]$ is to some extent defined inductively on the structure of σ, but in our case that is not enough, since we will need to form $\sigma[\vec{\rho}]$ for type constants (when using the internal language of a model of LAPL). We call $\sigma[\vec{\rho}]$ the *relational interpretation of the type* σ.

2.2.4 Axioms and Rules

The last judgement in figure 3 has not yet been mentioned. It says that in the given context, the formulas $\phi_1, \ldots, \phi_n$ collectively imply ψ. We will often write Φ for $\phi_1, \ldots, \phi_n$.

Having specified the language of LAPL, it is time to specify the axioms and inference rules. We have all the usual axioms and rules of predicate logic plus the axioms and rules specified below.

Rules for substitution:

Rule 2.4.
$$\frac{\Xi \mid \Gamma, x \colon \sigma \mid \Theta \mid \top \vdash \phi \quad \Xi \mid \Gamma \vdash t \colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[t/x]}$$

Rule 2.5.
$$\frac{\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\sigma, \tau) \mid \top \vdash \phi \quad \Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[\rho/R]}$$

Rule 2.6.
$$\frac{\Xi \mid \Gamma \mid \Theta, S \colon \mathsf{AdmRel}(\sigma, \tau) \mid \top \vdash \phi \quad \Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{AdmRel}(\sigma, \tau)}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[\rho/S]}$$

Rule 2.7.
$$\frac{\Xi, \alpha \mid \Gamma \mid \Theta \mid \top \vdash \phi \quad \Xi \vdash \sigma \colon \mathsf{Type}}{\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \mid \top \vdash \phi[\sigma/\alpha]}$$

The *substitution* axiom:

Axiom 2.8. $\forall \alpha, \beta \colon \mathsf{Type}. \forall x, x' \colon \alpha. \forall y, y' \colon \beta. \forall R \colon \mathsf{Rel}(\alpha, \beta.) R(x, y) \wedge$
$x =_\alpha x' \wedge y =_\beta y' \supset R(x', y')$

Rules for $\forall$-quantification:

Rule 2.9.
$$\begin{array}{c} \Xi, \alpha \mid \Gamma \mid \Theta \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall \alpha \colon \mathsf{Type}. \psi \end{array} \; \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

Rule 2.10.
$$\begin{array}{c} \Xi \mid \Gamma, x \colon \sigma \mid \Theta \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall x \colon \sigma. \psi \end{array} \; \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

Rule 2.11.
$$\begin{array}{c} \Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\tau, \tau') \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall R \colon \mathsf{Rel}(\tau, \tau'). \psi \end{array} \; \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

Rule 2.12.
$$\begin{array}{c} \Xi \mid \Gamma \mid \Theta, S \colon \mathsf{AdmRel}(\tau, \tau') \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall S \colon \mathsf{AdmRel}(\tau, \tau'). \psi \end{array} \; \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

Rules for $\exists$-quantification:

Rule 2.13.
$$\begin{array}{c} \Xi, \alpha \mid \Gamma \mid \Theta \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists \alpha \colon \mathsf{Type}. \phi \vdash \psi \end{array} \; \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.14.}\ \begin{array}{c} \Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists x\colon \sigma.\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.15.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau, \tau') \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists R\colon \mathsf{Rel}(\tau, \tau'.)\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.16.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\tau, \tau') \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists S\colon \mathsf{AdmRel}(\tau, \tau').\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

External equality implies internal equality:

$$\textbf{Rule 2.17.}\ \frac{\Xi \mid \Gamma; - \vdash t = u\colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash t =_\sigma u}$$

There are also obvious rules expressing that internal equality is an equivalence relation.

Intuitively admissible relations should relate $\bot$ to $\bot$ and we need an axiom stating this. In general, we will use $(-)\downarrow$ as the test for $x \neq \bot$.

$$\textbf{Rule 2.18.}\ \frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(!\sigma, !\tau), \rho'\colon \mathsf{AdmRel}(!\sigma, !\tau) \quad x, y \notin \Gamma}{\begin{array}{l} \Xi \mid \Gamma \mid \Theta \mid \ \forall x\colon \sigma, y\colon \tau.\, \rho(!x, !y) \supset \rho'(!x, !y) \vdash \\ \qquad\qquad\qquad \forall x\colon !\sigma, y\colon !\tau.\, x \downarrow \supset\subset y \downarrow \supset (\rho(x, y) \supset \rho'(x, y)) \end{array}}$$

We have rules concerning the interpretation of types as relations:

$$\textbf{Rule 2.19.}\ \frac{\vec{\alpha} \vdash \alpha_i\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \alpha_i[\vec{\rho}] \equiv \rho_i}$$

$$\textbf{Rule 2.20.}\ \frac{\vec{\alpha} \vdash \sigma \multimap \sigma'\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \multimap \sigma')[\vec{\rho}] \equiv (\sigma[\vec{\rho}] \multimap \sigma'[\vec{\rho}])}$$

$$\textbf{Rule 2.21.}\ \frac{\vec{\alpha} \vdash \sigma \otimes \sigma'\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \otimes \sigma')[\vec{\rho}] \equiv (\sigma[\vec{\rho}] \otimes \sigma'[\vec{\rho}])}$$

$$\textbf{Rule 2.22.}\ \frac{\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash I[\vec{\rho}] \equiv I_{Rel}}$$

$$\textbf{Rule 2.23.}\ \frac{\vec{\alpha} \vdash \prod \beta.\, \sigma(\vec{\alpha}, \beta)\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \top \vdash (\prod \beta.\, \sigma(\vec{\alpha}, \beta))[\vec{\rho}] \equiv \forall(\beta, \beta', R\colon \mathsf{AdmRel}(\beta, \beta')).\, \sigma[\vec{\rho}, R])}$$

$$\textbf{Rule 2.24.}\ \frac{\vec{\alpha} \vdash !\sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (!\sigma)[\vec{\rho}] \equiv !(\sigma[\rho])}$$

Here $\rho \equiv \rho'$ is shorthand for $\forall x, y.\, x\rho y \supset\subset x\rho' y$.

If the definable relation ρ is of the form $(x\colon \sigma, y\colon \tau).\, \phi(x, y)$, then $\rho(t, u)$ should be equivalent to ϕ with x, y substituted by t, u:

$$\textbf{Rule 2.25.}\ \frac{\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi\colon \mathsf{Prop} \quad \Xi \mid \Gamma; - \vdash t\colon \sigma, u\colon \tau}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(t, u) \supset\subset \phi[t, u/x, y]}$$

Axiom 2.26. $$\frac{}{\Xi \mid \Gamma; - \mid \Theta \vdash Y(\prod \alpha.\, (\alpha \to \alpha) \to \alpha)Y}$$

Given a definable relation ρ we may construct a proposition $\rho(x, y)$. On the other hand, if ϕ is a proposition containing two free variables x and y, then we may construct the definable relation $(x, y).\, \phi$. The next lemma tells us that these constructions give a correspondence between definable relations and propositions, which is bijective up to provable equivalence in the logic.

Lemma 2.27. *Suppose ϕ is a proposition with at least two free variables $x\colon \sigma, y\colon \tau$. Then*

$$((x\colon \sigma, y\colon \tau).\, \phi)(x, y) \supset\subset \phi$$

Suppose $\rho\colon \mathsf{Rel}(\sigma, \tau)$ is a definable relation, then

$$\rho \equiv (x\colon \sigma, y\colon \tau).\, \rho(x, y).$$

Proof. The first biimplication follows trivially from Rule 2.25. For the second, we need to prove

$$\forall z\colon \sigma, w\colon \tau.\, \rho(z, w) \supset\subset ((x\colon \sigma, y\colon \tau).\, \rho(x, y))(z, w),$$

which is trivial by the same rule. □

The substitution axiom above implies the *replacement* rule:

Lemma 2.28.

$$\frac{\Xi \mid \Gamma \mid - \vdash t =_\sigma t' \qquad \Xi \mid \Gamma, x\colon \sigma; - \vdash u\colon \tau}{\Xi \mid \Gamma \mid - \vdash u[t/x] =_\tau u[t'/x]}$$

Proof. Consider the definable relation

$$\rho = (y\colon \sigma, z\colon \sigma).\, u[y/x] =_\tau u[z/x].$$

Clearly $\rho(t, t)$ holds, so by substitution $\rho(t, t')$ holds. □

Lemma 2.29.

$$\rho(x, y) \wedge \rho'(x', y') \supset \rho \otimes \rho'(x \otimes x', y \otimes y')$$

Proof. Suppose $\rho(x, y) \wedge \rho'(x', y')$ and that $(\rho \multimap \rho' \multimap R)(t, t')$. Then clearly $R(t\ x\ x', t'\ y\ y')$ and thus, since

$$\text{let } x \otimes x' \text{ be } x \otimes x' \text{ in } t\ x\ x' = t\ x\ x',$$

we conclude $\rho \otimes \rho'(x \otimes x', y \otimes y')$. □

Lemma 2.30. *For $x\colon \sigma$, $(!x) \downarrow$ always holds in the logic.*

Proof. Define $f\colon\ !\sigma \multimap I$ as $\lambda^\circ x\colon !\sigma$. let $!y$ be x in $\star$. Then clearly $f(!x) = \star$. □

Lemma 2.31. *For any $\rho\colon \mathsf{Rel}(\sigma, \tau)$, $x\colon \sigma, y\colon \tau$*

$$x \rho y \supset\subset !x(!\rho)!y$$

Proof. Since $\epsilon(!x) =$ let $!y$ be $!x$ in $y = x$ this follows from Lemma 2.30. □

2.2.5 Admissible relations preserved by structure maps

We now proceed to show a couple of practical lemmas expressing that various structure maps preserve admissible relations. The maps that we are interested in are

$$\begin{gathered}\epsilon_\sigma\colon\ !\sigma \multimap \sigma\\ \delta_\sigma\colon\ !\sigma \multimap !!\sigma,\end{gathered}$$

which, categorically in the models of PILL$_Y$, are the structure maps of a comonad, and

$$\begin{gathered}d_\sigma\colon\ !\sigma \multimap !\sigma\otimes!\sigma\\ e_\sigma\colon\ !\sigma \multimap I,\end{gathered}$$

which are the maps that make the comonad into a linear category. The maps are syntactically given as

$$\begin{array}{rcl}\epsilon_\sigma &=& \lambda^\circ x\colon !\sigma.\,\text{let } !y \text{ be } x \text{ in } y\\ \delta_\sigma &=& \lambda^\circ x\colon !\sigma.\,\text{let } !y \text{ be } x \text{ in } !!y\\ d_\sigma &=& \lambda^\circ x\colon !\sigma.\,\text{let } !y \text{ be } x \text{ in } !y\otimes!y\\ e_\sigma &=& \lambda^\circ x\colon !\sigma.\,\text{let } !y \text{ be } x \text{ in } \star\,.\end{array}$$

Lemma 2.32. *For all admissible relations $\rho\colon \mathsf{AdmRel}(\sigma,\tau)$,*

$$(\epsilon_\sigma,\epsilon_\tau)\colon !\rho \multimap \rho,\quad (\delta_\sigma,\delta_\tau)\colon !\rho \multimap !!\rho$$

are maps of relations, i.e., $!\rho(x,y)$ implies $\rho(\epsilon_\sigma x,\epsilon_\tau y)$ and $!!\rho(\delta_\sigma x,\delta_\tau y)$.

Proof. The lemma clearly holds in the case of x,y of the form $!x',!y'$. Since $\rho(\epsilon_\sigma x,\epsilon_\tau y)$ and $!!\rho(\delta_\sigma x,\delta_\tau y)$ both define admissible relations from $!\sigma$ to $!\tau$, by Rule 2.18 we conclude that $!\rho(x,y)$ implies

$$(x\downarrow \supset\!\subset y\downarrow) \supset \rho(\epsilon_\sigma x,\epsilon_\tau y)$$

and $(x\downarrow \supset\!\subset y\downarrow) \supset !!\rho(\delta_\sigma x,\delta_\tau y)$. Since $!\rho(x,y) \supset (x\downarrow \supset\!\subset y\downarrow)$ we are done. □

Lemma 2.33. *For all admissible relations $\rho\colon \mathsf{AdmRel}(\sigma,\tau)$,*

$$(d_\sigma,d_\tau)\colon !\rho \multimap !\rho\otimes!\rho,\quad (e_\sigma,e_\tau)\colon !\rho \multimap I_{Rel}$$

are maps of relations, i.e., $!\rho(x,y)$ implies $!\rho\otimes!\rho(d_\sigma x,d_\tau y)$ and $I_{Rel}(e_\sigma x,e_\tau y)$.

Proof. To prove that (d,d) is a map of relations, since

$$\text{let } x'\otimes x'' \text{ be } (\text{let } !y \text{ be } x \text{ in } !y\otimes!y) \text{ in } t\;x'\;x'' = \text{let } !y \text{ be } x \text{ in } t\;!y\;!y$$

we need to prove that

$$\begin{gathered}!\rho(x,y) \supset (\forall\alpha,\beta,R\colon \mathsf{AdmRel}(\alpha,\beta)).\,\forall t\colon !\sigma \multimap !\sigma \multimap \alpha, t'\colon !\tau \multimap !\tau \multimap \beta.\\ t(!\rho \multimap !\rho \multimap R)t \supset R(\text{let } !z \text{ be } x \text{ in } t\;!z\;!z, \text{let } !z \text{ be } y \text{ in } t'\;!z\;!z)\end{gathered}$$

Since the expression on the right hand side of the first $\supset$ is admissible in x,y and $!\rho(x,y) \supset x\downarrow \supset\!\subset y\downarrow$, by Rule 2.18 it suffices to prove the implication in the case $x = !x', y = !y'$. In this case, let $!z$ be x in $t\;!z\;!z = t\;!x'\;!x'$, so the implication is trivial.

To prove that (e,e) is a map of relations, we need to prove that

$$\begin{gathered}!\rho(x,y) \supset (\forall\alpha,\beta,R\colon \mathsf{AdmRel}(\alpha,\beta)).\,\forall z\colon \alpha, w\colon \beta.\\ zRw \supset (\text{let } \star \text{ be } (\text{let } !v \text{ be } x \text{ in} \star) \text{ in } z)R(\text{let } \star \text{ be } (\text{let } !v \text{ be } y \text{ in} \star) \text{ in } w).\end{gathered}$$

The implication clearly holds in the case of x,y of the form $!x',!y'$, and so, since $!\rho(x,y) \supset x\downarrow \wedge y\downarrow$, as before we conclude from Rule 2.18 that the implication holds in general. □

2.2.6 Extensionality and Identity Extension Schemes

Consider the two **extensionality schemes**:

$$(\forall x\colon \sigma.\, t\, x =_\tau u\, x) \supset t =_{\sigma \multimap \tau} u$$
$$(\forall \alpha\colon \mathsf{Type}.\, t\, \alpha =_\tau u\, \alpha) \supset t =_{\prod \alpha\colon \mathsf{Type}.\tau} u.$$

These are taken as axioms in [23], but we shall not take these as axioms as we would like to be able to talk about models that are not necessarily extensional.

Lemma 2.34. *It is provable in the logic that*

$$\forall f, g\colon \sigma \to \tau.\, (\forall x\colon \sigma.\, f(!x) =_\tau g(!x)) \supset \forall x\colon !\sigma.\, f(x) =_\tau g(x).$$

In particular, extensionality implies

$$\forall f, g\colon \sigma \to \tau.\, (\forall x\colon \sigma.\, f(!x) =_\tau g(!x)) \supset f =_{\sigma\to\tau} g$$

Proof. This is just a special case of Rule 2.18. □

The schema

$$- \mid - \mid - \vdash \forall \vec{\alpha}\colon \mathsf{Type}.\, \sigma[eq_{\vec{\alpha}}] \equiv eq_{\sigma(\vec{\alpha})}$$

is called the **identity extension schema**. Here σ ranges over all types, and $eq_{\vec{\alpha}}$ is short notation for $eq_{\alpha_1}, \ldots, eq_{\alpha_n}$.

For any type $\beta, \alpha_1, \ldots, \alpha_n \vdash \sigma(\beta, \vec{\alpha})$ we can form the **parametricity schema**:

$$- \mid - \mid - \vdash \forall \vec{\alpha} \forall u\colon (\textstyle\prod \beta.\, \sigma).\, \forall \beta, \beta'.\, \forall R\colon \mathsf{AdmRel}(\beta, \beta').\, (u\, \beta)\sigma[R, eq_{\vec{\alpha}}](u\, \beta'),$$

where, for readability, we have omitted $:\mathsf{Type}$ after β, β'.

Proposition 2.35. *The identity extension schema implies the parametricity schema.*

Proof. The identity extension schema tells us that

$$\forall \vec{\alpha} \forall u\colon (\textstyle\prod \beta.\, \sigma).\, u(\prod \beta.\, \sigma)[eq_{\vec{\alpha}}]u.$$

Writing out this expression using Rule 2.23 for the relational interpretation of polymorphic types, one obtains the parametricity schema. □

In the case of second-order lambda-calculus, the parametricity schema implied identity extension for the pure calculus, since it provided the case of polymorphic types in a proof by induction. It is interesting to notice that this does not seem to be the case for PILL$_Y$, since it seems that we need identity extension to prove for example $eq_\sigma \otimes eq_\tau \equiv eq_{\sigma\otimes\tau}$.

Lemma 2.36. *Given linear contexts C and C', suppose*

$$\forall x\colon \sigma.\, \forall y\colon \tau.\, C[x \otimes y] =_\omega C'[x \otimes y].$$

then

$$\forall z\colon \sigma \otimes \tau.\, \textit{let } x \otimes y \textit{ be } z \textit{ in } C[x \otimes y] =_\omega \textit{let } x \otimes y \textit{ be } z \textit{ in } C'[x \otimes y]$$

Proof. Consider

$$f = \lambda^\circ x\colon \sigma.\, \lambda^\circ y\colon \tau.\, C[x \otimes y] \qquad f' = \lambda^\circ x\colon \sigma.\, \lambda^\circ y\colon \tau.\, C'[x \otimes y]$$

then

$$f\ (eq_\sigma \multimap eq_\tau \multimap eq_\omega)\ f'.$$

If $z\colon \sigma \otimes \tau$ then by identity extension $eq_\sigma \otimes eq_\tau(z, z)$. By definition of $eq_\sigma \otimes eq_\tau$ we have

$$\text{let } x \otimes x' \text{ be } z \text{ in } fxx' =_\omega \text{let } x \otimes x' \text{ be } z \text{ in } f'xx'$$

which proves the lemma. □

This completes our presentation of LAPL. In the following section we show how to use the logic to prove various consequences of parametricity. We shall write "using extensionality" and "using identity extension" to mean that we assume the extensionality schemes and the identity extension schema, respectively.

3 Proofs in LAPL

3.1 Logical Relations Lemma

Lemma 3.1. *In pure LAPL, for $\vec{\alpha}, \beta \vdash \tau$, $\vec{\alpha} \vdash \omega$ and $\vec{\rho}\colon \mathsf{Rel}(\vec{\sigma}, \vec{\tau})$,*

$$\tau[\omega/\beta][\vec{\rho}] \equiv \tau[\vec{\rho}, \omega[\vec{\rho}]]$$

Proof. Simple induction on the structure of τ. The cases $\tau = \alpha_i$ and $\tau = \beta$ are trivial. For the case $\tau = \tau' \otimes \tau''$,

$$\begin{gathered}(\tau' \otimes \tau'')[\omega/\beta]\ [\vec{\rho}] \equiv \tau'[\omega/\beta][\vec{\rho}] \otimes \tau''[\omega/\beta][\vec{\rho}] \equiv \\ \tau'[\rho, \omega[\vec{\rho}]] \otimes \tau''[\rho, \omega[\vec{\rho}]] \equiv (\tau' \otimes \tau'')[\rho, \omega[\vec{\rho}]]\end{gathered}$$

Likewise for the cases of $\tau = \tau' \multimap \tau''$ and $\tau = !\tau'$. The last case is $\tau = \prod \alpha'.\, \tau'$ and in this case α' is not free in ω, so

$$\begin{gathered}(\prod \alpha'.\, \tau')[\omega/\beta][\vec{\rho}] \equiv \forall \alpha'_0, \alpha'_1, R\colon \mathsf{AdmRel}(\alpha'_0, \alpha'_1).\, \tau'[\omega/\beta][\vec{\rho}, R] \equiv \\ \forall \alpha'_0, \alpha'_1, R\colon \mathsf{AdmRel}(\alpha'_0, \alpha'_1).\, \tau'[\vec{\rho}, R, \omega[\vec{\rho}]] \equiv (\prod \alpha'.\, \tau')[\vec{\rho}, \omega[\vec{\rho}]]\end{gathered}$$

□

Lemma 3.2. *Suppose $\Xi \mid \Gamma, z\colon \sigma; - \vdash t(z)\colon \tau$ and $\Xi \mid \Gamma, z'\colon \sigma'; - \vdash t'(z')\colon \tau'$ and*

$$\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma, \sigma'), \rho'\colon \mathsf{AdmRel}(\tau, \tau').$$

Then

$$\begin{aligned}\Xi \mid \Gamma \mid \Theta \mid \quad & \forall x\colon \sigma, y\colon \sigma'.\, \rho(x, y) \supset \rho'(t(x), t'(y)) \vdash \\ & \forall x\colon !\sigma, y\colon !\sigma'.\, !\rho(x, y) \supset \rho'(\textit{let } !z \textit{ be } x \textit{ in } t(z), \textit{let } !z' \textit{ be } y \textit{ in } t'(z'))\end{aligned}$$

Proof. Consider the special case of Rule 2.18 used on the relations $(x\colon !\sigma, y\colon !\sigma').\, !\rho(x, y)$ and

$$(x\colon !\sigma, y\colon !\sigma').\, \rho'(\text{let } !z \text{ be } x \text{ in } t(z), \text{let } !z' \text{ be } y \text{ in } t'(z')).$$

This gives us

$$\begin{aligned}\Xi \mid \Gamma \mid \Theta \mid \quad & \forall x\colon \sigma, y\colon \sigma'.\, !\rho(!x, !y) \supset \rho'(\text{let } !z \text{ be } !x \text{ in } t(z), \text{let } !z' \text{ be } !y \text{ in } t'(z')) \vdash \\ & \forall x\colon !\sigma, y\colon !\sigma'.\, !\rho(x, y) \wedge (x \downarrow\supset\subset y \downarrow) \supset \rho'(\text{let } !z \text{ be } x \text{ in } t(z), \text{let } !z' \text{ be } y \text{ in } t'(z'))\end{aligned}$$

From this we conclude the desired implication using the fact that $!\rho(!x, !y) \supset\subset \rho(x, y)$ (Lemma 2.31), let $!z$ be $!x$ in $t(z) = t(x)$ and $!\rho(x, y) \supset x \downarrow \supset\subset y \downarrow$. □

Lemma 3.3 (Logical Relations Lemma). *In pure LAPL, for any closed term* $- \mid -; - \vdash t \colon \tau$,

$$t \tau t.$$

In words, any closed term of closed type, is related to itself in the relational interpretation of the type.

Proof. We will prove that for any term

$$\vec{\alpha} \mid \vec{x}' \colon \vec{\sigma}'(\alpha); \vec{x} \colon \vec{\sigma}(\alpha) \vdash t(\vec{\alpha}, \vec{x}', \vec{x}) \colon \tau$$

in the pure calculus; the proposition

$$\begin{gathered} - \mid -; - \vdash \forall \vec{\alpha}, \vec{\beta}. \forall \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}). \forall \vec{x} \colon \vec{\sigma}(\vec{\alpha}), \vec{y} \colon \vec{\sigma}(\vec{\beta}). \forall \vec{x}' \colon \vec{\sigma}'(\vec{\alpha}), \vec{y}' \colon \vec{\sigma}'(\vec{\beta}). \\ \vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}' \supset t(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}]t(\vec{\beta}, \vec{y}', \vec{y}) \end{gathered}$$

holds in the logic. Here $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ is short for

$$x_1 \sigma_1 [\vec{R}] y_1 \wedge \ldots \wedge x_n \sigma_n [\vec{R}] y_n$$

The special case of the vectors $\vec{\alpha}, \vec{\sigma}, \vec{\sigma}'$ of length zero is the statement of the lemma. The proof proceeds by structural induction on t, and it is for the induction we need the seemingly stronger induction hypothesis described above.

Case $t = x_i$**:** In this case $\vec{x}$ is of length one, and the proposition is trivial.

Case $t = x_i'$**:** In this case $\vec{x}$ is empty, and this case is also trivial.

Case $t = \star$**:** We always have $\star I[\vec{R}]\star$.

Case $t = Y$**:** This is Axiom 2.26

Case $t = \lambda^\circ x_{n+1} \colon \sigma_{n+1}. t'$**:** By induction, the proposition holds for t'. We must show that if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})(\sigma_{n+1} \multimap \tau)[\vec{R}]t(\vec{\beta}\vec{y}'\vec{y}).$$

The induction hypothesis tells us that if further $x_{n+1}\sigma_{n+1}[\vec{R}]y_{n+1}$, then

$$t'(\vec{\alpha}, \vec{x}', \vec{x}, x_{n+1})\tau[\vec{R}]t'(\vec{\beta}, \vec{y}', \vec{y}, y_{n+1}),$$

and since $t(\vec{\alpha}, \vec{x}', \vec{x})x_{n+1} = t'(\vec{\alpha}, \vec{x}', \vec{x}, x_{n+1})$ we have the desired result.

Case $t = t' \; t''$**:** By induction the proposition holds for the terms t', t'', and so since

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{y}) \; t''(\vec{\alpha}, \vec{x}', \vec{z})$$

the proposition holds by definition of $(\tau \multimap \tau')[\vec{R}]$.

Case $t = t' \otimes t''$**:** By induction, the proposition holds for t', t''. Clearly

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{y}) \otimes t''(\vec{\alpha}, \vec{x}', \vec{z})$$

and so the proposition holds by Lemma 2.29.

Case $t = \Lambda\alpha_{m+1}.\, t'$: We must show that if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})(\prod \alpha_{m+1}.\, \tau)[\vec{R}]t(\vec{\beta}, \vec{y}', \vec{y}),$$

i.e., for all $\alpha_{m+1}, \alpha'_{m+1}, R_{m+1} \colon \mathsf{AdmRel}(\alpha_{m+1}, \alpha'_{m+1})$,

$$t(\vec{\alpha}, \vec{x}', \vec{x})\ \alpha_{m+1}\tau[\vec{R}, R_{m+1}]t(\vec{\beta}, \vec{y}', \vec{y})\ \alpha'_{m+1},$$

By induction, the proposition holds for t'. But up to the position of the quantifiers

$$\forall\alpha_{m+1}, \alpha'_{m+1}, R_{m+1} \colon \mathsf{AdmRel}(\alpha_{m+1}, \alpha'_{m+1}),$$

this is exactly the proposition we need, and the rest of the proof is just simple logic.

Case $t = t'(\omega)$: By induction, the proposition holds for t'. So since $t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{x})(\omega(\vec{\alpha}))$, if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}, \omega[\vec{R}]]t(\vec{\beta}, \vec{y}', \vec{y}).$$

By Lemma 3.1, we get the desired result.

Case $t = !t'$: In this case $\vec{x}$ is of length zero. By induction, under the usual assumptions,

$$t'(\vec{\alpha}, \vec{x}')\tau[\vec{R}]t'(\vec{\beta}, \vec{y}').$$

Since $t(\vec{\alpha}, \vec{x}') = !t'(\vec{\alpha}, \vec{x}')$, we need to show

$$!t'(\vec{\alpha}, \vec{x}')!\tau[\vec{R}]!t'(\vec{\beta}, \vec{y}').$$

which follows from Lemma 2.31.

Case $t =$ let $z \colon \omega \otimes z' \colon \omega'$ be t' in t'': We know by induction that

$$t'(\vec{\alpha}, \vec{x}', \vec{x})(\omega \otimes \omega')[\vec{R}]t'(\vec{\beta}, \vec{y}', \vec{y}),$$

and if further $z\omega[R]v$ and $z'\omega'[R]v'$ then

$$t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')\tau[\vec{R}]t''(\vec{\beta}, \vec{y}', \vec{y}, v, v').$$

The latter tells us that

$$\lambda^\circ z, z'.\, t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')(\omega[R] \multimap \omega'[R] \multimap \tau[R])\lambda^\circ v, v'.\, t''(\vec{\beta}, \vec{y}', \vec{y}, v, v'),$$

so by definition of $\omega[R] \otimes \omega'[R]$, we get

$$\begin{gathered}\text{let } z \colon \omega \otimes z' \colon \omega' \text{ be } t'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')\tau[R] \\ \text{let } v \colon \omega \otimes v' \colon \omega' \text{ be } t'(\vec{\beta}, \vec{y}', \vec{y}) \text{ in } t''(\vec{\beta}, \vec{y}', \vec{y}, v, v')\end{gathered}$$

as desired.

Case $t =$ let $!z \colon \omega$ be t' in t'': By definition

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = \text{let } !z \text{ be } t'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } t''(\vec{\alpha}, \vec{x}', \vec{x}, z).$$

Suppose we are given $\vec{\alpha}, \vec{\beta}, \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta})$, and suppose $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ and $\vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$. If we further know $z\omega[R]z'$, then by induction

$$t''(\alpha, \vec{x}', \vec{x}, z)\tau[R]t''(\beta, \vec{y}', \vec{y}, z').$$

By Lemma 3.2 we conclude that if $v(!\omega[R])v'$ then

$$(\text{let } !z \text{ be } v \text{ in } t''(\alpha, \vec{x}', \vec{x}, z))\tau[R](\text{let } !z \text{ be } v' \text{ in } t''(\beta, \vec{y}', \vec{y}, z)).$$

Since by induction $t'(\vec{\alpha}, \vec{x}', \vec{x})!\omega[\vec{R}]t'(\vec{\alpha}, \vec{y}', \vec{y})$, we are done.

$$\begin{array}{rcl}
\sigma & \cong & \prod \alpha .\, (\sigma \multimap \alpha) \multimap \alpha \\
\sigma \otimes \tau & \cong & \prod \alpha .\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha \\
I & \cong & \prod \alpha .\, \alpha \multimap \alpha \\
0 & = & \prod \alpha .\, \alpha \\
1 & = & \prod \alpha .\, \alpha \\
\sigma + \tau & = & \prod \alpha .\, (\sigma \multimap \alpha) \to (\tau \multimap \alpha) \to \alpha \\
\sigma \times \tau & = & \prod \alpha .\, (\sigma \multimap \alpha) + (\tau \multimap \alpha) \multimap \alpha \\
\mathbb{N} & = & \prod \alpha .\, (\alpha \multimap \alpha) \to \alpha \multimap \alpha \\
\coprod \alpha .\, \sigma & = & \prod \beta .\, (\prod \alpha .\, \sigma \multimap \beta) \multimap \beta \\
\mu\alpha .\, \sigma & = & \prod \alpha .\, (\sigma \multimap \alpha) \multimap \alpha \\
\nu\alpha .\, \sigma & = & \coprod \alpha .\, !(\alpha \multimap \sigma) \otimes \alpha
\end{array}$$

Figure 5: Types definable using parametricity

Case $s =$ let $\star$ be s' in s'': By induction, if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ and $\vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$ then

$$s''(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}]s''(\vec{\beta}, \vec{y}', \vec{y})$$

and

$$s'(\vec{\alpha}, \vec{x}', \vec{x})I_{Rel}s'(\vec{\beta}, \vec{y}', \vec{y}).$$

The definition of the latter tells us exactly that

$$(\text{let } \star \text{ be } s'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } s''(\vec{\alpha}, \vec{x}', \vec{x}))\tau[\vec{R}](\text{let } \star \text{ be } s'(\vec{\beta}, \vec{y}', \vec{y}) \text{ in } s''(\vec{\beta}, \vec{y}', \vec{y}))$$

as desired.

□

3.2 A category of linear functions

At this point we wish to show certain types definable via polymorphism as summed up in Figure 5. To state this precisely, we introduce for each kind context Ξ the category $\mathbf{LinType}_\Xi$ as follows:

Objects are types $\Xi \mid -; - \vdash \sigma\colon \mathsf{Type}$.

Morphisms $[\Xi \mid -; - \vdash f\colon \sigma \multimap \tau]$ are equivalence classes of terms of type $\sigma \multimap \tau$; the equivalence relation on these terms being *internal* equality.

Composition in this category is given by lambda abstraction, i.e. $f\colon \sigma \multimap \tau$ composed with $g\colon \omega \multimap \sigma$ yields $\lambda^\circ x\colon \omega .\, f(gx)$.

We start by proving that under the assumption of identity extension and extensionality, for all types $\Xi \vdash \sigma\colon \mathsf{Type}$ we have an isomorphism of objects of $\mathbf{LinType}_\Xi$:

$$\sigma \cong \prod \alpha .\, (\sigma \multimap \alpha) \multimap \alpha$$

for α not free in σ. We can define terms

$$f\colon \sigma \multimap \prod \alpha .\, ((\sigma \multimap \alpha) \multimap \alpha)$$

and

$$g\colon \prod \alpha.\, ((\sigma \multimap \alpha) \multimap \alpha) \multimap \sigma$$

by

$$f = \lambda^\circ x\colon \sigma.\, \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \alpha.\, h\, x$$

and

$$g = \lambda^\circ x\colon \prod \alpha.\, ((\sigma \multimap \alpha) \multimap \alpha).\, x\, \sigma\, id_\sigma$$

Clearly

$$g\,(f\ x) = (f\ x)\, \sigma\, id_\sigma = x$$

so $gf = id_\sigma$. Notice that this only involve external equality and thus we did not need extensionality here.

Proposition 3.4. *Using identity extension and extensionality, one may prove that fg is internally equal to the identity.*

Proof. For a term $a\colon \prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ we have

$$f \circ g\, a = \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \alpha.\, h(a\, \sigma\, id_\sigma).$$

Using extensionality, it suffices to prove that

$$\Xi, \alpha \mid h\colon \sigma \multimap \alpha \mid - \vdash h(a\, \sigma\, id_\sigma) =_\alpha a\, \alpha\, h$$

holds in the internal logic.

By the parametricity schema we know that for any admissible relation $\rho\colon \mathsf{AdmRel}(\tau, \tau')$

$$(a\, \tau)((eq_\sigma \multimap \rho) \multimap \rho)(a\, \tau')$$

If we instantiate this with the admissible relation $\langle h \rangle$, we get

$$(a\, \sigma)((eq_\sigma \multimap \langle h \rangle) \multimap \langle h \rangle)(a\, \alpha)$$

Since $id_\sigma(eq_\sigma \multimap \langle h \rangle)h$ we know that $(a\ \sigma\ id_\sigma)\langle h \rangle(a\ \alpha\ h)$, i.e.,

$$h(a\, \sigma\, id_\sigma) =_\alpha a\, \alpha\, h,$$

as desired. □

This proof may essentially be found in [5].

Intuitively, what happens here is that σ is a subtype of $\prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$, where the inclusion f maps x to application at x. We use parametricity to show that $\prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ does not contain anything that is not in σ.

3.3 Tensor types

The goal of this section is to prove

$$\sigma \otimes \tau \cong \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

using identity extension and extensionality, for $\Xi \vdash \sigma\colon \mathsf{Type}$ and $\Xi \vdash \tau\colon \mathsf{Type}$ types in the same context. The isomorphism is in the category $\mathbf{LinType}_\Xi$.

This isomorphism leads to the question of wether tensor types are actually superfluous in the language. The answer is yes in the following sense: Call the language without tensor types (and I) t and the language as is T. Then there are transformations $p : T \to t$ and $i : t \to T$, i being the inclusion, such that $p \circ i = id_T$ and $i \circ p \cong id_t$. This is all being stated more precisely, not to mention proved, in [16]. In this paper we settle for the isomorphism above.

We can construct terms

$$f : \sigma \otimes \tau \multimap \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

and

$$g : (\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha) \multimap \sigma \otimes \tau$$

by

$$f\ y = \text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x'$$

and

$$g\ y = y\ \sigma \otimes \tau\ \textit{pairing},$$

where the map *pairing* : $\sigma \multimap \tau \multimap \sigma \otimes \tau$ is

$$\textit{pairing} = \lambda^\circ x \colon \sigma.\, \lambda^\circ x' \colon \tau.\, x \otimes x'.$$

Let us show that the composition $g \circ f$ is the identity.

$$\begin{gathered} g \circ f\ y = g(\text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x') = \\ (\text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x')\ \sigma \otimes \tau\ \textit{pairing} = \\ (\Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, \text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } h\, x\, x')\ \sigma \otimes \tau\ \textit{pairing} = \\ \text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } x \otimes x' = y. \end{gathered}$$

Proposition 3.5. *Using extensionality and identity extension one may prove that the composition*

$$fg\colon (\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha) \multimap (\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha)$$

is internally equal to the identity.

Proof. We compute

$$\begin{gathered} f \circ g\ y = f(y\ \sigma \otimes \tau\ \textit{pairing}) = \\ \text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } (y\ \sigma \otimes \tau\ \textit{pairing}) \text{ in } \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x' \end{gathered}$$

Suppose we are given a type α and a map $h\colon \sigma \multimap \tau \multimap \alpha$. We can define $\phi_h \colon \sigma \otimes \tau \multimap \alpha$ as

$$\phi_h = \lambda^\circ y \colon \sigma \otimes \tau.\, \text{let } x \otimes x' \colon \sigma \otimes \tau \text{ be } y \text{ in } h\, x\, x'.$$

Then $\phi_h(\mathit{pairing}\; x\; x') = h\; x\; x'$, which means that $\mathit{pairing}(\mathit{eq}_\sigma \multimap \mathit{eq}_\tau \multimap \langle\phi_h\rangle)h$. By the parametricity schema

$$\begin{gathered}\Xi, \alpha \mid h\colon \sigma \multimap \tau \multimap \alpha, y\colon \textstyle\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha \mid - \mid \top \vdash \\ (y\; \sigma \otimes \tau)((\mathit{eq}_\sigma \multimap \mathit{eq}_\tau \multimap \langle\phi_h\rangle) \multimap \langle\phi_h\rangle)(y\; \alpha)\end{gathered}$$

so

$$(y\; \sigma \otimes \tau\; \mathit{pairing})\langle\phi_h\rangle(y\; \alpha\; h),$$

i.e,

$$\phi_h(y\; \sigma \otimes \tau\; \mathit{pairing}) =_\alpha y\; \alpha\; h.$$

Writing this out we get

$$\begin{gathered}\Xi, \alpha \mid h\colon \sigma \multimap \tau \multimap \alpha, y\colon \textstyle\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha \mid - \mid \top \vdash \\ \text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } (y\; \sigma \otimes \tau\; \mathit{pairing}) \text{ in } h\; x\; x' =_\alpha y\; \alpha\; h.\end{gathered}$$

Using extensionality we get

$$\Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, \text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } (y\; \sigma \otimes \tau\; \mathit{pairing}) \text{ in } (h\; x\; x') =_\alpha y.$$

This is enough, since by the rules for external equality the left hand side is

$$\text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } (y\; \sigma \otimes \tau\; \mathit{pairing}) \text{ in } (\Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\; x\; x').$$

□

3.4 Unit object

The goal of this section is to prove that identity extension together with extensionality implies

$$I \cong \textstyle\prod \alpha.\, \alpha \multimap \alpha.$$

The isomorphism holds in $\mathbf{LinType}_\Xi$ for all Ξ.

We first define maps $f\colon I \multimap \prod \alpha.\, \alpha \multimap \alpha$ and $g\colon (\prod \alpha.\, \alpha \multimap \alpha) \multimap I$ as

$$\begin{gathered} f = \lambda^\circ x\colon I.\, \text{let } \star \text{ be } x \text{ in } id, \\ g = \lambda^\circ t\colon \textstyle\prod \alpha.\, \alpha \multimap \alpha.\, t\; I\; \star,\end{gathered}$$

where

$$id = \Lambda\alpha.\, \lambda^\circ y\colon \alpha.\, y.$$

We first notice that

$$\begin{gathered} g(f(x)) = (\text{let } \star \text{ be } x \text{ in } id)\; I\; \star = \\ \text{let } \star \text{ be } x \text{ in } (id\; I\; \star) = \text{let } \star \text{ be } x \text{ in} \star = x.\end{gathered}$$

Proposition 3.6. *Using identity extension and extensionality, we have that fg is internally equal to the identity on $\prod \alpha.\, \alpha \multimap \alpha$.*

Proof. First we write out the definition

$$fg = \lambda^\circ t\colon (\textstyle\prod \alpha.\, \alpha \multimap \alpha).\, \text{let } \star \text{ be } (t\; I\; \star) \text{ in } id.$$

We show that for any $t\colon \prod \alpha.\, \alpha \multimap \alpha$, for any type σ, and any $x\colon \sigma$ we have $fg(t)\; \sigma\; x =_\sigma t\; \sigma\; x$.

Given σ, x as above, we can define $h\colon I \multimap \sigma$ as $h = \lambda^\circ z\colon I.\,\text{let} \star \text{be } z \text{ in } x$. Then $\langle h \rangle$ is admissible, so by identity extension

$$(t\,I)(\langle h \rangle \multimap \langle h \rangle)(t\,\sigma).$$

Since $h(\star) = x$ we have $h(t\,I\,\star) =_\sigma t\,\sigma\,x$, and by definition

$$h(t\,I\,\star) = \text{let } \star \text{ be } (t\,I\,\star) \text{ in } x = \text{let } \star \text{ be } (t\,I\,\star) \text{ in } (id\,\sigma\,x) = \\ (\text{let } \star \text{ be } (t\,I\,\star) \text{ in } id)\,\sigma\,x = f \circ g(t)\,\sigma\,x.$$

□

3.5 Initial objects and coproducts

We define

$$0 = \prod \alpha.\,\alpha$$

For each Ξ this defines a weak initial object in $\mathbf{LinType}_\Xi$, since for any type $\Xi \vdash \sigma$, there exists a term $0_\sigma\colon 0 \multimap \sigma$, defined as

$$\lambda^\circ x\colon 0.\,x\,\sigma$$

Proposition 3.7. *Suppose $f\colon 0 \multimap \sigma$ for some type $\Xi \vdash \sigma$. Using identity extension and extensionality it is provable that $f =_{0 \multimap \sigma} 0_\sigma$. Thus, 0 is an initial object in $\mathbf{LinType}_\Xi$ for each Ξ.*

Proof. First notice that for any map $h\colon \sigma \multimap \tau$, by identity extension $(x\,\sigma)\langle h \rangle(x\,\tau)$ for any $x\colon 0$. Thus, by extensionality, $h \circ 0_\sigma =_{0 \multimap \tau} 0_\tau$ for any $h\colon \sigma \multimap \tau$. In particular, for any type σ, the case $h = 0_\sigma$ gives us $x\,0\,\sigma =_\sigma x\,\sigma$, i.e., $0_0 =_{0 \multimap 0} id_0$. If $f\colon 0 \multimap \sigma$, by the above we have $0_\sigma =_{0 \multimap \sigma} f \circ 0_0 =_{0 \multimap \sigma} f$ □

Next, suppose $\Xi \vdash \sigma, \tau$ are types in the same context. We define

$$\sigma + \tau = \prod \alpha.\,(\sigma \multimap \alpha) \to (\tau \multimap \alpha) \to \alpha$$

and show under the assumption of identity extension and extensionality that this defines a coproduct of σ and τ in $\mathbf{LinType}_\Xi$.

First define terms $in_\sigma\colon \sigma \multimap \sigma + \tau$, $in_\tau\colon \tau \multimap \sigma + \tau$ as

$$\begin{array}{rcl} in_\sigma & = & \lambda^\circ x\colon \sigma.\,\Lambda\alpha.\,\lambda f\colon \sigma \multimap \alpha.\,\lambda g\colon \tau \multimap \alpha.\,f(x) \\ in_\tau & = & \lambda^\circ y\colon \tau.\,\Lambda\alpha.\,\lambda f\colon \sigma \multimap \alpha.\,\lambda g\colon \tau \multimap \alpha.\,g(y) \end{array}$$

For any pair of maps $f\colon \sigma \multimap \omega$, $g\colon \tau \multimap \omega$ define the copairing $[f, g]\colon \sigma + \tau \multimap \omega$ as

$$[f, g] = \lambda^\circ x\colon \sigma + \tau.\,x\,\omega\,!f\,!g,$$

then clearly $[f, g](in_\sigma(x)) = f(x)$ and $[f, g](in_\tau(y)) = g(y)$, and so $\sigma + \tau$ is a weak coproduct of σ and τ in $\mathbf{LinType}_\Xi$. We remark that the copairing constructor can also be defined as a polymorphic term

$$[-,-]\colon \Lambda\alpha.\,(\sigma \multimap \alpha) \to (\tau \multimap \alpha) \to \sigma + \tau \multimap \alpha$$

of *intuitionistic* function type. Of course we can define an even more generel copairing by abstracting σ, τ as well.

Lemma 3.8. *If $h\colon \omega \multimap \omega'$, $f\colon \sigma \multimap \omega$ and $g\colon \tau \multimap \omega$, then using extensionality and identity extension, it is provable that $[h \circ f, h \circ g] =_{\sigma + \tau \multimap \omega'} h \circ [f, g]$.*

Proof. Since

$$f(\mathit{eq}_\sigma \multimap \langle h\rangle)h \circ f$$
$$g(\mathit{eq}_\tau \multimap \langle h\rangle)h \circ g$$

for any $x\colon \sigma + \tau$,

$$(x\,\omega\,!f\,!g)\langle h\rangle(x\,\omega'\,!(h \circ f)\,!(h \circ g))$$

by identity extension, i.e., $h([f, g](x)) = [h \circ f, h \circ g](x)$. □

Lemma 3.9. *Using extensionality and identity extension, it is provable that* $[\mathit{in}_\sigma, \mathit{in}_\tau] =_{\sigma+\tau\multimap\sigma+\tau} \mathit{id}_{\sigma+\tau}$.

Proof. Given any $\omega, a\colon \sigma \multimap \omega, b\colon \tau \multimap \omega$, we have

$$[a, b]([\mathit{in}_\sigma, \mathit{in}_\tau](x)) =_\omega [[a, b] \circ \mathit{in}_\sigma, [a, b] \circ \mathit{in}_\tau](x) =_\omega [a, b](x)$$

for any $x\colon \sigma + \tau$. By unfolding the definition of $[a, b]$ in the above equality we get

$$[\mathit{in}_\sigma, \mathit{in}_\tau](x)\,\omega\,!a\,!b =_\omega x\,\omega\,!a\,!b.$$

Since ω, a, b were arbitrary, extensionality (and Lemma 2.34) implies $[\mathit{in}_\sigma, \mathit{in}_\tau](x) =_{\sigma+\tau} x$ for all x. □

Proposition 3.10. *For any* $f\colon \sigma \multimap \omega$, $g\colon \tau \multimap \omega$ *and* $h\colon \sigma+\tau \multimap \omega$, *if* $h\circ \mathit{in}_\sigma =_{\sigma\multimap\omega} f$ *and* $h\circ \mathit{in}_\tau =_{\tau\multimap\omega} g$, *then it is provable using identity extension and extensionality that* $h =_{\sigma+\tau\multimap\omega} [f, g]$. *Thus* $\sigma + \tau$ *is a coproduct of* σ *and* τ *in* $\mathbf{LinType}_\Xi$.

Proof.

$$[f, g] =_{\sigma+\tau\multimap\omega} [h \circ \mathit{in}_\sigma, h \circ \mathit{in}_\tau] =_{\sigma+\tau\multimap\omega} h \circ [\mathit{in}_\sigma, \mathit{in}_\tau] =_{\sigma+\tau\multimap\omega} h$$

□

3.6 Terminal objects and products

The initial object 0 is also weakly terminal, since for any type σ,

$$\Omega_{\sigma\multimap 0} = Y\,\sigma\,!\mathit{id}_{\sigma\multimap 0}$$

is a term of type $\sigma \multimap 0$. In fact, using parametricity, 0 can be proved to be terminal.

Proposition 3.11. *Suppose* $f, g\colon \sigma \multimap 0$. *Using identity and extensionality it is provable that* $f =_{\sigma\multimap 0} g$. *Thus* 0 *is a terminal object in* $\mathbf{LinType}_\Xi$ *for any* Ξ.

Proof. We will prove

$$\forall x, y\colon 0.\, x =_0 y$$

which, by extensionality, implies the proposition. Suppose we are given $x, y\colon 0$. The term

$$\lambda^\circ z\colon 0.\, z\, 0 \multimap \sigma\, y$$

has type $0 \multimap \sigma$, and thus is equal to 0_σ. This means that $x\,\sigma =_\sigma x\, 0 \multimap \sigma y$. Likewise $x\, 0 \multimap \sigma y =_\sigma y\,\sigma$, so $x\,\sigma =_\sigma y\,\sigma$. Since this holds for all σ, by extensionality $x =_0 y$. □

Suppose σ, τ are types in the same context Ξ. Define

$$\sigma \times \tau = \prod \alpha.\, (\sigma \multimap \alpha) + (\tau \multimap \alpha) \multimap \alpha.$$

This defines a weak product in $\mathbf{LinType}_\Xi$ with projections $\pi_\sigma \colon \sigma \times \tau \multimap \sigma$ and $\pi_\tau \colon \sigma \times \tau \multimap \tau$ defined as

$$\begin{array}{rcl} \pi_\sigma & = & \lambda^\circ x \colon \sigma \times \tau.\, x\, \sigma\, (\mathit{in}_{\sigma \multimap \sigma} \mathit{id}_\sigma) \\ \pi_\tau & = & \lambda^\circ x \colon \sigma \times \tau.\, x\, \tau\, (\mathit{in}_{\tau \multimap \tau} \mathit{id}_\tau) \end{array}$$

The pairing of terms $f \colon \omega \multimap \sigma$ and $g \colon \omega \multimap \tau$ is $\langle f, g \rangle \colon \omega \multimap \sigma \times \tau$ defined as

$$\langle f, g \rangle = \lambda^\circ x \colon \omega.\, \Lambda\alpha.\, \lambda^\circ h \colon (\sigma \multimap \alpha) + (\tau \multimap \alpha).\, [\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ f, \lambda^\circ z \colon \tau \multimap \alpha.\, z \circ g]\, h\, x$$

Then

$$\pi_\sigma(\langle f, g \rangle(x)) = \langle f, g \rangle(x)\, \sigma\, (\mathit{in}_{\sigma \multimap \sigma} \mathit{id}_\sigma) = (\lambda^\circ z \colon \sigma \multimap \sigma.\, z \circ f)\; \mathit{id}_\sigma\, x = f(x)$$

and so $\pi_\sigma \circ \langle f, g \rangle = f$ and likewise $\pi_\tau \circ \langle f, g \rangle = g$ proving that $\sigma \times \tau$ defines a weak product.

Lemma 3.12. *Using identity extension and extensionality it is provable that for any $f \colon \omega \multimap \sigma, g \colon \omega \multimap \tau, k \colon \omega' \multimap \omega$,*

$$\langle f, g \rangle \circ k =_{\omega' \multimap \sigma \times \tau} \langle f \circ k, g \circ k \rangle$$

Proof. The lemma is easily proved by the following direct computation using properties of coproducts established above. The notation $(- \circ k)$ below denotes the term $\lambda^\circ y \colon \omega \multimap \alpha.\, y \circ k$ of type $(\omega \multimap \alpha) \multimap \omega' \multimap \alpha$.

$$\begin{array}{rcl} \langle f \circ k, g \circ k \rangle(x) & =_{\sigma \times \tau} & \Lambda\alpha.\, \lambda^\circ h \colon (\sigma \multimap \alpha) + (\tau \multimap \alpha).\, [\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ f \circ k, \lambda^\circ z \colon \tau \multimap \alpha.\, z \circ g \circ k]\, h\, x \\ & =_{\sigma \times \tau} & \Lambda\alpha.\, \lambda^\circ h.\, [(- \circ k) \circ (\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ f), (- \circ k) \circ (\lambda^\circ z \colon \tau \multimap \alpha.\, z \circ g)]\, h\, x \\ & =_{\sigma \times \tau} & \Lambda\alpha.\, \lambda^\circ h.\, (- \circ k) \circ [(\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ f), (\lambda^\circ z \colon \tau \multimap \alpha.\, z \circ g)]\, h\, x \\ & =_{\sigma \times \tau} & \Lambda\alpha.\, \lambda^\circ h.\, [(\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ f), (\lambda^\circ z \colon \tau \multimap \alpha.\, z \circ g)]\, h\, (k(x)) \\ & =_{\sigma \times \tau} & \langle f, g \rangle \circ k(x) \end{array}$$

□

Lemma 3.13. *Identity extension and extensionality implies that $\langle \pi_\sigma, \pi_\tau \rangle =_{\sigma \times \tau \multimap \sigma \times \tau} \mathit{id}_{\sigma \times \tau}$.*

Proof. We must show that for any $x \colon \sigma \times \tau$, any α and any $h \colon (\sigma \multimap \alpha) + (\tau \multimap \alpha)$

$$[\lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ \pi_\sigma, \lambda^\circ z \colon \sigma \multimap \alpha.\, z \circ \pi_\tau]\, h\, x =_\alpha x\, \alpha\, h$$

In fact, since we are dealing with coproducts, it suffices to show that for any $l \colon \sigma \multimap \alpha$ and $k \colon \tau \multimap \alpha$

$$\begin{array}{rcl} l(\pi_\sigma(x)) & =_\alpha & x\, \alpha\, (\mathit{in}_{\sigma \multimap \alpha}\, l) \\ k(\pi_\tau(x)) & =_\alpha & x\, \alpha\, (\mathit{in}_{\tau \multimap \alpha}\, k) \end{array}$$

We just prove the first of these equations. Since

$$\mathit{id}_\sigma(\mathit{eq}_\sigma \multimap \langle l \rangle) l$$

by parametricity of a polymorphic version of *in*,

$$\mathit{in}_{\sigma \multimap \sigma}(\mathit{id}_\sigma)((\mathit{eq}_\sigma \multimap \langle l \rangle) + (\mathit{eq}_\tau \multimap \langle l \rangle)\mathit{in}_{\sigma \multimap \alpha}(l)$$

and so by parametricity of $x\colon \sigma \times \tau$

$$x\ \sigma\ (\mathit{in}_{\sigma\multimap\sigma}\ \mathit{id}_\sigma)\langle l\rangle x\ \alpha\ (\mathit{in}_{\sigma\multimap\alpha}\ l)$$

i.e.

$$\pi_\sigma(x)\langle l\rangle x\ \alpha\ (\mathit{in}_{\sigma\multimap\alpha}\ l)$$

as desired. □

Proposition 3.14. *Suppose $h\colon \omega \multimap \sigma \times \tau$ is such that $\pi_\sigma \circ h =_{\omega\multimap\sigma} f$ and $\pi_\tau \circ h =_{\omega\multimap\tau} g$ then it is provable using identity extension and extensionality that $h =_{\omega\multimap\sigma\times\tau} \langle f, g\rangle$. Thus $\sigma \times \tau$ is a product of σ and τ in $\mathbf{LinType}_\Xi$.*

Proof.

$$h =_{\omega\multimap\sigma\times\tau} \langle \pi_\sigma, \pi_\tau\rangle \circ h =_{\omega\multimap\sigma\times\tau} \langle \pi_\sigma \circ h, \pi_\tau \circ h\rangle =_{\omega\multimap\sigma\times\tau} \langle f, g\rangle.$$

□

3.7 Natural Numbers

We define the type of natural numbers as

$$\mathbb{N} = \prod \alpha.\,(\alpha \multimap \alpha) \to \alpha \multimap \alpha.$$

We further define terms $0\colon \mathbb{N}$, $s\colon \mathbb{N} \multimap \mathbb{N}$ as

$$0 = \Lambda\alpha.\,\lambda f\colon \alpha \multimap \alpha.\,\lambda^\circ x\colon \alpha.\,x, \quad s = \lambda^\circ y\colon \mathbb{N}.\,\Lambda\alpha.\,\lambda f\colon \alpha \multimap \alpha.\,\lambda^\circ x\colon \alpha.\,f(y\ \alpha\ !f\ x)$$

and prove that $(\mathbb{N}, 0, s)$ is a weak natural numbers object in each $\mathbf{LinType}_\Xi$, and, using parametricity and extensionality, an honest natural numbers object.

Suppose we are given a type σ, a term $a\colon \sigma$ and a morphism $b\colon \sigma \multimap \sigma$. We can then define $h\colon \mathbb{N} \multimap \sigma$ as $h(y) = y\ \sigma\ !b\ a$. Then clearly $h(0) = a$, and $h(s\ x) = b(x\ \sigma\ !b\ a) = b(h(x))$, so $(\mathbb{N}, 0, s)$ is a weak natural numbers object.

We can express the weak natural numbers object property as: for all a, b, there exists an h such that

$$\begin{tikzcd}
I \arrow[r, "0"] \arrow[dr, "a"'] & \mathbb{N} \arrow[r, "s"] \arrow[d, "h"] & \mathbb{N} \arrow[d, "h"] \\
 & \sigma \arrow[r, "b"] & \sigma
\end{tikzcd}$$

commutes.

Lemma 3.15. *Identity Extension and extensionality implies*

$$\forall x\colon \mathbb{N}.\,x\ \mathbb{N}\ !s\ 0 =_{\mathbb{N}} x$$

Proof. Suppose we are given σ, a, b and define h as above. Since $b \circ h = h \circ s$ and $h\ 0 = a$, we have $s(\langle h\rangle \multimap \langle h\rangle)b$ and $0\langle h\rangle a$, by parametricity of x, $(x\ \mathbb{N}\ !s\ 0)\langle h\rangle(x\ \sigma\ !b\ a)$, i.e.,

$$(x\ \mathbb{N}\ !s\ 0)\ \sigma\ !b\ a =_\sigma x\ \sigma\ !b\ a.$$

Letting σ range over all types and a, b over all terms, using extensionality and Lemma 2.34, we have

$$x\ \mathbb{N}\ !s\ 0 =_{\mathbb{N}} x,$$

as desired. □

We can now prove that $\mathbb{N}$ is a natural numbers object in each $\mathbf{LinType}_{\Xi}$.

Lemma 3.16. *Assuming identity extension and extensionality, given σ, a, b, the map h defined as above is up to internal equality the unique h' such that $h'(0) = a$, $h'(s\,x) = b(h'\,x)$.*

Proof. Suppose h' satisfies the requirements of the lemma. Then $s(\langle h'\rangle \multimap \langle h'\rangle)b$ and $0\langle h'\rangle a$ (this is just a reformulation of the requirements), so for arbitrary $x\colon \mathbb{N}$, by parametricity of x,

$$x\,\sigma\,!b\,a =_{\sigma} h'(x\,\mathbb{N}\,!s\,0) =_{\sigma} h'(x).$$

Thus, by extensionality, $h' =_{\mathbb{N}\multimap\sigma} h$. □

3.7.1 Induction principle

The parametricity principle for the natural numbers implies, that if $R\colon \mathsf{AdmRel}(\mathbb{N},\mathbb{N})$, and $x\colon \mathbb{N}$, then

$$(x\,\mathbb{N})((R \multimap R) \to R \multimap R)(x\,\mathbb{N}).$$

So if $s(R \multimap R)s$ and $R(0,0)$, then

$$(x\,\mathbb{N}\,!s\,0)R(x\,\mathbb{N}\,!s\,0).$$

By Lemma 3.15, $x\,\mathbb{N}\,!s\,0 =_{\mathbb{N}} x$, so we can conclude that $R(x,x)$. If ϕ is a proposition on $\mathbb{N}$ such that $(x\colon \mathbb{N}, y\colon \mathbb{N}).\,\phi(x)$ is admissible, then from parametricity we obtain the usual induction principle

$$(\phi(0) \wedge \forall x\colon \mathbb{N}.\,\phi(x) \supset \phi(s(x))) \supset \forall x\colon \mathbb{N}.\,\phi(x).$$

3.8 Types as functors

Definition 3.17. We say that $\vec{\alpha} \vdash \sigma\colon \mathsf{Type}$ is an inductively constructed type, if it can be constructed from free variables $\vec{\alpha}$ and closed types using the type constructors of PILL$_Y$, i.e., $\multimap, \otimes, I, !$ and $\prod \alpha.$.

For example, all types of pure PILL$_Y$ are inductively defined, and if σ is a closed type then $\prod \alpha.\,\sigma \times \alpha$ is an inductively constructed type. However, some models may contain types that are not inductively constructed! For example, in syntactical models, any basic open type, such as the type $\alpha \vdash \textit{lists}(\alpha)$ is not inductively constructed.

We define positive and negative occurences of free type variables in inductively defined types as usual. The type variable α occurs positive in the type α and the positive occurences of a type variable α in $\sigma \multimap \tau$ are the positive occurences of α in τ and the negative in σ. The negative occurences of α in $\sigma \multimap \tau$ are the positive in σ and the negative in τ. The positive and negative occurences of α in $\prod \beta.\,\sigma$ are the positive and negative occurences in σ for $\alpha \neq \beta$. The rest of the type constructors preserve positive and negative occurences of type variables.

If $\sigma(\alpha,\beta)$ is an inductively defined type in which the free type variable α appears only negatively and the free type variable β appears only positively, then we can consider σ as a functor $\mathbf{LinType}^{op} \times \mathbf{LinType} \to \mathbf{LinType}$ by defining the term

$$M_{\sigma(\alpha,\beta)}\colon \prod \alpha, \beta, \alpha', \beta'.\,(\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to \sigma(\alpha,\beta) \multimap \sigma(\alpha',\beta'),$$

which behaves as the morphism part of a functor, i.e., it respects composition and preserves identities. We define $M_{\sigma(\alpha,\beta)}$ by structural induction on σ. This construction immediately generalizes to types with less or more than two free type variables, all of which appear only positively or negatively.

For the base case of the induction, if $\sigma(\alpha,\beta)=\beta$, define

$$M_\beta = \Lambda\alpha,\beta,\alpha',\beta'.\,\lambda f,g.\,g.$$

In the case $\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta)$ we define the term

$$\begin{gathered} M_{\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta)}\colon \\ \prod\alpha,\beta,\alpha',\beta'.\,(\alpha'\multimap\alpha)\to(\beta\multimap\beta')\to(\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta))\multimap\sigma(\beta',\alpha')\multimap\tau(\alpha',\beta') \end{gathered}$$

by

$$\begin{gathered} M_{\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta)} = \Lambda\alpha,\beta,\alpha',\beta'.\,\lambda f,g. \\ \lambda^\circ h\colon \sigma(\beta,\alpha)\multimap\tau(\alpha,\beta).\,(M_\tau\;\alpha\;\beta\;\alpha'\;\beta'\;f\;g)\circ h\circ(M_\sigma\;\beta'\;\alpha'\;\beta\;\alpha\;g\;f). \end{gathered}$$

For bang types, we define:

$$\begin{gathered} M_{!\sigma(\alpha,\beta)} = \Lambda\alpha,\beta,\alpha',\beta'.\,\lambda f\colon\alpha'\multimap\alpha.\,\lambda g\colon\beta\multimap\beta'.\,\lambda^\circ x\colon !\sigma(\alpha,\beta). \\ \text{let } !y \text{ be } x \text{ in } !(M_{\sigma(\alpha,\beta)}\;\alpha\;\beta\;\alpha'\;\beta'\;f\;g\;y). \end{gathered}$$

For tensor types, we define:

$$\begin{gathered} M_{\sigma(\alpha,\beta)\otimes\tau(\alpha,\beta)} = \Lambda\alpha,\beta,\alpha',\beta'.\,\lambda f,g.\,\lambda^\circ z\colon \sigma(\alpha,\beta)\otimes\tau(\alpha,\beta). \\ \text{let } x\otimes y\colon \sigma(\alpha,\beta)\otimes\tau(\alpha,\beta) \text{ be } z \text{ in } (M_\sigma\alpha\;\beta\;\alpha'\;\beta'\;f\;g\;x)\otimes(M_\tau\alpha\;\beta\;\alpha'\;\beta'\;f\;g\;y). \end{gathered}$$

The last case is the case of polymorphic types:

$$\begin{gathered} M_{\prod\omega.\sigma(\alpha,\beta)} = \Lambda\alpha,\beta,\alpha',\beta'.\,\lambda f,g.\,\lambda^\circ z\colon \prod\omega.\,\sigma(\alpha,\beta). \\ \Lambda\omega\colon \mathsf{Type}.\,M_{\sigma(\alpha,\beta)}\;\alpha\;\beta\;\alpha'\;\beta'\;f\;g\;(z\;\omega). \end{gathered}$$

Lemma 3.18. *The term M_σ respects composition and preserves identities, i.e., for $f'\colon\alpha''\multimap\alpha'$, $f\colon\alpha'\multimap\alpha$, $g\colon\beta\multimap\beta'$, and $g'\colon\beta'\multimap\beta''$,*

- $M_{\sigma(\alpha,\beta)}\;\alpha\;\beta\;\alpha''\;\beta''!(f\circ f')\;!(g'\circ g) = (M_{\sigma(\alpha,\beta)}\;\alpha'\;\beta'\;\alpha''\;\beta''\;!f'\;!g')\circ(M_{\sigma(\alpha,\beta)}\;\alpha\;\beta\;\alpha'\;\beta'\;!f\;!g)$,
- $M_{\sigma(\alpha,\beta)}\alpha\;\beta\;\alpha\;\beta\;!id_\alpha!id_\beta = id_{\sigma(\alpha,\beta)}$.

Proof. The proof proceeds by induction over the structure of σ, and most of it is the same as in [23], except the case of tensor-types and !. These cases are essentially proved in [2]. □

Notice that in the proof of Lemma 3.18 we do not need parametricity. Suppose

$$\Xi\mid -;-\vdash f\colon\alpha'\multimap\alpha, g\colon\beta\multimap\beta'.$$

We shall write $\sigma(f,g)$ for

$$M_{\sigma(\alpha,\beta)}\alpha\;\beta\;\alpha'\;\beta'\;!f\;!g.$$

The type of $\sigma(f,g)$ is $\sigma(\alpha,\beta)\multimap\sigma(\alpha',\beta')$. Notice that we apply M to $!f,!g$, since M is of intuitionistic function type ($\to$ instead of $\multimap$). By the previous lemma, σ defines a bifunctor $\mathbf{LinType}^{op}\times\mathbf{LinType}\to\mathbf{LinType}$.

First we consider this in the case of only one argument:

Lemma 3.19 (Graph lemma). *Assuming identity extension, for any type $\alpha \vdash \sigma$ with α occuring only positively and any map $f\colon \tau \multimap \tau'$*

$$\sigma[\langle f \rangle] \equiv \langle \sigma(f) \rangle.$$

Likewise, suppose $\alpha \vdash \sigma'$ is a type with α only occuring negatively. Then identity extension implies

$$\sigma[\langle f \rangle] \equiv \langle \sigma(f) \rangle^{\mathrm{op}},$$

where $\sigma(f))\rangle^{\mathrm{op}}$ is $(x\colon \sigma(\tau), y\colon \sigma(\tau')).\, \langle \sigma(f) \rangle (y, x)$.

Proof. We will only prove the first half of the lemma; the other half is proved the same way. Since α occurs only positively in σ, we will assume for readability that M_σ has type $\prod \alpha, \beta.\, (\alpha \multimap \beta) \to \sigma(\alpha) \multimap \sigma(\beta)$. By parametricity of M_σ, for any pair of admissible relations $\rho\colon \mathsf{AdmRel}(\alpha, \alpha')$ and $\rho'\colon \mathsf{AdmRel}(\beta, \beta')$

$$(M_\sigma\, \alpha\, \beta)((\rho \multimap \rho') \to (\sigma[\rho] \multimap \sigma[\rho']))(M_\sigma\, \alpha'\, \beta'). \tag{2}$$

Let $f : \tau \multimap \tau'$ be arbitrary. If we instantiate (2) with $\rho = eq_\tau$ and $\rho' = \langle f \rangle$, we get

$$(M_\sigma\, \tau\, \tau)((eq_\tau \multimap \langle f \rangle) \to (eq_{\sigma(\tau)} \multimap \sigma[\langle f \rangle]))(M_\sigma\, \tau\, \tau'),$$

using the identity extension schema. Since $id_\tau(eq_\tau \multimap \langle f \rangle)f$,

$$!id_\tau!(eq_\tau \multimap \langle f \rangle)!f,$$

and using $M_\sigma\, \tau\, \tau'\, !f = \sigma(f)$ we get

$$id_{\sigma(\tau)}(eq_{\sigma(\tau)} \multimap \sigma[\langle f \rangle])\sigma(f),$$

i.e.,

$$\forall x\colon \sigma(\tau).\, x(\sigma[\langle f \rangle])(\sigma(f)x).$$

We have thus proved $\langle \sigma(f) \rangle$ implies $\sigma[\langle f \rangle]$.

To prove the other direction, instantiate (2) with the admissible relations $\rho = \langle f \rangle$, $\rho' = eq_{\tau'}$ for $f\colon \tau \multimap \tau'$. Since $f(\langle f \rangle \multimap eq_{\tau'})id_{\tau'}$,

$$\sigma(f)(\sigma[\langle f \rangle] \to eq_{\sigma(\tau')})id_{\sigma(\tau')}.$$

So for any $x\colon \sigma(\tau)$ and $y\colon \sigma(\tau')$ we have $x(\sigma[\langle f \rangle])y$ implies $\sigma(f)x =_{\sigma(\tau')} y$. This just means that $\sigma[\langle f \rangle]$ implies $\langle \sigma(f) \rangle$. □

3.9 Existential types

In this section we consider existential or sum types. If $\Xi, \alpha \vdash \sigma$ is a type, we define the type $\Xi \vdash \coprod \alpha.\, \sigma$ as

$$\coprod \alpha.\, \sigma = \prod \beta.\, (\prod \alpha.\, \sigma \multimap \beta) \multimap \beta$$

In fact, this defines a functor

$$\mathbf{LinType}_{\Xi,\alpha} \to \mathbf{LinType}_{\Xi}$$

with functorial action as defined in Section 3.8. In this section we show that this functor is left adjoint to the weakening functor

$$\mathbf{LinType}_{\Xi} \to \mathbf{LinType}_{\Xi,\alpha}$$

mapping a type $\Xi \vdash \sigma$ to $\Xi, \alpha \vdash \sigma$. In other words, we show that for any type $\Xi \vdash \tau$, there is a one-to-one correspondence between terms $\Xi \vdash t \colon (\coprod \alpha.\, \sigma) \multimap \tau$ and terms $\Xi, \alpha \vdash \sigma \multimap \tau$ if we consider terms up to internal equality provable using identity extension and extensionality.

First define the term

$$\mathit{pack} \colon \prod \alpha.\, (\sigma \multimap \coprod \alpha.\, \sigma)$$

as $\Lambda\alpha.\, \lambda^\circ x \colon \sigma.\, \Lambda\beta.\, \lambda^\circ f \colon \prod \alpha.\, (\sigma \multimap \beta).\, f\ \alpha\ x$. The correspondence is as follows. Suppose first $\Xi, \alpha \vdash t \colon \sigma \multimap \tau$. Then $\Xi \vdash \hat{t} \colon (\coprod \alpha.\sigma) \multimap \tau$ is $\lambda^\circ x \colon \coprod \alpha.\sigma.\, x\ \tau\ (\Lambda\alpha.\, t)$. If $\Xi \vdash s \colon (\coprod \alpha.\sigma) \multimap \tau$ then $\Xi, \alpha \vdash \hat{s} \colon \sigma \multimap \tau$ is defined to be $\lambda x \colon \sigma.\, s(\mathit{pack}\ \alpha\ x)$.

Now, suppose we start with a term $\Xi, \alpha \vdash t \colon \sigma \multimap \tau$ then

$$\begin{aligned} \hat{\hat{t}} &= \lambda^\circ x \colon \sigma.\, (\lambda^\circ y \colon \coprod \alpha.\, \sigma.\, y\ \tau\ (\Lambda\alpha.\, t))\ (\mathit{pack}\ \alpha\ x) \\ &= \lambda^\circ x \colon \sigma.\, \mathit{pack}\ \alpha\ x\ \tau\ (\Lambda\alpha.\, t) \\ &= \lambda^\circ x \colon \sigma.\, (\Lambda\alpha.\, t)\ \alpha\ x \\ &= t. \end{aligned}$$

It remains to prove that $\hat{\hat{s}}$ is equal to s for any $\Xi \vdash s \colon (\coprod \alpha.\, \sigma) \multimap \tau$. For this we need to use identity extension.

Lemma 3.20. *Suppose $x \colon \coprod \alpha.\, \sigma$, τ, τ' are types and $f \colon \tau \multimap \tau', g \colon \prod \alpha.\, \sigma \multimap \tau$. Then using identity extension and extensionality,*

$$x\ \tau'\ (\Lambda\alpha.\, f \circ (g\ \alpha)) =_{\tau'} f\ (x\ \tau\ g)$$

Proof. Using identity extension on g it is easy to see that $g(\prod \alpha.\, \sigma \multimap \langle f \rangle)\Lambda\alpha.\, f \circ (g\, \alpha)$. If $x \colon \coprod \alpha.\, \sigma$ then by identity extension

$$x\ \tau\ g \langle f \rangle x\ \tau'\ (\Lambda\alpha.\, f \circ (g\ \alpha))$$

which is what we needed to prove. □

Lemma 3.21. *It is provable using identity extension and extensionality that*

$$\forall x \colon (\coprod \alpha.\, \sigma).\, x\ \coprod \alpha.\, \sigma\ \mathit{pack} =_{\coprod \alpha.\sigma} x$$

Proof. Suppose we are given β and $f \colon \prod \alpha.\, \sigma \multimap \beta$. We show that

$$x\ \beta\ f =_\beta x\ (\coprod \alpha.\, \sigma)\ \mathit{pack}\ \beta\ f$$

Define $f' = \lambda^\circ x \colon (\coprod \alpha.\, \sigma)\ x\ \beta\ f$ of type $(\coprod \alpha.\, \sigma) \multimap \beta$. By Lemma 3.20

$$x\ \beta\ (\Lambda\alpha.\, f' \circ (\mathit{pack}\ \alpha)) =_\beta f'(x\ \coprod \alpha.\, \sigma\ \mathit{pack}) =_\beta x\ \coprod \alpha.\, \sigma\ \mathit{pack}\ \beta\ f$$

so we just need to show that $\Lambda\alpha.\, f' \circ (\mathit{pack}\ \alpha)$ is internally equal to f. But

$$\Lambda\alpha.\, f' \circ (\mathit{pack}\ \alpha)\ \alpha\ y =_\beta f'\ (\mathit{pack}\ \alpha\ y) =_\beta \mathit{pack}\ \alpha\ y\ \beta\ f =_\beta f\ \alpha\ y.$$

□

Proposition 3.22. *Suppose $\Xi \vdash s \colon (\coprod \alpha.\, \sigma) \multimap \tau$. It is provable using identity extension and extensionality that $\hat{\hat{s}}$ is internally equal to s.*

Proof.

$$\hat{\hat{s}}(x) =_\tau x\ \tau\ (\Lambda\alpha.\, \lambda^\circ x' \colon \sigma.\, s\ (\mathit{pack}\ \alpha\ x')) =_\tau s\ (x\ \coprod \alpha.\, \sigma\ \mathit{pack}) =_\tau s\ x$$

where for the second equality we have used Lemma 3.20. □

3.10 Initial algebras

Suppose $\alpha \vdash \sigma\colon$ Type is an inductively constructed type in which α occurs only positively. As we have just seen, such a type induces a functor

$$\mathbf{LinType}_\Xi \to \mathbf{LinType}_\Xi$$

for each Ξ. We aim to define an initial algebra for this type.

Define the closed type

$$\mu\alpha.\, \sigma(\alpha) = \prod \alpha.\, (\sigma(\alpha) \multimap \alpha) \to \alpha,$$

and define

$$\mathit{fold}\colon \prod \alpha.\, (\sigma(\alpha) \multimap \alpha) \to (\mu\alpha.\, \sigma(\alpha) \multimap \alpha)$$

as

$$\mathit{fold} = \Lambda\alpha.\, \lambda f\colon \sigma(\alpha) \multimap \alpha.\, \lambda^\circ u\colon \mu\alpha.\, \sigma(\alpha).\, u\,\alpha\,!f,$$

and

$$\mathit{in}\colon \sigma(\mu\alpha.\, \sigma(\alpha)) \multimap \mu\alpha.\, \sigma(\alpha)$$

as

$$\mathit{in}\, z = \Lambda\alpha.\, \lambda f\colon \sigma(\alpha) \multimap \alpha.\, f(\sigma(\mathit{fold}\,\alpha\,!f)\; z).$$

Lemma 3.23. *For any algebra $f\colon \sigma(\tau) \multimap \tau$, $\mathit{fold}\,\tau\,!f$ is a map of algebras from $(\mu\alpha.\,\sigma(\alpha), \mathit{in})$ to (τ, f), i.e., the diagram*

$$\begin{tikzcd}
\sigma(\mu\alpha.\, \sigma(\alpha)) \arrow[r, "\mathit{in}", multimap] \arrow[d, "\sigma(\mathit{fold}\,\tau\,!f)"', multimap] & \mu\alpha.\, \sigma(\alpha) \arrow[d, "\mathit{fold}\,\tau\,!f", multimap] \\
\sigma(\tau) \arrow[r, "f", multimap] & \tau
\end{tikzcd}$$

commutes.

Proof. For $x\colon \sigma(\mu\alpha.\, \sigma(\alpha))$

$$(\mathit{fold}\,\tau\,!f) \circ \mathit{in}\, x = \mathit{in}\, x\,\tau\,!f = f(\sigma(\mathit{fold}\,\tau\,!f)\; x),$$

as desired. □

In words we have shown that *in* defines a weakly initial algebra for the functor defined by σ in $\mathbf{LinType}_\Xi$ for each Ξ. Notice that parametricity was not needed in this proof.

Lemma 3.24. *Suppose $\Xi \mid \Gamma; - \vdash f\colon \sigma(\tau) \multimap \tau$ and $\Xi \mid \Gamma; - \vdash g\colon \sigma(\omega) \multimap \omega$ are algebras for σ, and $\Xi \mid \Gamma; - \vdash h\colon \tau \multimap \omega$ is a map of algebras, i.e., $h\, f = g\, \sigma(h)$. Then, assuming identity extension and extensionality,*

$$h \circ (\mathit{fold}\,\tau\,!f) =_{\mu\alpha.\sigma(\alpha)\multimap\omega} \mathit{fold}\,\omega\,!g.$$

Proof. Since h is a map of algebras

$$f(\langle\sigma(h)\rangle \multimap \langle h\rangle)g,$$

so by the Graph Lemma (3.19)

$$f(\sigma[\langle h\rangle] \multimap \langle h\rangle)g$$

and by Lemma 2.31

$$!f(!(\sigma[\langle h\rangle] \multimap \langle h\rangle))!g.$$

Clearly $(\mathit{fold},\mathit{fold}) \in \mathit{eq}_{\prod\alpha.(\sigma(\alpha)\multimap\alpha)\to(\mu\alpha.\sigma(\alpha)\multimap\alpha)}$, and thus, by identity extension,

$$(\mathit{fold},\mathit{fold}) \in \prod\alpha.\,(\sigma(\alpha) \multimap \alpha) \to (\beta \multimap \alpha)[\mathit{eq}_{\mu\alpha.\sigma(\alpha)}/\beta],$$

so for any $x\colon \mu\alpha.\,\sigma(\alpha)$,

$$(\mathit{fold}\,\tau\;!f\;x)\langle h\rangle(\mathit{fold}\,\omega\;!g\;x),$$

i.e.,

$$h \circ (\mathit{fold}\,\tau\;!f) =_{\mu\alpha.\sigma(\alpha)\multimap\omega} \mathit{fold}\,\omega\;!g,$$

as desired. □

Lemma 3.25. *Using identity extension and extensionality,*

$$\mathit{fold}\;\mu\alpha.\,\sigma(\alpha)\;!\mathit{in} =_{\mu\alpha.\sigma(\alpha)\multimap\mu\alpha.\sigma(\alpha)} \mathit{id}_{\mu\alpha.\sigma(\alpha)}.$$

Proof. By Lemma 3.24 we know that for any type τ, $f\colon \sigma(\tau) \multimap \tau$ and $u\colon \mu\alpha.\,\sigma(\alpha)$

$$(\mathit{fold}\,\tau\;!f) \circ (\mathit{fold}\;\mu\alpha.\,\sigma(\alpha)\;!\mathit{in})\;u =_\tau \mathit{fold}\,\tau\;!f\;u.$$

The left hand side of this equation becomes

$$\mathit{fold}\,\tau\;!f\;(u\;\mu\alpha.\,\sigma(\alpha)\;!\mathit{in}) = (u\;\mu\alpha.\,\sigma(\alpha)\;!\mathit{in})\tau\;!f$$

and, since the right hand side is simply

$$u\,\tau\;!f,$$

the lemma follows from Lemma 2.34. □

Theorem 3.26. *Suppose* $\Xi \mid -;- \vdash f\colon \sigma(\tau) \multimap \tau$ *is an algebra and* $\Xi \mid -;- \vdash h\colon \mu\alpha.\,\sigma(\alpha) \multimap \tau$ *is a map of algebras from in to* f*. Then if we assume identity extension and extensionality,* $h =_{\mu\alpha.\sigma(\alpha)\multimap\tau} \mathit{fold}\,\tau\;!f$*.*

Proof. By Lemma 3.24 we have

$$h \circ (\mathit{fold}\;\mu\alpha.\,\sigma(\alpha)\;!\mathit{in}) =_{\mu\alpha.\sigma(\alpha)\multimap\tau} \mathit{fold}\,\tau\;!f.$$

Lemma 3.25 finishes the job. □

We have shown that *in* defines an initial algebra.

3.11 Final Coalgebras

As in section 3.10 we will assume that $\alpha \vdash \sigma(\alpha)\colon$ Type is a type in which α occurs only positively, and this time we construct final coalgebras for the induced functor.

Define

$$\nu\alpha.\,\sigma(\alpha) = \coprod\alpha.\,!(\alpha \multimap \sigma(\alpha)) \otimes \alpha = \prod\beta.\,(\prod\alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta)) \multimap \beta$$

with combinators

$$\mathit{unfold}\colon \prod\alpha.\,(\alpha \multimap \sigma(\alpha)) \to \alpha \multimap \nu\alpha.\,\sigma(\alpha),$$
$$\mathit{out}\colon \nu\alpha.\,\sigma(\alpha) \multimap \sigma(\nu\alpha.\,\sigma(\alpha))$$

defined by

$$\begin{array}{rcl} \mathit{unfold} & = & \Lambda\alpha.\, \lambda^\circ f \colon !(\alpha \multimap \sigma(\alpha)).\, \lambda^\circ x \colon \alpha.\, \mathit{pack}\, \alpha\, (f \otimes x) \\ \mathit{out} & = & \lambda^\circ x \colon \nu\alpha.\, \sigma(\alpha).\, x\, \sigma(\nu\alpha.\, \sigma(\alpha))\, r, \end{array}$$

where

$$r : \prod \alpha.\, !(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \sigma(\nu\alpha.\, \sigma(\alpha))$$
$$r = \Lambda\alpha.\, \lambda^\circ y \colon !(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\, \text{let } w \otimes z \text{ be } y \text{ in } \sigma(\mathit{unfold}\, \alpha\, w)(\text{let } !f \text{ be } w \text{ in } f\ z).$$

Lemma 3.27. *For any coalgebra $f \colon \tau \multimap \sigma(\tau)$, the map unfold $\tau\ !f$ is a map of coalgebras from f to out.*

Proof. We need to prove that the following diagram commutes

$$\begin{tikzcd} \tau \arrow[r, "f", -o] \arrow[d, "\mathit{unfold}\, \tau\ !f"', -o] & \sigma(\tau) \arrow[d, "\sigma(\mathit{unfold}\, \tau\ !f)", -o] \\ \nu\alpha.\, \sigma(\alpha) \arrow[r, "\mathit{out}", -o] & \sigma(\nu\alpha.\, \sigma(\alpha)). \end{tikzcd}$$

But this is done by a simple computation

$$\mathit{out}(\mathit{unfold}\, \tau\ !f\ x) = \mathit{out}(\mathit{pack}\ \ \tau(!f) \otimes x) =$$
$$\mathit{pack}\, \tau(!f) \otimes x\ \sigma(\nu\alpha.\, \sigma(\alpha))\, r = r\ \tau\ ((!f) \otimes x) =$$
$$\sigma(\mathit{unfold}\, \tau\ (!f))\ (f\ x).$$

□

Lemma 3.27 shows that *out* is a weakly final coalgebra for the functor induced by σ on $\mathbf{LinType}_\Xi$ for each Ξ. Notice that parametricity was not needed here.

Lemma 3.28. *Suppose $h \colon (f \colon \tau \multimap \sigma(\tau)) \multimap (f' \colon \tau \multimap \sigma(\tau))$ is a map of coalgebras. If we assume identity extension, then the diagram*

$$\begin{tikzcd} \tau \arrow[r, "\mathit{unfold}\, \tau\ !f", -o] \arrow[d, "h"', -o] & \nu\alpha.\, \sigma(\alpha) \\ \tau' \arrow[ur, "\mathit{unfold}\, \tau'\ !f'"', -o] & \end{tikzcd}$$

commutes internally.

Proof. Using the Graph Lemma, the notion of h being a map of coalgebras can be expressed as

$$f(\langle h \rangle \multimap \sigma[\langle h \rangle])f'.$$

Now, by parametricity of *unfold*,

$$\mathit{unfold}\, \tau\ !f(\langle h \rangle \multimap \mathit{eq}_{\nu\alpha.\sigma(\alpha)})\mathit{unfold}\, \tau'\ !f',$$

which is exactly what we wanted to prove. □

Lemma 3.29. *Using extensionality and identity extension,*

$$\mathit{unfold}\, \nu\alpha.\, \sigma(\alpha)\ !\mathit{out}$$

is internally equal to the identity on $\nu\alpha.\, \sigma(\alpha)$.

Proof. Set $h = \mathit{unfold}\,\nu\alpha.\,\sigma(\alpha)\ !\mathit{out}$ in the following.

By Lemma 3.27 h is a map of coalgebras from *out* to *out*, so by Lemma 3.28, $h = h^2$. Intuitively, all we need to prove now is that h is "surjective".

Consider any $g : \prod\alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta)$. For any coalgebra map $k : (f : \alpha \multimap \sigma(\alpha)) \multimap (f' : \alpha' \multimap \sigma(\alpha'))$, we must have, by Lemmas 3.19, 2.31, and 2.29,

$$(!f \otimes x)(!(\langle k\rangle \multimap \sigma[\langle k\rangle]) \otimes \langle k\rangle)(!f' \otimes kx),$$

so by identity extension and parametricity of g,

$$\forall x\colon \alpha.\, g\,\alpha\,(!f) \otimes x =_\beta g\,\alpha'\,(!f') \otimes k(x).$$

Using this on the coalgebra map $\mathit{unfold}\,\alpha\,!f$ from f to *out* we obtain

$$\forall x\colon \alpha.\, g\,\alpha\,(!f) \otimes x =_\beta g\,\nu\alpha.\,\sigma(\alpha)\,(!\mathit{out}) \otimes \mathit{unfold}\,\alpha\,!f\ x.$$

By Lemma 2.34 this implies that

$$\forall f\colon !(\alpha \multimap \sigma(\alpha)), x\colon \alpha.\, g\,\alpha\,f \otimes x =_\beta g\,\nu\alpha.\,\sigma(\alpha)\,(!\mathit{out}) \otimes \mathit{unfold}\,\alpha\,f\ x,$$

which implies

$$\forall z\colon !(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\, g\,\alpha\,z =_\beta g\,\nu\alpha.\,\sigma(\alpha)\,(\text{let } f \otimes x \text{ be } z \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\,\alpha\,f\ x)$$

using Lemma 2.36.

In other words, if we define

$$k\colon \prod\alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \tau),$$

where $\tau = !(\nu\alpha.\,\sigma(\alpha) \multimap \sigma(\nu\alpha.\,\sigma(\alpha))) \otimes \nu\alpha.\,\sigma(\alpha)$, to be

$$k = \Lambda\alpha.\,\lambda^\circ y :!(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\,\text{let } f \otimes x \text{ be } y \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\,\alpha\,f\ x,$$

then

$$\forall\alpha.\, g\,\alpha =_{!(\alpha\multimap\sigma(\alpha))\otimes\alpha\multimap\beta} (g\,\nu\alpha.\,\sigma(\alpha)) \circ (k\,\alpha). \tag{3}$$

Now, suppose we are given $\alpha, \alpha', R\colon \mathsf{Rel}(\alpha, \alpha')$ and terms f, f' such that

$$f(!(R \multimap \sigma[R]) \otimes R)f'.$$

Then, by (3) and parametricity of g

$$g\,\alpha\,f =_\beta g\,\alpha'\,f' =_\beta (g\,\nu\alpha.\,\sigma(\alpha))(k\,\alpha'\,f'),$$

from which we conclude

$$g(\forall(\alpha, \beta, R\colon \mathsf{Rel}(\alpha, \beta)).\,(!(R \multimap \sigma[R]) \otimes R \multimap \langle g\,\nu\alpha.\,\sigma(\alpha)\rangle^{op}))k.$$

(Here we use S^{op} for the inverse relation of S.) Using parametricity, this implies that, for any $x\colon \nu\alpha.\,\sigma(\alpha)$, we have

$$x\,\beta\,g =_\beta g\,\nu\alpha.\,\sigma(\alpha)\,(x\,\tau\,k).$$

Thus, since g was arbitrary, we may apply the above to $g = k$ and get

$$x\ \tau\ k =_{\tau} k\ \nu\alpha.\, \sigma(\alpha)\ (x\ \tau\ k) = \text{let } f \otimes z \text{ be } (x\ \tau\ k) \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ f\ z.$$

If we write

$$l = \lambda x \colon \nu\alpha.\, \sigma(\alpha).\, \text{let } f \otimes z \text{ be } (x\ \tau\ k) \text{ in } \mathit{unfold}\ \alpha\ f\ z,$$

then, since k is a closed term, so is l, and from the above calculations we conclude that we have

$$\forall \beta.\, \forall g : \textstyle\prod \alpha.\, !(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta.\, x\ \beta\ g =_{\beta} g\ \nu\alpha.\, \sigma(\alpha)\ (!\mathit{out}) \otimes (l\ x).$$

Now, finally,

$$\begin{gathered}
h(l\ x) = \mathit{unfold}\ \nu\alpha.\, \sigma(\alpha)\ !\mathit{out}\ (l\ x) = \\
\mathit{pack}\ \nu\alpha.\, \sigma(\alpha)\ !\mathit{out} \otimes (l\ x) = \\
\Lambda\beta.\, \lambda g : \textstyle\prod \alpha.\, (!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta).\, g\ \nu\alpha.\, \sigma(\alpha)\ !\mathit{out} \otimes (l\ x) =_{\nu\alpha.\sigma(\alpha)} \\
\Lambda\beta.\, \lambda g : \textstyle\prod \alpha.\, (!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta).\, x\ \beta\ g = x,
\end{gathered}$$

where we have used extensionality. Thus l is a right inverse to h, and we conclude

$$h\ x =_{\nu\alpha.\sigma(\alpha)} h^2(l\ x) =_{\nu\alpha.\sigma(\alpha)} h(l\ x) =_{\nu\alpha.\sigma(\alpha)} x.$$

□

Theorem 3.30. *Suppose* $\Xi \mid -; - \vdash f \colon \tau \multimap \sigma(\tau)$ *is a coalgebra and* $\Xi \mid -; - \vdash h \colon \tau \multimap \mu\alpha.\, \sigma(\alpha)$ *is a map of algebras from* f *to out. Then if we assume identity extension and extensionality* $h =_{\tau \multimap \mu\alpha.\sigma(\alpha)}$ *unfold* $\alpha\ !f$.

Proof. Consider a map of coalgebras into *out*:

$$\begin{tikzcd}
\tau \arrow[r, "f", -o] \arrow[d, "h", -o] & \sigma(\tau) \arrow[d, "\sigma(h)", -o] \\
\nu\alpha.\, \sigma(\alpha) \arrow[r, "\mathit{out}", -o] & \sigma(\nu\alpha.\, \sigma(\alpha)).
\end{tikzcd}$$

By Lemmas 3.28 and 3.29,

$$\mathit{unfold}\ \tau\ !f =_{\tau \multimap \nu\alpha.\sigma(\alpha)} (\mathit{unfold}\ \nu\alpha.\, \sigma(\alpha)\ !\mathit{out}) \circ g =_{\tau \multimap \nu\alpha.\sigma(\alpha)} g.$$

□

Theorem 3.30 shows that *out* is a final coalgebra for the endofunctor on $\mathbf{LinType}_{\Xi}$ induced by σ for each Ξ.

3.12 Recursive type equations

In this section we consider inductively constructed types $\alpha \vdash \sigma(\alpha)$ and construct closed types τ such that $\sigma(\tau) \cong \tau$. In Sections 3.10 and 3.11 we solved the problem in the special case of α occuring only positively in σ, by finding initial algebras and final coalgebras for the functor induced by σ.

This section details the sketch of [22], but the theory is due to Freyd [8, 7, 9]. In short, the main observation is that because of the presense of fixed points, the initial algebras and final coalgebras of Sections 3.10, 3.11 coincide (Theorem 3.36 below). This phenomenon is called compactness, and was studied by Freyd in *loc. cit.*.

Before we start, observe that we may split the occurences of α in σ in positive and negative occurences. So our standard assumption in this section is that we are given a type $\alpha, \beta \vdash \sigma(\alpha, \beta)$, in which α occurs only negatively and β only positively, and we look for a type τ, such that $\sigma(\tau, \tau) \cong \tau$.

3.12.1 Parametrized initial algebras

Set $\omega(\alpha) = \mu\beta.\, \sigma(\alpha, \beta) = \prod \beta.\, (\sigma(\alpha, \beta) \multimap \beta) \multimap \beta$. Now, ω induces a contravariant functor from types to types.

Lemma 3.31. *Assuming identity extension and extensionality, for $f \colon \alpha' \multimap \alpha$, $\omega(f) \colon \omega(\alpha) \multimap \omega(\alpha')$ is (up to internal equality) the unique h such that*

$$\begin{tikzcd}
\sigma(\alpha, \omega(\alpha)) \arrow[r, "\mathit{in}", -o] \arrow[d, "\sigma(id,h)"', -o] & \omega(\alpha) \arrow[dd, "h", -o] \\
\sigma(\alpha, \omega(\alpha')) \arrow[d, "\sigma(f,id)"', -o] & \\
\sigma(\alpha', \omega(\alpha')) \arrow[r, "\mathit{in}", -o] & \omega(\alpha')
\end{tikzcd}$$

commutes internally.

Proof. One may define *in* as a polymorphic term

$$\mathit{in} \colon \prod \alpha.\, \sigma(\alpha, \omega(\alpha)) \multimap \omega(\alpha)$$

by

$$\mathit{in} = \Lambda\alpha.\, \lambda^{\circ} z \colon \sigma(\alpha, \omega(\alpha)).\, \Lambda\beta.\, \lambda f \colon \sigma(\alpha, \beta) \multimap \beta.\, f(\sigma(\lambda x \colon \alpha.\, x, \mathit{fold}\, \beta\, !f)\; z).$$

By parametricity we have

$$\mathit{in}\, \alpha'(\sigma(\langle f \rangle, \omega(\langle f \rangle)) \multimap \omega(\langle f \rangle))\mathit{in}\, \alpha,$$

which, by the Graph Lemma (Lemma 3.19), means that

$$\mathit{in}\, \alpha'(\langle \sigma(f, \omega(f)) \rangle^{\mathrm{op}} \multimap \langle \omega(f) \rangle^{\mathrm{op}})\mathit{in}\, \alpha,$$

which in turn amounts to internal commutativity of the diagram of the lemma.

Uniqueness is by initiality of *in* (in $\mathbf{LinType}_{\alpha}$, proved as before) used on the diagram

$$\begin{tikzcd}
\sigma(\alpha, \omega(\alpha)) \arrow[rr, "\mathit{in}", -o] \arrow[d, "\sigma(id,h)"', -o] & & \omega(\alpha) \arrow[d, "h", -o] \\
\sigma(\alpha, \omega(\alpha')) \arrow[r, "\sigma(f,id)", -o] & \sigma(\alpha', \omega(\alpha')) \arrow[r, "\mathit{in}", -o] & \omega(\alpha').
\end{tikzcd}$$

$\square$

3.12.2 Dialgebras

Definition 3.32. A dialgebra for σ is a quadruple (τ, τ', f, f') such that τ and τ' are types, and $f\colon \sigma(\tau', \tau) \multimap \tau$ and $f'\colon \tau' \multimap \sigma(\tau, \tau')$ are morphisms. A morphism of dialgebras from $(\tau_0, \tau_0', f_0, f_0')$ to $(\tau_1, \tau_1', f_1, f_1')$ is a pair of morphisms $h\colon \tau_0 \multimap \tau_1$, $h'\colon \tau_1' \multimap \tau_0'$, such that

$$\begin{tikzcd}
\sigma(\tau_0', \tau_0) \arrow[r, multimap, "f_0"] \arrow[d, multimap, "\sigma(h', h)"'] & \tau_0 \arrow[d, multimap, "h"] & \tau_1' \arrow[r, multimap, "f_1'"] \arrow[d, multimap, "h'"'] & \sigma(\tau_1, \tau_1') \arrow[d, multimap, "\sigma(h, h')"] \\
\sigma(\tau_1', \tau_1) \arrow[r, multimap, "f_1"'] & \tau_1 & \tau_0' \arrow[r, multimap, "f_0'"'] & \sigma(\tau_0, \tau_0').
\end{tikzcd}$$

Lemma 3.33. *If (h, h') is a map of dialgebras and h, h' are isomorphisms, then (h, h') is an isomorphism of dialgebras.*

Proof. The only thing to prove here is that $(h^{-1}, (h')^{-1})$ is in fact a map of dialgebras, which is trivial. □

Remark 3.34. If we for the type $\alpha, \beta \vdash \sigma\colon \mathsf{Type}$ consider the endofunctor

$$\langle \sigma^{\mathrm{op}}, \sigma \rangle \colon \mathbf{LinType^{op}}_{\Xi} \times \mathbf{LinType}_{\Xi} \to \mathbf{LinType^{op}}_{\Xi} \times \mathbf{LinType}_{\Xi}$$

defined by $(\alpha, \beta) \mapsto (\sigma(\beta, \alpha), \sigma(\alpha, \beta))$, then dialgebras for σ are exactly the algebras for $\langle \sigma^{\mathrm{op}}, \sigma \rangle$, maps of dialgebras are maps of algebras for $\langle \sigma^{\mathrm{op}}, \sigma \rangle$ and initial dialgebras correspond to initial algebras.

Theorem 3.35. *Assuming identity extension and extensionality, initial dialgebras exist for all functors induced by types $\sigma(\alpha, \beta)$, up to internal equality.*

Proof. In this proof, commutativity of diagrams will mean commutativity up to internal equality.

Set $\omega(\alpha) = \mu\beta.\, \sigma(\alpha, \beta)$. Then, ω defines a contravariant functor. Define

$$\tau' = \nu\alpha.\, \sigma(\omega(\alpha), \alpha), \qquad \tau = \omega(\tau') = \mu\beta.\, \sigma(\tau', \beta).$$

Since τ' is defined as the final coalgebra for a functor, we have a morphism

$$\mathit{out}\colon \tau' \multimap \sigma(\omega(\tau'), \tau') = \sigma(\tau, \tau'),$$

and since τ is defined to be an initial algebra, we get a morphism

$$\mathit{in}\colon \sigma(\tau', \tau) \multimap \tau.$$

We will show that $(\tau, \tau', \mathit{in}, \mathit{out})$ is an initial dialgebra.

Suppose we are given a dialgebra (τ_0, τ_0', g, g'). Since *in* is an initial algebra, there exists a unique map a, such that

$$\begin{tikzcd}
\sigma(\tau_0', \omega(\tau_0')) \arrow[r, multimap, "\mathit{in}"] \arrow[d, multimap, "\sigma(id, a)"'] & \omega(\tau_0') \arrow[d, multimap, "a"] \\
\sigma(\tau_0', \tau_0) \arrow[r, multimap, "g"] & \tau_0,
\end{tikzcd}$$

and thus, since *out* is a final coalgebra, we find a map h' making the diagram

$$\begin{tikzcd}
\tau'_0 \arrow[r, -o, "g'"] \arrow[d, -o, "h'"'] & \sigma(\tau_0, \tau'_0) \arrow[r, -o, "\sigma(a,id)"] & \sigma(\omega(\tau'_0), \tau'_0) \arrow[d, -o, "\sigma(\omega(h'),h')"] \\
\tau' \arrow[rr, -o, "\mathit{out}"] & & \sigma(\omega(\tau'), \tau')
\end{tikzcd} \tag{4}$$

commute. Set $h = a \circ \omega(h')$. We claim that (h, h') defines a map of dialgebras. The second diagram of Definition 3.32 is simply (4). The first diagram of 3.32 follows from the commutativity of the composite diagram

$$\begin{tikzcd}
\sigma(\tau', \omega(\tau')) \arrow[r, -o, "\mathit{in}"] \arrow[d, -o, "\sigma(h',\omega(h'))"'] & \omega(\tau') \arrow[d, -o, "\omega(h')"] \\
\sigma(\tau'_0, \omega(\tau'_0)) \arrow[r, -o, "\mathit{in}"] \arrow[d, -o, "\sigma(id,a)"'] & \omega(\tau'_0) \arrow[d, -o, "a"] \\
\sigma(\tau'_0, \tau_0) \arrow[r, -o, "g"] & \tau_0,
\end{tikzcd} \tag{5}$$

where the top diagram commutes by Lemma 3.31.

Finally, we will prove that (h, h') is the unique dialgebra morphism. Suppose we are given a map of dialgebras (k, k') from (τ, τ', in, out) to (τ_0, τ'_0, g, g'). By the first diagram of Definition 3.32, we have a commutative diagram

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[rr, -o, "\mathit{in}"] \arrow[d, -o, "\sigma(id,k)"'] & & \tau \arrow[d, -o, "k"] \\
\sigma(\tau', \tau_0) \arrow[r, -o, "\sigma(k',id)"] & \sigma(\tau'_0, \tau_0) \arrow[r, -o, "g"] & \tau_0.
\end{tikzcd}$$

Since clearly (5) also commutes when k' is substituted for h', by (strong) initiality of *in*, we conclude that $k =_{\tau \multimap \tau'} a \circ \omega(k')$. Finally, by the second diagram of Definition 3.32 we have commutativity of

$$\begin{tikzcd}
\tau'_0 \arrow[r, -o, "g'"] \arrow[d, -o, "k'"'] & \sigma(\tau_0, \tau'_0) \arrow[r, -o, "\sigma(a,id)"] & \sigma(\omega(\tau'_0), \tau'_0) \arrow[d, -o, "\sigma(\omega(k'),k')"] \\
\tau' \arrow[rr, -o, "\mathit{out}"] & & \sigma(\omega(\tau'), \tau').
\end{tikzcd}$$

So since *out* is a final coalgebra we conclude $k' =_{\tau'_0 \multimap \tau'} h'$. □

3.12.3 Compactness

As advertised in the introduction to this section, the presence of fixed points makes initial algebras and final coalgebras coincide.

Theorem 3.36 (Compactness). *Assuming identity extension and extensionality, for all types $\alpha \vdash \sigma(\alpha)$ in which α occurs only positively, in^{-1} is internally a final coalgebra and out^{-1} is internally an initial algebra. Furthermore in^{-1} and out^{-1} can be written as terms of PILL$_Y$.*

Proof. By Theorems 3.26 and 3.30 *in* is an initial algebra, and *out* is a final coalgebra for σ. Consider

$$h = Y\ (\nu\alpha.\,\sigma(\alpha)) \multimap \mu\alpha.\,\sigma(\alpha)\ (\lambda h\colon \nu\alpha.\,\sigma(\alpha) \multimap \mu\alpha.\,\sigma(\alpha).\ \mathit{in} \circ \sigma(h) \circ \mathit{out}).$$

Since Y is a fixed-point operator, we know that

$$\begin{tikzcd}
\sigma(\nu\alpha.\,\sigma(\alpha)) \arrow[d, "\sigma(h)"', -o] & \nu\alpha.\,\sigma(\alpha) \arrow[l, "\mathit{out}"', -o] \arrow[d, "h", -o] \\
\sigma(\mu\alpha.\,\sigma(\alpha)) \arrow[r, "\mathit{in}", -o] & \mu\alpha.\,\sigma(\alpha)
\end{tikzcd}$$

commutes. Since in^{-1} is a coalgebra, we also have a map k going the other way, and since *out* is a final coalgebra, $kh =_{\nu\alpha.\sigma(\alpha) \multimap \nu\alpha.\sigma(\alpha)} id_{\nu\alpha.\sigma(\alpha)}$. Since *in* is an initial algebra, we know that $hk =_{\mu\alpha.\sigma(\alpha) \multimap \mu\alpha.\sigma(\alpha)} id_{\mu\alpha.\sigma(\alpha)}$. So $\mathit{in}^{-1} \cong \mathit{out}$ as coalgebras and $\mathit{out}^{-1} \cong \mathit{in}$ as algebras, internally. □

Lemma 3.37. *Assume identity extension and extensionality. Let* $(\tau, \tau', \mathit{in}, \mathit{out})$ *be the initial dialgebra from the proof of Theorem 3.35. Then* $(\tau', \tau, \mathit{out}^{-1}, \mathit{in}^{-1})$ *is also an initial dialgebra internally.*

Proof. In this proof, commutativity of diagrams is up to internal equality.

Suppose we are given a dialgebra (τ_0, τ_0', g, g'). We will show that there exists a unique morphism of dialgebras from $(\tau', \tau, \mathit{out}^{-1}, \mathit{in}^{-1})$ to (τ_0, τ_0', g, g').

By Theorem 3.36, for all types α, $\mathit{in}^{-1}\colon \omega(\alpha) \multimap \sigma(\alpha, \omega(\alpha))$ is a final coalgebra for the functor $\beta \mapsto \sigma(\alpha, \beta)$, and $\mathit{out}^{-1}\colon \sigma(\tau, \tau') \multimap \tau'$ is an initial algebra for the functor $\alpha \mapsto \sigma(\omega(\alpha), \alpha)$.

Let a be the unique map making the diagram

$$\begin{tikzcd}
\tau_0' \arrow[r, "g'", -o] \arrow[d, "a"', -o] & \sigma(\tau_0, \tau_0') \arrow[d, "\sigma(id{,}a)", -o] \\
\omega(\tau_0) \arrow[r, "\mathit{in}^{-1}", -o] & \sigma(\tau_0, \omega(\tau_0))
\end{tikzcd}$$

commute. Define h to be the unique map making

$$\begin{tikzcd}
\sigma(\tau, \tau') \arrow[rr, "\mathit{out}^{-1}", -o] \arrow[d, "\sigma(\omega(h){,}h)"', -o] & & \tau' \arrow[d, "h", -o] \\
\sigma(\omega(\tau_0), \tau_0) \arrow[r, "\sigma(a{,}id)", -o] & \sigma(\tau_0', \tau_0) \arrow[r, "g", -o] & \tau_0
\end{tikzcd} \tag{6}$$

commute. We define h' to be $\omega(h) \circ a$ and prove that (h, h') is a map of dialgebras. The first diagram of Definition 3.32 is simply (6). Commutativity of the second diagram follows from commutativity of

$$\begin{tikzcd}
\tau_0' \arrow[r, "g'", -o] \arrow[d, "a"', -o] & \sigma(\tau_0, \tau_0') \arrow[d, "\sigma(id{,}a)", -o] \\
\omega(\tau_0) \arrow[r, "\mathit{in}^{-1}", -o] \arrow[d, "\omega(h)"', -o] & \sigma(\tau_0, \omega(\tau_0)) \arrow[d, "\sigma(h{,}\omega(h))", -o] \\
\omega(\tau') \arrow[r, "\mathit{in}^{-1}", -o] & \sigma(\tau', \omega(\tau')),
\end{tikzcd} \tag{7}$$

where commutativity of the last diagram follows from Lemma 3.31.

Finally, we will show that if (k, k') is another map of dialgebras from $(\tau', \tau, \mathit{out}^{-1}, \mathit{in}^{-1})$ to (τ_0, τ'_0, g, g') then $h =_{\tau' \multimap \tau_0} k$ and $h' =_{\tau'_0 \multimap \tau} k'$. By the second diagram of Definition 3.32 we know that

$$\begin{tikzcd}
\tau'_0 \arrow[r, "g'", -o] \arrow[d, "k'"', -o] & \sigma(\tau_0, \tau'_0) \arrow[r, "\sigma(k,id)", -o] & \sigma(\tau', \tau'_0) \arrow[d, "\sigma(id,k')", -o] \\
\tau \arrow[rr, "\mathit{in}^{-1}", -o] & & \sigma(\tau', \tau)
\end{tikzcd} \tag{8}$$

commutes. Clearly, if we substitute k for h in (7), we obtain a diagram that commutes by Lemma 3.31. So, using the fact that in^{-1} is a final coalgebra on (8), we get $k' =_{\tau'_0 \multimap \tau} \omega(k) \circ a$.

The first diagram of Definition 3.32 implies that

$$\begin{tikzcd}
\sigma(\tau, \tau') \arrow[rr, "\mathit{out}^{-1}", -o] \arrow[d, "\sigma(\omega(k),k)"', -o] & & \tau' \arrow[d, "k", -o] \\
\sigma(\omega(\tau_0), \tau_0) \arrow[r, "\sigma(a,id)", -o] & \sigma(\tau'_0, \tau_0) \arrow[r, "g", -o] & \tau_0
\end{tikzcd}$$

commutes. Comparing this to (6) we obtain $h =_{\tau' \multimap \tau_0} k$, by initiality of out^{-1}. □

Theorem 3.38. *Assuming identity extension and extensionality, for all types $\sigma(\alpha, \beta)$ where α occurs only negatively and β only positively, there exists a type τ and a map $f \colon \sigma(\tau, \tau) \multimap \tau$, such that (τ, τ, f, f^{-1}) is an initial dialgebra up to internal equality.*

Proof. As usual commutativity of diagrams will be up to internal equality.

We have a unique map of dialgebras

$$(h, h') \colon (\tau, \tau', \mathit{in}, \mathit{out}) \to (\tau', \tau, \mathit{out}^{-1}, \mathit{in}^{-1})$$

We claim that (h', h) is also a map of dialgebras from $(\tau, \tau', \mathit{in}, \mathit{out})$ to $(\tau', \tau, \mathit{out}^{-1}, \mathit{in}^{-1})$. To prove this we need to prove commutativity of the diagrams

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[r, "\mathit{in}", -o] \arrow[d, "\sigma(h,h')"', -o] & \tau \arrow[d, "h'", -o] & \tau \arrow[r, "\mathit{in}^{-1}", -o] \arrow[d, "h"', -o] & \sigma(\tau', \tau) \arrow[d, "\sigma(h',h)", -o] \\
\sigma(\tau, \tau') \arrow[r, "\mathit{out}^{-1}", -o] & \tau' & \tau' \arrow[r, "\mathit{out}", -o] & \sigma(\tau, \tau')
\end{tikzcd} \quad ,$$

but the fact that (h, h') is a map of dialgebras tells us exactly that

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[r, "\mathit{in}", -o] \arrow[d, "\sigma(h',h)"', -o] & \tau \arrow[d, "h", -o] & \tau \arrow[r, "\mathit{in}^{-1}", -o] \arrow[d, "h'"', -o] & \sigma(\tau', \tau) \arrow[d, "\sigma(h,h')", -o] \\
\sigma(\tau, \tau') \arrow[r, "\mathit{out}^{-1}", -o] & \tau' & \tau' \arrow[r, "\mathit{out}", -o] & \sigma(\tau, \tau'),
\end{tikzcd}$$

and these two diagram are the same as the above but in opposite order. Thus, by uniqueness of maps of dialgebras out of $(\tau, \tau', \mathit{in}, \mathit{out})$, we get $h =_{\tau \multimap \tau'} h'$. Since (h, h) is a map between initial dialgebras, h is an isomorphism.

Now define $f\colon \sigma(\tau,\tau) \multimap \tau$ to be *in* $\circ\, \sigma(h^{-1}, id_\tau)$. Then clearly (id_τ, h^{-1}) is a morphism of dialgebras from (τ,τ,f,f^{-1}) to $(\tau,\tau',\mathit{in},\mathit{out})$, since the diagrams proving (id_τ, h^{-1}) to be a map of dialgebras are

$$\begin{tikzcd}
\sigma(\tau,\tau) \arrow[r, "\sigma(h^{-1},id)"] \arrow[rr, bend right, "f"'] \arrow[d, "\sigma(h^{-1},id)"'] & \sigma(\tau',\tau) \arrow[r, "in"] & \tau \arrow[d, "id"] \\
\sigma(\tau',\tau) \arrow[rr, "in"] & & \tau
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau' \arrow[rr, "out"] \arrow[d, "h^{-1}"'] & & \sigma(\tau,\tau') \arrow[d, "\sigma(id,h^{-1})"] \\
\tau \arrow[r, "in^{-1}"] \arrow[rr, bend right, "f^{-1}"'] & \sigma(\tau',\tau) \arrow[r, "\sigma(h,id)"] & \sigma(\tau,\tau).
\end{tikzcd}$$

Clearly the first diagram commutes, and the second diagram is just part of the definition of (h,h) being a map of dialgebras. Thus (id_τ, h^{-1}) defines an isomorphism of dialgebras from (τ,τ,f,f^{-1}) to $(\tau,\tau',\mathit{in},\mathit{out})$, as desired. □

Corollary 3.39. *Assuming identity extension and extensionality, for all types $\alpha,\beta \vdash \sigma(\alpha,\beta)$, where α occurs only negatively and β only positively, there exists a type τ such that $\sigma(\tau,\tau) \cong \tau$ in each $\mathbf{LinType}_\Xi$.*

Proof. The isomorphism is *in* $\circ\, \sigma(h^{-1}, id)$. □

Notice that the closed terms $\tau \multimap \sigma(\tau,\tau)$ and $\sigma(\tau,\tau) \multimap \tau$ always exist, independent of the assumption of parametricity. We use parametricity to prove that they are each others inverses.

3.13 Recursive type equations with parameters

We now consider recursive type equations with parameters, i.e., we consider types $\vec\alpha,\alpha \vdash \sigma(\vec\alpha,\alpha)$ and look for types $\vec\alpha \vdash \tau(\vec\alpha)$ satisfying $\sigma(\vec\alpha,\tau(\vec\alpha)) \cong \tau(\vec\alpha)$. As before, we need to split occurences of the variable α into positive and negative occurences, and since we would like to be able to construct nested recursive types, we need to keep track of positive and negative occurences of the variables $\vec\alpha$ in the solution τ as well. So we will suppose that we are given a type $\vec\alpha,\vec\beta,\alpha,\beta \vdash \sigma(\vec\alpha,\vec\beta,\alpha,\beta)$ in which the variables $\vec\alpha,\alpha$ occur only negatively and the variables $\vec\beta,\beta$ only positively.

Of course, the proof proceeds as in the case without parameters. However, one must take care to obtain the right occurences of parameters, and so we sketch the proof here.

Lemma 3.40. *Suppose $\vec\alpha,\vec\beta,\alpha,\beta \vdash \sigma(\vec\alpha,\vec\beta,\alpha,\beta)$ is a type in which the variables $\vec\alpha,\alpha$ occur only negatively and the variables $\vec\beta,\beta$ only positively. There exists types $\vec\alpha,\vec\beta \vdash \tau(\vec\alpha,\vec\beta)$ in which $\vec\alpha$ occurs only negatively and $\vec\beta$ only positively and $\vec\alpha,\vec\beta \vdash \tau'(\vec\alpha,\vec\beta)$ in which $\vec\alpha$ occurs only positively and $\vec\beta$ only negatively and terms*

$$\begin{aligned} \mathit{in}\colon \sigma(\vec\alpha,\vec\beta,\tau'(\vec\alpha,\vec\beta),\tau(\vec\alpha,\vec\beta)) &\multimap \tau(\vec\alpha,\vec\beta) \\ \mathit{out}\colon \tau'(\vec\alpha,\vec\beta) &\multimap \sigma(\vec\beta,\vec\alpha,\tau(\vec\alpha,\vec\beta),\tau'(\vec\alpha,\vec\beta)) \end{aligned}$$

such that for any pair of types $\vec\alpha,\vec\beta \vdash \omega,\omega'$, and terms

$$\begin{aligned} g\colon \sigma(\vec\alpha,\vec\beta,\omega',\omega) &\multimap \omega \\ g'\colon \omega' &\multimap \sigma(\vec\beta,\vec\alpha,\omega,\omega') \end{aligned}$$

there exists unique h, h' making

$$\begin{tikzcd}
\sigma(\vec\alpha,\vec\beta,\tau'(\vec\alpha,\vec\beta),\tau(\vec\alpha,\vec\beta)) \arrow[r, "in"] \arrow[d, "\sigma(\vec\alpha,\vec\beta,h',h)"'] & \tau(\vec\alpha,\vec\beta) \arrow[d, "h"] \\
\sigma(\vec\alpha,\vec\beta,\omega',\omega) \arrow[r, "g"] & \omega
\end{tikzcd}
\qquad
\begin{tikzcd}
\omega' \arrow[r, "g'"] \arrow[d, "h'"'] & \sigma(\vec\beta,\vec\alpha,\omega,\omega') \arrow[d, "\sigma(\vec\beta,\vec\alpha,h,h')"] \\
\tau'(\vec\alpha,\vec\beta) \arrow[r, "out"] & \sigma(\vec\beta,\vec\alpha,\tau(\vec\alpha,\vec\beta),\tau'(\vec\alpha,\vec\beta))
\end{tikzcd}$$

commute up to internal equality.

Proof. Define

$$\begin{array}{rcl} \omega(\vec{\alpha},\vec{\beta},\alpha) & = & \mu\beta.\,\sigma(\vec{\alpha},\vec{\beta},\alpha,\beta) \\ \tau'(\vec{\alpha},\vec{\beta}) & = & \nu\alpha.\,\sigma(\vec{\beta},\vec{\alpha},\omega(\vec{\alpha},\vec{\beta},\alpha),\alpha) \\ \tau(\vec{\alpha},\vec{\beta}) & = & \omega(\vec{\alpha},\vec{\beta},\tau'(\vec{\alpha},\vec{\beta})) \end{array}$$

Notice that we have swapped the occurences of $\vec{\alpha},\vec{\beta}$ in σ in the definition of τ', making all occurences of $\vec{\alpha}$ in τ' positive and all occurences of $\vec{\beta}$ in τ' negative. The rest of the proof proceeds exactly as the proof of Theorem 3.35. □

Theorem 3.41. *Suppose $\vec{\alpha},\vec{\beta},\alpha,\beta \vdash \sigma(\vec{\alpha},\vec{\beta},\alpha,\beta)$ is a type as in Lemma 3.40. Then there exists a type $\tau(\vec{\alpha},\vec{\beta})$ with $\vec{\alpha}$ occuring only negatively and $\vec{\beta}$ only positively, and a term*

$$\mathit{in}\colon \sigma(\vec{\alpha},\vec{\beta},\tau(\vec{\beta},\vec{\alpha}),\tau(\vec{\alpha},\vec{\beta})) \multimap \tau(\vec{\alpha},\vec{\beta})$$

satisfying the conclusion of Lemma 3.41 with $\tau'(\vec{\alpha},\vec{\beta}) = \tau(\vec{\beta},\vec{\alpha})$ and

$$\mathit{out} = \mathit{in}^{-1}\colon \tau(\beta,\alpha) \multimap \sigma(\vec{\beta},\vec{\alpha},\tau(\vec{\alpha},\vec{\beta}),\tau(\vec{\beta},\vec{\alpha})).$$

Proof. Using Theorem 3.36, we can prove as in the proof of Lemma 3.37 that the pair

$$\begin{array}{c} \mathit{out}^{-1}\colon \sigma(\vec{\alpha},\vec{\beta},\tau(\vec{\beta},\vec{\alpha}),\tau'(\vec{\beta},\vec{\alpha})) \multimap \tau'(\vec{\beta},\vec{\alpha}) \\ \mathit{in}^{-1}\colon \tau(\vec{\beta},\vec{\alpha}) \multimap \sigma(\vec{\beta},\vec{\alpha},\tau'(\vec{\beta},\vec{\alpha}),\tau(\vec{\beta},\vec{\alpha})) \end{array}$$

also satisfies the conclusion of Lemma 3.41. Proceeding as in the proof of Lemma 3.38 we get an isomorphism $\tau(\vec{\alpha},\vec{\beta}) \cong \tau'(\vec{\beta},\vec{\alpha})$ up to internal equality, which implies the theorem. □

Corollary 3.42. *For any type $\vec{\alpha},\vec{\beta},\alpha,\beta \vdash \sigma(\vec{\alpha},\vec{\beta},\alpha,\beta)$ is a type as in Lemma 3.40, there exist a type $\tau(\vec{\alpha},\vec{\beta})$ with $\vec{\alpha}$ occuring only negatively and $\vec{\beta}$ only positively and an isomorphism*

$$\sigma(\vec{\alpha},\vec{\beta},\tau(\vec{\beta},\vec{\alpha}),\tau(\vec{\alpha},\vec{\beta})) \cong \tau(\vec{\alpha},\vec{\beta})$$

in $\mathbf{LinType}_{\vec{\alpha},\vec{\beta}}$.

4 LAPL-structures

In this section we introduce the notion of LAPL-structure. An LAPL-structure is a model of LAPL.

First, however, we call to mind what a model of PILL is and how PILL is interpreted in such a model (for a full description of models for PILL and interpretations in these, see e.g. [19, 17, 2, 15, 5]).

A model of PILL is a fibred symmetric monoidal adjunction

$$\begin{tikzcd} \mathbf{LinType} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr, "p"'] & & \mathbf{Type} \arrow[ll, bend right, "F"'] \arrow[dl] \\ & \mathbf{Kind}, & \end{tikzcd}$$

such that $\mathbf{LinType}$ is fibred symmetric monoidal closed; the tensor in $\mathbf{Type}$ is a fibred cartesian product; $\mathbf{Type}$ is equivalent to the category of finite products of free coalgebras for the comonad FG on $\mathbf{LinType}$;

$\mathbf{Kind}$ is cartesian; p has a generic object and simple products with respect to projections forgetting Ω, where Ω is p of the generic object. See [17] for detailed explanation of this definition.

PILL is interpreted in such models as follows. A type σ is interpreted as an object $[\![\sigma]\!] \in \mathbf{LinType}$ using the SMCC structure to interpret $\otimes, \multimap, I$ and the comonad FG to interpret !, and we interpret a term

$$\vec{\alpha} \mid \vec{x} \colon \vec{\sigma}; \vec{x}' \colon \vec{\sigma}' \vdash t \colon \tau$$

as a morphism

$$![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_n]\!] \otimes [\![\sigma'_1]\!] \ \otimes \ldots \otimes [\![\sigma'_m]\!] \multimap [\![\tau]\!]$$

in $\mathbf{LinType}$, where $! = FG$. Notice that we denote the morphisms in $\mathbf{LinType}$ by $\multimap$.

The comonad structure on $\mathbf{LinType}$ induced by the adjunction gives us two natural transformations $\delta \colon ! \multimap !!$ and $\epsilon \colon ! \multimap id$. These are defined in the internal language as

$$\begin{aligned} \delta_\sigma &= \lambda^\circ x \colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } !!y, \\ \epsilon_\sigma &= \lambda^\circ x \colon !\sigma.\, \text{let } !z \text{ be } x \text{ in } z. \end{aligned}$$

It turns out that we may interpret the intuitionistic part of the calculus, that is, the terms in the calculus with no free linear variables, in $\mathbf{Type}$. For suppose we are given such a term

$$\Xi \mid \vec{x} \colon \vec{\sigma}; - \vdash t \colon \tau.$$

Then the interpretation of this term in $\mathbf{LinType}$ is

$$[\![\Xi \mid \vec{x} \colon \vec{\sigma}; - \vdash t \colon \tau]\!] \colon \otimes_i ![\![\Xi \mid \sigma_i]\!] \multimap [\![\Xi \mid \tau]\!].$$

Since $\otimes_i ![\![\Xi \mid \sigma_i]\!] \cong F(\prod_i G([\![\Xi \mid \sigma_i]\!]))$ (F is strong) and $! = FG$, we have, using the adjunction $F \dashv G$, that such a term corresponds to

$$[\![\Xi \mid \vec{x} \colon \vec{\sigma}; - \vdash t]\!]_{\mathbf{Type}} \colon \ \textstyle\prod_i G([\![\sigma_i]\!]) \to G([\![\tau]\!])$$

in $\mathbf{Type}$. It is easy to prove that

$$\begin{gathered} [\![\Xi \mid \Gamma; - \vdash s[t/x]]\!]_{\mathbf{Type}} = \\ [\![\Xi \mid \Gamma, x \colon \sigma; - \vdash s \colon \tau]\!]_{\mathbf{Type}} \circ \langle id_{[\![\Xi \mid \Gamma; -]\!]}, [\![\Xi \mid \Gamma; - \vdash t]\!]_{\mathbf{Type}} \rangle, \end{gathered}$$

using Lemma 3.2.2 of [2].

Definition 4.1. A model of PILL$_Y$ is a model of PILL, which models a fixed point operator

$$Y \colon \Pi\alpha.\, (\alpha \to \alpha) \to \alpha$$

Definition 4.2. A **pre-LAPL**-structure is

1. a schema of categories and functors

$$\begin{tikzcd} & & \mathbf{Prop} \arrow[d, "r"] \\ \mathbf{LinType} \arrow[r, bend right] \arrow[rrd, "p"'] & \mathbf{Type} \arrow[l, bend right] \arrow[r, hook, "I"] \arrow[rd] & \mathbf{Ctx} \arrow[d, "q"] \\ & & \mathbf{Kind} \end{tikzcd}$$

such that

- the diagram

$$\begin{tikzcd}
\mathbf{LinType} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr, "p"'] & & \mathbf{Type} \arrow[ll, bend right, "F"'] \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

  is a model of PILL$_Y$.

- q is a fibration with fibred finite products
- (r, q) is an indexed first-order logic fibration [5] which has products and coproducts with respect to projections $\Xi \times \Omega \to \Xi$ in $\mathbf{Kind}$ [5], where Ω is p applied to the generic object of p.
- I is a faithful product-preserving map of fibrations.

2. a contravariant morphism of fibrations:

$$\begin{tikzcd}
\mathbf{LinType} \times_{\mathbf{Kind}} \mathbf{LinType} \arrow[rr, "U"] \arrow[dr] & & \mathbf{Ctx} \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

3. a family of bijections

$$\Psi : \mathrm{Hom}_{\mathbf{Ctx}_\Xi}(\xi, U(\sigma, \tau)) \to \mathrm{Obj}\left(\mathbf{Prop}_{\xi \times I(G(\sigma) \times G(\tau))}\right)$$

for σ and τ in $\mathbf{LinType}_\Xi$ and ξ in $\mathbf{Ctx}_\Xi$, which

- is natural in the domain variable ξ
- is natural in σ, τ
- commutes with reindexing functors; that is, if $\rho : \Xi' \to \Xi$ is a morphism in $\mathbf{Kind}$ and $u : \xi \to U(\sigma, \tau)$ is a morphism in $\mathbf{Ctx}_\Xi$, then

$$\Psi(\rho^*(u)) = (\bar{\rho})^*(\Psi(u))$$

  where $\bar{\rho}$ is the cartesian lift of ρ.

Notice that Ψ is only defined on vertical morphisms.

By contravariance of the fibred functor U we mean that U is contravariant in each fibre. Since U is uniquely defined by the requirements on the rest of the structure so we will often refer to a pre-LAPL structure simply as the diagram in item 1. Strictly speaking, we should denote the bijection Ψ by $\Psi_{\Xi,\xi,\sigma,\tau}$ since it depends on all these, but for ease of notation we simply write Ψ.

We now explain how to interpret a subset of LAPL in a pre-LAPL structure. The subset of LAPL we consider at this stage is LAPL without admissible relations and without the relational interpretation of types.

We interpret the full contexts of the considered subset of LAPL in the category $\mathbf{Ctx}$ as follows. A context

$$\Xi \mid x_1 \colon \sigma_1, \ldots x_n \colon \sigma_n \mid R_1 \colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, R_m \colon \mathsf{Rel}(\tau_m, \tau_m')$$

is interpreted as

$$\textstyle\prod_i IG([\![\sigma_i]\!]) \times \prod_j U([\![\tau_j]\!], [\![\tau_j']\!]),$$

where the interpretations of the types is the usual interpretation of types in $\mathbf{LinType} \to \mathbf{Kind}$.

For notational convenience we shall write $[\![\Xi \mid \Gamma \mid \Theta \vdash t\colon \tau]\!]$ for the interpretation of t in $\mathbf{Ctx}$, that is for

$$I([\![\Xi \mid \Gamma; - \vdash t\colon \tau]\!]_{\mathbf{Type}}) \circ \pi$$

(note the subscript $\mathbf{Type}$), where π is the projection

$$\pi\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma \mid -]\!]$$

in $\mathbf{Ctx}_{[\![\Xi]\!]}$.

The propositions in the logic are interpreted in $\mathbf{Prop}$ as follows.

Let $\Delta_I\colon I \to I \times I$ denote the diagonal map, then

$$[\![\Xi \mid x\colon \tau, y\colon \tau \vdash x =_\tau y]\!] = \textstyle\coprod_{\Delta_{[\![\tau]\!]}}(\top),$$

where $\coprod_{\Delta_{[\![\tau]\!]}}$ denotes the left adjoint to reindexing along Δ. Now we can define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash t =_\sigma u]\!] = \langle [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash u]\!]\rangle^* [\![\Xi \mid x\colon \tau, y\colon \tau \vdash x =_\tau y]\!].$$

To interpret $\forall x\colon \sigma_{i_0}.\phi$, recall that a context $\Xi \mid x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n \mid \Theta$ is interpreted as

$$\textstyle\prod_i IG[\![\sigma_i]\!] \times [\![\Theta]\!],$$

where $[\![\sigma_i]\!]$ is the usual interpretation of types in $\mathbf{LinType}$ and the product refers to the fibrewise product in $\mathbf{Ctx}$. We may therefore interpret $\forall x\colon \sigma_{i_0}.\phi$ using the right adjoint to reindexing along the projection

$$\pi\colon \textstyle\prod_i IG[\![\sigma_i]\!] \times [\![\Theta]\!] \to \prod_{i \neq i_0} IG[\![\sigma_i]\!] \times [\![\Theta]\!].$$

Likewise, $\forall R\colon \mathsf{Rel}(\sigma, \tau.)\phi$ is interpreted using right adjoints to reindexing functors related to the appropriate projection in $\mathbf{Ctx}$. The existential quantifiers $\exists x\colon \sigma_{i_0}.\phi$ and $\exists R\colon \mathsf{Rel}(\sigma, \tau.)\phi$ are interpreted using left adjoints to the same reindexing functors.

Quantification over types $\forall\alpha.\phi$ and $\exists\alpha.\phi$ is interpreted using respectively right and left adjoints to $\bar{\pi}^*$ where $\bar{\pi}$ is the lift of the projection $\pi : [\![\Xi, \alpha\colon \mathsf{Type}]\!] \to [\![\Xi]\!]$ in $\mathbf{Kind}$ to $\mathbf{Ctx}$. To be more precise, one may easily show that for $\Xi \mid \Gamma \mid \Theta$ wellformed $[\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] = \pi^*[\![\Xi \mid \Gamma \mid \Theta]\!]$ using the corresponding result for the interpretation of PILL$_Y$, and so the cartesian lift of π is a map:

$$\bar{\pi}\colon [\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] \ \to [\![\Xi \mid \Gamma \mid \Theta]\!]$$

and we define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \forall\alpha.\, \phi]\!] = \textstyle\prod_{\bar{\pi}} [\![\Xi,\ \alpha \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where $\prod_{\bar{\pi}}$ is the right adjoint to $\bar{\pi}^*$.

Definable relations with domain σ and codomain τ in contexts $\Xi \mid \Gamma \mid \Theta$ are interpreted as maps from $[\![\Xi \mid \Gamma \mid \Theta]\!]$ into $U([\![\sigma]\!], [\![\tau]\!])$. The definable relation

$$\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau) \vdash R\colon \mathsf{Rel}(\sigma, \tau)$$

is interpreted as the projection, and

$$[\![\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi\colon \mathsf{Rel}(\sigma, \tau)]\!] = \Psi^{-1}([\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi]\!]).$$

We now define the interpretation of $\rho(t,s)$, for a definable relation ρ and terms t,s of the right types. First, for $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)$, we define

$$[\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x,y)]\!] = \Psi([\![\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)]\!]).$$

Next, if $\Xi \mid \Gamma \vdash t\colon \sigma, s\colon \tau$, then

$$\begin{gathered}[\![\Xi \mid \Gamma \mid \Theta \vdash \rho(t,s)]\!] = \\ \langle\langle \pi, \langle [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash s]\!]\rangle\rangle, \pi'\rangle^* [\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x,y)]\!],\end{gathered}$$

where π, π' are the projections

$$\pi\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma]\!] \qquad \pi'\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid - \mid \Theta]\!].$$

One may think of the isomorphism Ψ as a model-theoretic version of Lemma 2.27.

To interpret admissible relations, we will assume that we are given a subfunctor V of U, i.e., a contravariant functor V with domain and codomain as U and a natural transformation $V \Rightarrow U$ whose components are all monomorphic. Thus, for all σ, τ, we can consider $V(\sigma,\tau)$ as a subobject of $U(\sigma,\tau)$. We think of $V(\sigma,\tau)$ as the subset of all admissible relations (since the isomorphism Ψ allows us to think of $U(\sigma,\tau)$ as the set of all definable relations).

We may interpret the logic containing admissible relations by interpreting $S\colon \mathsf{AdmRel}(\sigma,\tau)$ as $V([\![\sigma]\!],[\![\tau]\!])$. Admissible relations are interpreted as maps into $V(\sigma,\tau)$. For this to make sense we need, of course, to make sure that the admissible relations in the model (namely the relations that factor through the object of admissible relations) in fact contain the relations that are admissible in the logic. We need to assume that of the functor V.

Definition 4.3. A pre-LAPL structure together with a subfunctor V of U is said to **model admissible relations**, if V is closed under the rules of Figure 4 and Rule 2.18 holds.

Lemma 4.4. *In the interpretation given above of the subset of LAPL excluding the relational interpretation of types in a pre-LAPL structure modeling admissible relations, if*

$$\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}$$

is a proposition in the logic, and

$$\Xi \mid \Gamma \vdash t\colon \sigma$$

is a term, then

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi[t/x]\colon \mathsf{Prop}]\!] = \langle\langle \pi, [\![\Xi \mid \Gamma \mid \Theta \vdash t\colon \sigma]\!]\rangle, \pi'\rangle^* [\![\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}]\!],$$

where π, π' are the projections

$$\pi\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma]\!] \qquad \pi'\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid - \mid \Theta]\!].$$

Proof. By induction on the structure of ϕ. Cases $R(s,s')$ and $s =_\tau s'$ are easy from definitions, simply using the fact that

$$\begin{gathered}[\![\Xi \mid \Gamma, x\colon \sigma \vdash s[t/x]]\!]_{\mathbf{Type}} = \\ [\![\Xi \mid \Gamma, x\colon \sigma \vdash s\colon \tau]\!]_{\mathbf{Type}} \circ \langle \pi_{[\![\Xi \mid \Gamma; -]\!]}, [\![\Xi \mid \Gamma \vdash t]\!]_{\mathbf{Type}}\rangle\end{gathered}$$

in the PILL model. The cases $\phi \wedge \phi', \phi \supset \phi'$, etc., are just the fact that the fibrewise structure of $\mathbf{Prop}$ is preserved by reindexing, and the cases of the quantifiers is by the Beck-Chevalley condition. □

Lemma 4.5. *For* $[\![\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma,\tau)]\!] \colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to U(\sigma,\tau)$ *and* $t \colon \sigma' \multimap \sigma$ *and* $s \colon \tau' \multimap \tau$*,*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash (x \colon \sigma', y \colon \tau').\, \rho(t\,x, s\,y)]\!] \colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to U(\sigma',\tau') = U(t,s) \circ [\![\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma,\tau)]\!].$$

Proof. This follows from Lemma 4.4 and naturality of Ψ in σ, τ: Assume for simplicity that Γ and Θ are empty.

Observe $[\![\Xi \mid x \colon \sigma' \vdash t\,x \colon \sigma]\!]_{\mathbf{Type}} = G\,\sigma' \xrightarrow{\eta_{G\,\sigma'}} GFG\,\sigma' \overset{G\,\epsilon_{\sigma'}}{\multimap} G\,\sigma' \xrightarrow{G\,t} G\,\sigma = G(t)$, where η is the unit of the adjunction $F \dashv G$. Now

$$[\![\Xi \mid - \mid - \vdash (x:\sigma, y:\tau).\rho(t\,x, s\,y)]\!] = \Psi_{\Xi}^{-1}([\![\Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(t\,x, s\,y)]\!])$$

which using Lemma 4.4 and the calculation above gives

$$\begin{gathered}\Psi_{\Xi}^{-1}((t \times s)^*([\![\Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(x,y)]\!])) = \\ U(t,s) \circ \Psi_{\Xi}^{-1}([\![\Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(x,y)]\!]) = U(t,s) \circ [\![\Xi \mid - \mid - \vdash \rho \colon \mathsf{Rel}(\sigma,\tau)]\!].\end{gathered}$$

□

Given a pre-LAPL structure modeling admissible relations, we may define a fibration

$$\begin{tikzcd} \mathbf{LinAdmRelations} \arrow[d] \\ \mathbf{AdmRelCtx} \end{tikzcd} ,$$

which we think of as a model consisting of admissible relations. We first define the category $\mathbf{AdmRelCtx}$ by the pullback

$$\begin{tikzcd} \mathbf{AdmRelCtx} \arrow[r] \arrow[d, "\langle \partial_0, \partial_1 \rangle"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Ctx} \arrow[d] \\ \mathbf{Kind} \times \mathbf{Kind} \arrow[r, "\times"] & \mathbf{Kind}. \end{tikzcd}$$

We write an object Θ in $\mathbf{AdmRelCtx}$ over (Ξ, Ξ') as $\Xi, \Xi' \mid \Theta$. The fibre of $\mathbf{LinAdmRelations}$ over an object $\Xi, \Xi' \mid \Theta$ is

objects triples (ϕ, σ, τ) where σ and τ are objects in $\mathbf{LinType}$ over Ξ and Ξ' respectively and ϕ is an admissible relation, i.e. a vertical map

$$\phi \colon \Theta \to V(\pi^*\sigma, \pi'^*\tau)$$

in $\mathbf{Ctx}$. Here π, π' are first and second projection respectively out of $\Xi \times \Xi'$.

morphisms A morphism $(\phi, \sigma, \tau) \to (\psi, \sigma', \tau')$ is a pair of morphism

$$(t \colon \sigma \multimap \sigma', u \colon \tau \multimap \tau')$$

in $\mathbf{LinType}_{\Xi}$ and $\mathbf{LinType}_{\Xi'}$ respectively, such that

$$\Psi(\phi) \leq \Psi(V(t,u) \circ \psi)),$$

where we have left the inclusion of V into U implicit.

Reindexing with respect to vertical maps $\rho\colon \Theta \to \Theta'$ in $\mathbf{Ctx}$ is done by composition. Reindexing objects of $\mathbf{LinAdmRelations}$ with respect to lifts of maps in $\mathbf{Kind} \times \mathbf{Kind}$ is done by reindexing in the fibration $\mathbf{Ctx} \to \mathbf{Kind}$. Reindexing of morphisms in $\mathbf{LinAdmRelations}$ with respect to maps in $\mathbf{Kind} \times \mathbf{Kind}$ is done by reindexing each map in $\mathbf{LinType} \to \mathbf{Kind}$. This defines all reindexing since all maps in $\mathbf{AdmRelCtx}$ can be written as a vertical map followed by a cartesian map.

Remark 4.6. In the internal language, objects of $\mathbf{LinAdmRelations}$ are admissible relations

$$\Xi; \Xi' \mid \Theta \ \vdash \rho\colon \mathsf{AdmRel}(\sigma, \tau).$$

A vertical morphisms in $\mathbf{LinAdmRelations}$ from $\rho\colon \mathsf{AdmRel}(\sigma, \tau)$ to $\rho'\colon \mathsf{AdmRel}(\sigma', \tau')$ is a pair of morphisms $f\colon \sigma \multimap \sigma'$, $g\colon \tau \multimap \tau'$ in $\mathbf{LinType}$ such that in the internal language the formula

$$\forall x\colon \sigma, y\colon \tau \,.\, \rho(x, y) \supset \rho'(f\,x, g\,y)$$

holds (this follows directly from Lemma 4.4).

There exist two canonical maps of fibrations:

$$\left(\begin{tikzcd} \mathbf{LinAdmRelations} \arrow[d] \\ \mathbf{AdmRelCtx} \end{tikzcd}\right) \begin{tikzcd} {} \arrow[r, shift left, "\partial_0"] \arrow[r, shift right, "\partial_1"'] & {} \end{tikzcd} \left(\begin{tikzcd} \mathbf{LinType} \arrow[d] \\ \mathbf{Kind} \end{tikzcd}\right).$$

On the base category ∂_0, ∂_1 map an object $\Xi, \Xi' \mid \Theta$ to Ξ and Ξ' respectively. On the total category they map (ϕ, σ, τ) to σ and τ respectively. In words, ∂_0 and ∂_1 map a relation to its domain and codomain respectively.

Lemma 4.7. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has products in the base, a generic object and simple products with respect to projections in* $\mathbf{AdmRelCtx}$ *forgetting the generic object. The maps* ∂_0, ∂_1 *preserve this structure.*

Proof. The category $\mathbf{AdmRelCtx}$ has products:

$$(\Xi_1, \Xi_1' \mid \Theta_1) \times (\Xi_2, \Xi_2' \mid \Theta_2) = \Xi_1 \times \Xi_2, \Xi_1' \times \Xi_2' \mid \pi^*(\Theta_1) \times \pi'^*(\Theta_2)$$

(see [12, Proposition 9.2.1]).

The fibration has a generic object $\Omega, \Omega \mid V(\widehat{id_\Omega}, \widehat{id_\Omega})$, since a morphism into this from $\Xi, \Xi' \mid \Theta$ in $\mathbf{AdmRelCtx}$ consists of pairs of types $(f : \Xi \to \Omega, g : \Xi' \to \Omega)$ and a morphism from Θ to $V(\hat{f}, \hat{g})$.

We now show that we have products with respect to projections forgetting the generic object. Given a relation

$$\Xi, \alpha; \Xi', \beta \mid \Theta, R\colon \mathsf{AdmRel}(\alpha, \beta) \vdash \rho\colon \mathsf{AdmRel}(\tau, \tau')$$

we can define

$$\Xi, \Xi' \mid \Theta \vdash \forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, \rho\colon \mathsf{AdmRel}((\textstyle\prod \alpha\colon \mathsf{Type}.\, \tau), (\prod \beta\colon \mathsf{Type}.\, \tau'))$$

as

$$\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, \rho = (t, u).\, \forall \alpha, \beta\colon \mathsf{Type}.\, \forall R\colon \mathsf{AdmRel}(\alpha, \beta).\, (t\alpha)\rho(u\beta).$$

We will to show that this defines a right adjoint to weakening. Suppose we have another relation

$$\Xi, \Xi' \mid \Theta \vdash \omega \colon \mathsf{AdmRel}(\sigma, \sigma').$$

We will use the usual adjunction in $\mathbf{LinType}$, where a map $\Xi, \alpha \mid - \vdash t \colon \sigma \multimap \tau$, with $\Xi \vdash \sigma \colon \mathsf{Type}$ corresponds to

$$\Xi \mid - \vdash \hat{t} = \lambda^{\circ} x \colon \sigma.\, \Lambda\alpha.\, (t\, x) \colon \sigma \multimap \prod \alpha.\, \tau.$$

We need to prove that (t, u) preserves relations iff $(\hat{t}, \hat{u})$ does, but it is clear that

$$\Xi, \alpha; \Xi', \beta \mid x : \sigma, y : \sigma' \mid \Theta, R \colon \mathsf{AdmRel}(\alpha, \beta) \mid x\omega y \vdash (t\, x)\rho(u\, y)$$

iff

$$\Xi, \Xi' \mid x : \sigma, y : \sigma' \mid \Theta \mid x\omega y \vdash \forall \alpha, \beta \colon \mathsf{Type}.\, \forall R \colon \mathsf{AdmRel}(\alpha, \beta).\, (\hat{t}\, x\, \alpha)\rho(\hat{u}\, y\, \beta),$$

which establishes the bijective correspondence between maps

$$\frac{\pi^* \omega \multimap \rho}{\omega \multimap \forall(\alpha, \beta, R \colon \mathsf{Rel}(\alpha, \beta)).\, \rho}$$

proving that we have in fact defined a product. □

Lemma 4.8. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has a fibrewise SMCC-structure and the two maps* ∂_0, ∂_1 *are fibred strict symmetric monoidal functors.*

Proof. We prove that the constructions $\otimes$, $\multimap$ on definable relations given in Section 2.2.2 define a fibrewise symmetric monoidal structure on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$. Notice that since V is closed under the rules of Figure 4, Proposition 2.3 tells us that the constructions on definable relations of Section 2.2.2 indeed do define operations on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$.

First we will prove that the two operators $- \otimes -$, $\rho \multimap -$ do in fact define functors on $\mathbf{LinAdmRelations}$. That is, we need to check that if

$$(t_0, s_0) \colon \rho_0 \multimap \rho_0' \quad (t_1, s_1) \colon \rho_1 \multimap \rho_1',$$

then

$$(t_0 \otimes t_1, s_0 \otimes s_1) \colon \rho_0 \otimes \rho_1 \multimap \rho_0' \otimes \rho_1',$$

and, if $(t, s) \colon \rho' \multimap \rho''$, then

$$(t \circ -, s \circ -) \colon (\rho \multimap \rho') \multimap (\rho \multimap \rho'').$$

To see that $\otimes$ defines a functor, suppose $x(\rho_0 \otimes \rho_1)y$ and $f(\rho_0' \multimap \rho_1' \multimap R)g$. We need to show that

$$R(\text{let } z \otimes z' \text{ be } (t_0 \otimes t_1)(x) \text{ in } f\ z\ z', \text{let } z \otimes z' \text{ be } (s_0 \otimes s_1)(y) \text{ in } g\ z\ z').$$

Recall that $(t_0 \otimes t_1)x = \text{let } \omega \otimes \omega' \text{ be } x \text{ in } t_0\ \omega \otimes t_1\ \omega'$ in PILL. Notice then that

$$\text{let } z \otimes z' \text{ be } (t_0 \otimes t_1)(y) \text{ in } f\ z\ z' = \text{let } z \otimes z' \text{ be } y \text{ in } f\ (t_0(z))\ (t_1(z')),$$

and

$$(\lambda^{\circ} z, z'.\, f\ (t_0(z))\ (t_1(z')))(\rho_0 \multimap \rho_1 \multimap R)(\lambda^{\circ} z, z'.\, g\ (s_0(z))\ (s_1(z'))).$$

The result now follows from the assumption that $x(\rho_0 \otimes \rho_1)y$.

To prove that $\rho \multimap -$ is a functor, suppose $(f, g)\colon \rho \multimap \rho'$ and $\rho(x, y)$. Then clearly $\rho''(t \circ f(x), s \circ g(y))$, as required.

We need to show that $\rho \multimap -$ is right adjoint to $- \otimes \rho$, and that the adjoint components are natural in ρ. Since we are given a similar adjunction in $\mathbf{LinType}$, all we need to show is that

$$(t, s)\colon \rho \multimap (\rho' \multimap \rho'')$$

iff

$$(\hat{t}, \hat{s})\colon \rho \otimes \rho' \multimap \rho'',$$

where $\hat{t}, \hat{s}$ are the maps corresponding to t, s in the adjunction on $\mathbf{LinType}$. Suppose first that

$$(t, s)\colon \rho \multimap (\rho' \multimap \rho'') \text{ and } x(\rho \otimes \rho')y.$$

The definition of the latter says exactly that, for all $(t, s)\colon \rho \multimap (\rho' \multimap \rho'')$, we must have $\rho''(\hat{t}\, x, \hat{s}\, y)$.

Now, suppose $(\hat{t}, \hat{s})\colon \rho \otimes \rho' \multimap \rho''$ and $x\rho y \wedge x'\rho' y'$. By Lemma 2.29 $\rho \otimes \rho'(x \otimes x', y \otimes y')$ and so $\rho''(\hat{t}(x \otimes x'), \hat{s}(y \otimes y'))$. Hence, since $\hat{t}(x \otimes x') = t\, x\, x'$ (likewise for s), we are done.

We now proceed to prove that the functors $- \otimes -$, $\rho \multimap -$ define a fibred SMCC structure on

$$\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}.$$

The unit in a fibre is $I_{Rel}\colon \mathsf{AdmRel}(I, I)$ where I is the unit in the appropriate fibres of $\mathbf{LinType}$. The maps giving the isomorphisms

$$\begin{gathered}(-) \otimes ((=) \otimes (\equiv)) \cong ((-) \otimes (=)) \otimes (\equiv),\\ (-) \otimes I \cong (-), \quad (-) \otimes (=) \cong (=) \otimes (-)\end{gathered}$$

are simply pairs of the corresponding maps in the fibrewise SMC-structure of $\mathbf{LinType}$. These maps satisfy the coherence properties simply because the maps in $\mathbf{LinType}$ do the same. One has to check that the maps defined by pairing maps in fact define maps in $\mathbf{LinAdmRelations}$, i.e., that they preserve relations.

One direction of the isomorphism $\sigma \otimes I \cong \sigma$ is given by the map $\lambda^\circ x\colon \sigma.\, x \otimes \star$. To see that this preserves relations, suppose $x\rho y$. Since $\star I_{Rel} \star$, $(x \otimes \star)(\rho \otimes I_{Rel})(y \otimes \star)$ by Lemma 2.29. For the other direction, consider $f_\sigma\colon \sigma \multimap I \multimap \sigma$ given as $f = \lambda^\circ x\colon \sigma \lambda^\circ x'\colon I.\,\mathrm{let}\ \star\ \text{be}\ x'\ \text{in}\ x$. Then the map $\sigma \otimes I \multimap \sigma$ is simply $\hat{f}_\sigma$, and we have proved earlier that it now suffices to prove that f_σ preserves relations. So suppose $x\rho y \wedge x' I_{Rel} y'$. By definition of $x' I_{Rel} y'$ we can conclude that $\rho(f_\sigma\, x\, x', f_\tau\, y\, y')$.

The isomorphism $(\rho \otimes \rho') \otimes \rho'' \cong \rho \otimes (\rho' \otimes \rho'')$ is obtained using the adjunction as follows

$$\begin{array}{c}(\rho \otimes \rho') \otimes \rho'' \multimap \rho''' \\ \hline\hline \rho \multimap \rho' \multimap \rho'' \multimap \rho''' \\ \hline\hline \rho \multimap (\rho' \otimes \rho'') \multimap \rho''' \\ \hline\hline \rho \otimes (\rho' \otimes \rho'') \multimap \rho'''\end{array}$$

and it is easily seen that this isomorphism is given by pairs of the usual maps in $\mathbf{LinType}$. Likewise the isomorphism $\rho \otimes \rho' \cong \rho' \otimes \rho$ comes from

$$\begin{array}{c}\rho \otimes \rho' \multimap \rho'' \\ \hline\hline \rho \multimap \rho' \multimap \rho'' \\ \hline\hline \rho' \multimap \rho \multimap \rho'' \\ \hline\hline \rho' \otimes \rho \multimap \rho''.\end{array}$$

By construction, the functors ∂_0, ∂_1 are fibred strict symmetric monoidal functors. □

Lemma 4.9. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has a fibred comonad structure induced by the functor* $\rho \mapsto !\rho$*. The maps* ∂_0, ∂_1 *map this comonad to the fibred comonad* ! *on the nose.*

Proof. We need to check that ! defines a functor, i.e., that if $(f, g) \colon \rho \multimap \rho'$, then $(!f, !g) \colon !\rho \multimap !\rho'$. It is easy to see that

$$\forall x, y.\, !\rho(!x, !y) \supset !\rho'((!f)(!x), (!g)(!y))$$

since $(!f)(!x) = !(f(x))$. Now the result follows from Rule 2.18.

The comonad maps are given by $(\epsilon, \epsilon) \colon !\rho \multimap \rho$ and $(\delta, \delta) \colon !\rho \multimap !!\rho$. These preserve relations by Lemma 2.32, and the commutative diagrams of a fibred comonad are preserved since they hold for the fibred comonad on $\mathbf{LinType} \to \mathbf{Kind}$. □

Lemma 4.10. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has natural transformations*

$$d \colon !(-) \Rightarrow !(-) \otimes !(-), e \colon !(-) \Rightarrow I_{Rel}$$

making it a fibred linear fibration. The maps ∂_0, ∂_1 *preserve this structure on the nose.*

Proof. The maps d, e are given by (d, d) and (e, e), which preserve relations by Lemma 2.33. The necessary diagrams commute by the same diagrams for the fibred comonad on $\mathbf{LinType} \to \mathbf{Kind}$, and the functors ∂_0, ∂_1 preserve the structure on the nose by construction. □

If we define $\mathbf{AdmRelations}$ to be the category of finite products of coalgebras [17], we obtain a PILL-model

$$\begin{tikzcd}
\mathbf{LinAdmRelations} \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{AdmRelations} \arrow[ll, bend right=10] \arrow[dl] \\
& \mathbf{AdmRelCtx} &
\end{tikzcd}$$

and two maps of PILL-models ∂_0, ∂_1. This model need not be a PILL$_Y$-model, since for pre-LAPL-structures Y does not necessarily preserve relations.

Definition 4.11. An **LAPL-structure** is a pre-LAPL-structure modeling admissible relations, together with a map of PILL-models J from

$$\begin{tikzcd}
\mathbf{LinType} \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{Type} \arrow[ll, bend right=10] \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

to

$$\begin{tikzcd}
\mathbf{LinAdmRelations} \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{AdmRelations} \arrow[ll, bend right=10] \arrow[dl] \\
& \mathbf{AdmRelCtx} &
\end{tikzcd}$$

such that when restricting to the fibred linear categories, J together with ∂_0, ∂_1 is a reflexive graph, i.e., $\partial_0 \circ J = \partial_1 \circ J = id$.

In the following, we will often confuse J with the map of fibred linear categories from $\mathbf{LinType} \to \mathbf{Kind}$ to $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$.

We need to show how to interpret the rule

$$\frac{\alpha_1, \dots, \alpha_n \vdash \sigma(\vec{\alpha})\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho_1\colon \mathsf{AdmRel}(\tau_1, \tau_1'), \dots, \rho_n\colon \mathsf{AdmRel}(\tau_n, \tau_n')}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}]\colon \mathsf{AdmRel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))}$$

in LAPL-structures.

Since J preserves products in the base and generic objects, $J(\llbracket \vec{\alpha} \vdash \sigma(\vec{\alpha}) \rrbracket)$ is a relation from $\sigma(\vec{\alpha})$ to $\sigma(\vec{\beta})$ in context $\llbracket \vec{\alpha}; \vec{\beta} \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}) \rrbracket$. It thus makes sense to define

$$\llbracket \vec{\alpha}, \vec{\beta} \mid - \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[\vec{R}] \rrbracket$$

to be $J(\llbracket \vec{\alpha} \mid \sigma(\vec{\alpha}) \rrbracket)$, so all we need to do now is to reindex this object. We reindex it to the right Kind context using

$$\langle \vec{\tau}, \vec{\tau}' \rangle : \llbracket \Xi \rrbracket \to \Omega^{2n},$$

thus obtaining

$$\llbracket \Xi \mid - \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}') \vdash \sigma[\vec{R}]\colon \mathsf{Rel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}')) \rrbracket.$$

For $\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')$, we define

$$\begin{gathered}\llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}]\colon \mathsf{AdmRel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}')) \rrbracket = \\ \llbracket \Xi \mid - \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}') \vdash \sigma[\vec{R}] \rrbracket \circ \llbracket \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}') \rrbracket.\end{gathered}$$

where by $\llbracket \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}') \rrbracket$ we mean the pairing

$$\langle \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho_1 \rrbracket, \dots, \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho_n \rrbracket \rangle.$$

Remark 4.12. To model a version of Linear Abadi & Plotkin Logic for unary or other arities of parametricity as in Remark 2.2, the functor U should, have corresponding arity and the domain and codomain of the bijection Ψ should be changed accordingly. Furthermore instead of considering the fibration of binary relations $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ we should consider a fibration of relations of the appropriate arity.

4.1 Soundness

In this section we prove that the interpretation of LAPL in LAPL-structures is sound. First, we present a series of reindexing lemmas.

Lemma 4.13. *If* $\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}$ *is a proposition in the logic, and*

$$\Xi \mid \Gamma \vdash t\colon \sigma$$

is a term, then

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi[t/x]\colon \mathsf{Prop} \rrbracket = \langle \langle \pi, \llbracket \Xi \mid \Gamma \mid \Theta \vdash t\colon \sigma \rrbracket \rangle, \pi' \rangle^* \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop} \rrbracket$$

where π, π' *are the projections*

$$\pi\colon \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket \to \llbracket \Xi \mid \Gamma \rrbracket \qquad \pi'\colon \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket \to \llbracket \Xi \mid - \mid \Theta \rrbracket.$$

Lemma 4.14. *If $\Xi \mid \Gamma, x \mid \Theta \vdash \rho \colon \mathsf{Rel}(\tau, \tau')$ is a definable relation in the logic, and*

$$\Xi \mid \Gamma \vdash t \colon \sigma$$

is a term, then

$$\llbracket \Xi \mid \Gamma, x \mid \Theta \vdash \rho \rrbracket \circ \langle\langle \pi, \llbracket \Xi \mid \Gamma \mid \Theta \vdash t \colon \sigma \rrbracket \rangle, \pi' \rangle = \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho[t/x] \rrbracket.$$

Notice that Lemma 4.13 differs from Lemma 4.4 since the latter only concerns the interpretation of the part of the logic not including the relational interpretation of types.

Proof. The two lemmas above are proved simultaneously. We only include the proof of the former, for which we only need to extend the proof of Lemma 4.4 to the case of $\rho(s, u)$. But this follows easily by induction using the latter lemma. □

Lemma 4.15. *If $\Xi \mid \Gamma \mid \Theta \vdash \phi \colon \mathsf{Prop}$ then*

$$\llbracket \Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \phi \colon \mathsf{Prop} \rrbracket = \pi^* \llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi \colon \mathsf{Prop} \rrbracket,$$

where π is the obvious projection. Likewise, if $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)$ then

$$\llbracket \Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau) \rrbracket = \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau) \rrbracket \circ \pi,$$

where π is the obvious projection.

Proof. The lemma can be proved in a way similar to Lemmas 4.13 and 4.15. □

Lemma 4.16. *If $\Xi \vdash \sigma \colon \mathsf{Type}$ then*

$$\llbracket \Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \vdash \phi[\sigma/\alpha] \rrbracket = \overline{\langle id_{\llbracket \Xi \rrbracket}, \llbracket \sigma \rrbracket \rangle}^* \llbracket \Xi, \alpha \colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash \phi \rrbracket,$$

and

$$\llbracket \Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \vdash \rho[\sigma/\alpha] \rrbracket = \overline{\langle id_{\llbracket \Xi \rrbracket}, \llbracket \sigma \rrbracket \rangle}^* \llbracket \Xi, \alpha \colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash \rho \rrbracket,$$

where the vertical line in $\overline{\langle id_{\llbracket \Xi \rrbracket}, \llbracket \sigma \rrbracket \rangle}$ denotes the cartesian lift.

Proof. We know that

$$\llbracket \Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \rrbracket = \langle id_{\llbracket \Xi \rrbracket}, \llbracket \sigma \rrbracket \rangle^* \llbracket \Xi, \alpha \colon \mathsf{Type} \mid \Gamma \mid \Theta \rrbracket$$

since the corresponding statement holds in the PILL-model and the functors F, G, I commute with reindexing.

Now one proceeds by simultaneous induction on ϕ and ρ. For $\rho = R$ and for $\rho = \tau[\vec{\rho}]$ one uses that Ψ commutes with reindexing. For $\phi = u =_\tau u'$ one uses the Beck-Chevalley condition, as is also done for the cases of $\exists$ and $\forall$. The remaining cases either follow by induction or from the fact that the fibrewise structure in $\mathbf{Prop}$ ($\supset$, $\wedge$, etc.) is preserved by reindexing.

□

Lemma 4.17. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi$ *then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!] = \overline{\pi}^*_{\Xi,\alpha\to\Xi}[\![\Xi, \alpha \mid \Gamma \mid \Theta \vdash \phi]\!].$$

Likewise, if If $\Xi \mid \Gamma \mid \Theta \vdash \rho$ *then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!] = \overline{\pi}^*_{\Xi,\alpha\to\Xi}[\![\Xi, \alpha \mid \Gamma \mid \Theta \vdash \rho]\!].$$

Proof. By simultaneous induction. □

Lemma 4.18. *If* $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation and*

$$\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\tau, \tau') \vdash \phi,$$

then

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi[\rho/R]]\!] = (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\rho]\!]\rangle)^*[\![\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\tau, \tau') \vdash \phi]\!].$$

Likewise, If $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation and*

$$\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\tau, \tau') \vdash \rho' \colon \mathsf{Rel}(\sigma, \sigma'),$$

then

$$[\![\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\tau, \tau') \vdash \rho']\!] \circ (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!]\rangle) = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho'[\rho/R]]\!].$$

The same holds for substitution of admissible relations.

Proof. By simultaneous induction on ϕ and ρ', using naturality of Ψ, Beck-Chevalley and the fact that the fibrewise structure in Prop is preserved by reindexing. □

Lemma 4.19. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi$ *is a proposition then*

$$[\![\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\sigma, \tau) \vdash \phi]\!] = \pi^*[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where π *is the obvious projection. Likewise, if* $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma', \tau')$ *is a definable relation then*

$$[\![\Xi \mid \Gamma \mid \Theta, R \colon \mathsf{Rel}(\sigma, \tau) \vdash \rho]\!] = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!] \circ \pi,$$

where π *is the obvious projection. The same holds for substitution of admissible relations.*

Proof. Again by simultaneous induction. □

Theorem 4.20 (Soundness). *The interpretation given above of LAPL in LAPL-structures is sound with respect to the Rules and Axioms 2.9-2.26.*

Proof. Rules 2.9-2.16 hold since the interpretation of quantification is given by adjoints to weakening, considering Lemmas 4.15, 4.17, 4.19 above.

Rules 2.4-2.7 hold since substitution corresponds to reindexing as in the lemmas above.

Rule 2.8 is proved exactly as in [5].

Rule 2.17 holds since externally equal maps are interpreted equally in the model, by soundness of the interpretation of PILL$_Y$. Clearly internal equality is an equivalence relation.

Rule 2.18 is required to hold in Definition 4.3.

Rules 2.19-2.24 all hold since J preserves SMCC-structure, generic objects, simple products and !.

To prove soundness of Rule 2.25, it suffices to prove soundness of

$$\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \mid \top \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(x,y) \supset\subset \phi,$$

but

$$\llbracket \Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(x,y) \rrbracket = \Psi(\llbracket \Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi \rrbracket) =$$
$$\Psi \circ \Psi^{-1}(\llbracket \Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi \rrbracket) = \llbracket \Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi \rrbracket.$$

To prove Axiom 2.26, notice that J is required to be a functor. This means that it maps $\llbracket Y \rrbracket \colon I \multimap \llbracket \prod \alpha.\, (\alpha \to \alpha) \to \alpha \rrbracket$ to a morphism from I_{Rel} to the relational interpretation of $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$. By the requirement, that $(J, \partial_0, \partial_1)$ is a reflexive graph, this map must be $(\llbracket Y \rrbracket, \llbracket Y \rrbracket)$. Since $\star I_{Rel} \star$ and $\llbracket Y \rrbracket(\star) = Y$ we get $Y(\prod \alpha.\, (\alpha \to \alpha) \to \alpha) Y$. □

4.2 Completeness

Theorem 4.21 (Completeness). *There exists an LAPL-structure with the property that any formula of LAPL over pure PILL$_Y$ holds in this model iff it is provable in LAPL.*

Proof. We construct the LAPL-structure syntactically, giving the categories in question the same names as in the diagrams of the definitions of pre-LAPL- and LAPL-structures.

- The category $\mathbf{Kind}$ has as objects sequences of the form

  $$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type},$$

  where we identify these contexts up to renaming (in other words, we may think of objects as natural numbers). A morphism from Ξ into

  $$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_n \colon \mathsf{Type}$$

  is a sequence of types $(\sigma_1, \ldots, \sigma_n)$ such that all σ_i are well-formed in context Ξ.

- Objects in the fibre of $\mathbf{LinType}$ over Ξ are well-formed types in this context. Morphisms in this fibre from σ to τ are equivalence classes of terms t such that $\Xi \mid -; x\colon \sigma \vdash t\colon \tau$, where we identify terms up to external equality. Equivalently, we may think of morphisms as terms $\Xi \mid -; - \vdash t\colon \sigma \multimap \tau$. Composition is by substitution, and reindexing with respect to morphisms in $\mathbf{Kind}$ is by substitution.

- Objects in the fibre of $\mathbf{Type}$ over Ξ are well-formed sequences of types in this context. Morphism in this fibre from $\sigma_1, \ldots, \sigma_n$ to $\tau_1, \ldots, \tau_m$ are equivalence classes of sequences of terms $(t_i)_{i \leq m}$, such that for each i the term

  $$\Xi \mid \vec{x}\colon \vec{\sigma}; - \vdash t_i \colon \tau_i$$

  is well-formed, where the sequences (t_i) and (t'_i) are identified if, for each i, t_i is externally equal to t'_i. Reindexing with respect to morphisms in $\mathbf{Kind}$ is by substitution.

- The functor $\mathbf{LinType} \to \mathbf{Type}$ maps a morphism $-; x\colon \sigma \vdash t\colon \tau$ to $x\colon \sigma; - \vdash t\colon \tau$. The functor going the other way maps a sequence of objects (σ_i) to $\otimes_i !\sigma_i$. It maps a morphism (t_i) from (σ_i) to (τ_i) to

  $$\Xi \mid -; y\colon \otimes_i !\sigma_i \vdash \text{let } \otimes_i x'_i \colon \otimes_i !\sigma_i \text{ be } y \text{ in let } !\vec{x} \text{ be } \vec{x}' \text{ in } \otimes_i !t_i.$$

  For further details of the term model for PILL see [2].

- The category $\mathbf{Ctx}$ has as objects in the fibre over Ξ well-formed contexts of LAPL: $\Xi \mid \Gamma \mid \Theta$. A vertical morphism from $\Xi \mid \Gamma \mid \Theta$ to

  $$\Xi \mid \Gamma' \mid R_1 \colon \mathsf{Rel}(\sigma_1, \tau_1), \dots, R_n \colon \mathsf{Rel}(\sigma_n, \tau_n), S_1 \colon \mathsf{AdmRel}(\sigma'_1, \tau'_1), \dots, S_m \colon \mathsf{AdmRel}(\sigma'_m, \tau'_m)$$

  is a triple, consisting of a morphism $\Xi \mid \Gamma \to \Xi \mid \Gamma'$ in the sense of morphisms in $\mathbf{Type}$, a sequence of definable relations $(\rho_1, \dots, \rho_n)$, and a sequence of admissible relations $(\omega_1, \dots, \omega_m)$, such that $\Xi \mid \Gamma \mid \Theta \vdash \rho_i \colon \mathsf{Rel}(\sigma_i, \tau_i)$ and $\Xi \mid \Gamma \mid \Theta \vdash \omega_i \colon \mathsf{AdmRel}(\sigma'_i, \tau'_i)$. We identify two such morphisms represented by the same type morphism and the definable relations $(\rho_1, \dots, \rho_n)$ and $(\rho'_1, \dots, \rho'_n)$ and admissible relations $(\omega_1, \dots, \omega_m)$ and $(\omega'_1, \dots, \omega'_m)$, respectively, if, for each i, j, the formulas $\rho_i \equiv \rho'_i$ and $\omega_j \equiv \omega'_j$ are provable in the logic, where, as usual, $\rho_i \equiv \rho'_i$ is short for

  $$\forall x \colon \sigma_i, y \colon \tau_i.\, \rho_i(x_i, y_i) \supset\subset \rho'_i(x_i, y_i),$$

  and likewise for $\omega_j \equiv \omega'_j$. The inclusion functor I is the obvious one. Reindexing is by substitution.

- The fibre of the category $\mathbf{Prop}$ over a context $\Xi \mid \Gamma \mid \Theta$ has as objects formulas in that context, where we identify two formulas if they are provably equivalent. These are ordered by the implication in the logic. Reindexing is done by substitution, that is, reindexing with respect to lifts of morphisms from $\mathbf{Kind}$ is done by substitution in type-variables, whereas reindexing with respect to vertical maps in $\mathbf{Ctx}$ is by substitution in term variables and relation variables.

An easy fibred version of the completeness proof in [2] shows that $\mathbf{Kind}, \mathbf{Type}, \mathbf{LinType}$ together with the functors described above form a PILL$_Y$ model. The fibration $\mathbf{Ctx} \to \mathbf{Kind}$ clearly has fibred products formed by appending contexts, and the inclusion functor I is clearly faithful and product-preserving.

We need to prove that $\mathbf{Prop} \to \mathbf{Ctx} \to \mathbf{Kind}$ is an indexed first-order logic fibration with products and coproducts with respect to simple projections in $\mathbf{Kind}$. The fibrewise bicartesian structure is given by $\vee, \wedge, \supset, \bot, \top$. Fibred simple products and coproducts are given by quantifying over relations and variables, simple products in the composite fibration is given by quantifying over types. We can in fact prove that the composite fibration has all indexed products and coproducts (in particular, that it has equality).

Suppose $(\vec{t}, \vec{\rho})$ represents a morphism from $\Xi \mid \vec{x} \colon \vec{\sigma} \mid \vec{R}$ to $\Xi \mid \vec{y} \colon \vec{\tau} \mid \vec{S}$ (the vectors $\vec{R}, \vec{S}$ consist of both definable and admissible relations, and the vector $\vec{\rho}$ is a concatenation of the corresponding vectors of admissible and definable relations from the definition above). We can then define the product functor in $\mathbf{Prop}$ as:

$$\begin{gathered}\textstyle\prod_{(\vec{t},\vec{\rho})}(\Xi \mid \vec{x} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) = \\ \Xi \mid \vec{y} \mid \vec{S} \vdash \forall \vec{x}. \forall \vec{R}(\vec{t}\vec{x} = \vec{y} \wedge (\vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S}) \supset \phi(\vec{x}, \vec{R})).\end{gathered}$$

We define coproduct as:

$$\begin{gathered}\textstyle\coprod_{(\vec{t},\vec{\rho})}(\Xi \mid \vec{x} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) = \\ \Xi \mid \vec{y} \mid \vec{S} \vdash \exists \vec{x}. \exists \vec{R}. \vec{t}\vec{x} = \vec{y} \wedge \vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S} \wedge \phi(\vec{x}, \vec{R})).\end{gathered}$$

We remark that the equality we will actually be using in the model is the obvious

$$(\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \phi) \mapsto (\Xi \mid \Gamma, x \colon \sigma, y \colon \sigma \mid \Theta \vdash \phi \wedge x =_\sigma y)$$

The functor U of item 2 is defined as

$$U(\sigma, \tau) = R \colon \mathsf{Rel}(\sigma, \tau)$$

and

$$U(t\colon \sigma \multimap \sigma', u\colon \tau \multimap \tau') = \Xi \mid R\colon \mathsf{Rel}(\sigma', \tau') \vdash (x\colon \sigma, y\colon \tau).\, R(tx, uy).$$

The required isomorphism Ψ is just the isomorphism given by Lemma 2.27. The functor V is defined as

$$V(\sigma, \tau) = R\colon \mathsf{AdmRel}(\sigma, \tau)$$

and

$$V(t\colon \sigma \multimap \sigma', u\colon \tau \multimap \tau') = \Xi \mid R\colon \mathsf{AdmRel}(\sigma', \tau') \vdash (x\colon \sigma, y\colon \tau).\, R(tx, uy).$$

We have defined a pre-LAPL-structure modeling admissible relations. If we construct $\mathbf{AdmRelCtx}$ as in the definition of LAPL-structure, we obtain:

Objects $\vec{\alpha}, \vec{\beta} \mid \Gamma \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$.

Morphisms A morphism from

$$\vec{\alpha}, \vec{\beta} \mid \Gamma \ \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$$

to

$$\vec{\alpha}', \vec{\beta}' \mid \Gamma' \ \mid \vec{S}\colon \mathsf{AdmRel}(\vec{\omega}(\vec{\alpha}'), \vec{\kappa}(\vec{\beta}')), \vec{S}'\colon \mathsf{Rel}(\vec{\omega}'(\vec{\alpha}'), \vec{\kappa}'(\vec{\beta}'))$$

consists of two morphism in $\mathbf{Kind}$:

$$\vec{\mu} : \vec{\alpha} \to \vec{\alpha}'$$

and

$$\vec{\nu} : \vec{\beta} \to \vec{\beta}',$$

a morphism from $\vec{\alpha}, \vec{\beta} \mid \Gamma$ to $\vec{\alpha}, \vec{\beta} \mid \Gamma'[\vec{\mu}, \vec{\nu}/\vec{\alpha}', \vec{\beta}']$ in $\mathbf{LinType}_{\vec{\alpha}, \vec{\beta}}$, and a sequence of admissible relations $\vec{\rho}$ and a sequence of definable relations $\vec{\rho}'$ such that, for all i,j,

$$\vec{\alpha}, \vec{\beta} \mid \Gamma \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta})) \vdash \rho_i\colon \mathsf{AdmRel}(\omega_i(\vec{\mu}), \kappa_i(\vec{\nu}))$$
$$\vec{\alpha}, \vec{\beta} \mid \Gamma' \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta})) \vdash \rho_j'\colon \mathsf{Rel}(\omega_j'(\vec{\mu}), \kappa_j'(\vec{\nu})).$$

As in $\mathbf{Ctx}$ these morphisms are identified up to provable equivalence of the definable relations.

The fibre of $\mathbf{LinAdmRelations}$ over an object $\vec{\alpha}, \vec{\beta} \mid \Gamma \mid R\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$ in $\mathbf{AdmRelCtx}$ becomes:

Objects Equivalence classes of definable relations

$$\vec{\alpha}, \vec{\beta} \mid \vec{\Gamma} \mid R\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta})) \vdash \rho\colon \mathsf{AdmRel}(\sigma(\vec{\alpha}), \tau(\vec{\beta})).$$

Morphisms A morphism from $\rho\colon \mathsf{AdmRel}(\sigma(\vec{\alpha}), \tau(\vec{\beta}))$ to $\rho'\colon \mathsf{AdmRel}(\sigma'(\vec{\alpha}), \tau'(\vec{\beta}))$ is a pair of morphisms $t : \sigma \multimap \sigma'$, $u : \tau \multimap \tau'$ such that it is provable in the logic that:

$$\forall x\colon \sigma. \forall y\colon \tau.\, \rho(x, y) \supset \rho'(tx, uy)$$

We will construct the map J as a map of fibred linear categories from $\mathbf{LinType} \to \mathbf{Kind}$ to

$$\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$$

as follows. On the base categories J is defined on objects as

$$J(\alpha_1, \dots, \alpha_n) = \alpha_1, \dots, \alpha_n; \beta_1, \dots, \beta_n \mid R_1 \colon \mathsf{AdmRel}(\alpha_1, \beta_1), \dots, R_n \colon \mathsf{AdmRel}(\alpha_n, \beta_n).$$

We define J on the objects of the total categories (and on the morphisms of the base category) as

$$J(\vec{\alpha} \vdash \sigma \colon \mathsf{Type}) = \vec{\alpha}, \vec{\beta} \mid \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[R] \colon \mathsf{AdmRel}(\sigma(\vec{\alpha}), \sigma(\vec{\beta})).$$

To define J on morphisms of the fibre categories, suppose $\vec{\alpha} \mid -; - \vdash t \colon \sigma \multimap \tau$. We define $J(t) = (t, t)$. To see that (t, t) in fact is a map from $\sigma[\vec{R}]$ to $\tau[\vec{R}]$, notice that the Logical Relations Lemma (3.3) tells us that

$$\Lambda \vec{\alpha}.\, t (\textstyle\prod \vec{\alpha}.\, \sigma \multimap \tau) \Lambda \vec{\alpha}.\, t,$$

which means exactly that $(t, t) \colon \sigma[\vec{R}] \multimap \tau[\vec{R}]$.

Rules 2.19-2.24 tell us that J is a strict fibred symmetric monoidal closed functor preserving products and ! on the nose. Since the ϵ and δ of the fibred comonad on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ are simply (ϵ, ϵ) and (δ, δ) it is clear that J preserve these as well.

Now, by definition, a formula holds in this LAPL-structure iff it is provable LAPL. □

5 Parametric LAPL-structures

Definition 5.1. A **parametric LAPL-structure** is an LAPL-structure with very strong equality in which identity extension holds in the internal logic.

Recall that very strong equality implies extensionality. We ask that identity extension and extensionality hold because this means that all the results from Section 3 apply to the internal logic of the LAPL-structure. Strong equality will be used to conclude that properties proved in the internal logic also hold externally, as exemplified in the following subsection, where we show how to solve recursive domain equations in parametric LAPL-structures.

5.1 Solving recursive domain equations in parametric LAPL-structures

Definition 5.2. An endofunctor $T : \mathbb{B}^{\mathrm{op}} \times \mathbb{B} \to \mathbb{B}$, for $\mathbb{B}$ an SMCC, is called **strong** if there exists a natural transformation $t_{\sigma,\tau,\sigma',\tau'} : !(\sigma^{\sigma'}) \otimes !((\tau')^{\tau}) \multimap T(\sigma', \tau')^{T(\sigma,\tau)}$ preserving identity and composition:

$$\begin{tikzcd}
I \arrow[r, -o, "\widehat{!id_\sigma} \otimes \widehat{!id_\tau}"] \arrow[dr, -o, "\widehat{id_{T\sigma}}"'] & !(\sigma^\sigma) \otimes !(\tau^\tau) \arrow[d, -o, "t_{\sigma,\tau,\sigma,\tau}"] \\
& T(\sigma, \tau)^{T(\sigma,\tau)}
\end{tikzcd}
\qquad
\begin{tikzcd}
!(\sigma^{\sigma'}) \otimes !((\tau')^{\tau}) \otimes !((\sigma')^{\sigma''}) \otimes !((\tau'')^{\tau'}) \arrow[r, -o, "comp"] \arrow[d, -o, "t \otimes t"'] & !(\sigma^{\sigma''}) \otimes !((\tau'')^{\tau}) \arrow[d, -o, "t"] \\
T(\sigma', \tau')^{T(\sigma,\tau)} \otimes T(\sigma'', \tau'')^{T(\sigma',\tau')} \arrow[r, -o, "comp"] & T(\sigma'', \tau'')^{T(\sigma,\tau)}.
\end{tikzcd}$$

The natural transformation t is called the **strength** of the functor T. (Note that we here used exponential notation X^Y for the closed structure in $\mathbb{B}$.)

One should note that t in the definition above represents the morphism part of the functor T in the sense that it makes the diagram

$$\begin{tikzcd}
I \arrow[r, "!\widehat{f}\otimes!\widehat{g}", -{Circle[open]}] \arrow[dr, "\widehat{T(f,g)}"', -{Circle[open]}] & !(\sigma^{\sigma'})\otimes!((\tau')^{\tau}) \arrow[d, "t_{\sigma,\tau,\sigma',\tau'}", -{Circle[open]}] \\
 & T(\sigma',\tau')^{T(\sigma,\tau)}
\end{tikzcd} \tag{9}$$

commute, for any pair of morphisms $f\colon \sigma' \multimap \sigma, g\colon \tau \multimap \tau'$. This follows from the commutative diagram

$$\begin{tikzcd}
I \arrow[dr, "!\widehat{id}\otimes!\widehat{id}"', -{Circle[open]}] \arrow[drr, bend left, "\widehat{id}", -{Circle[open]}] \arrow[ddr, bend right, "!\widehat{f}\otimes!\widehat{g}"', -{Circle[open]}] & & \\
 & !\sigma^{\sigma}\otimes!\tau^{\tau} \arrow[r, "t", -{Circle[open]}] \arrow[d, "!(\sigma^{f})\otimes!(g^{\tau})", -{Circle[open]}] & T(\sigma,\tau)^{T(\sigma,\tau)} \arrow[d, "T(f,g)^{T(\sigma,\tau)}", -{Circle[open]}] \\
 & !\sigma^{\sigma'}\otimes!(\tau')^{\tau} \arrow[r, "t", -{Circle[open]}] & T(\sigma',\tau')^{T(\sigma,\tau)}.
\end{tikzcd}$$

In fact, (9) shows that we can define the action of T on morphisms from the strength.

Definition 5.3. If $\mathbb{E} \to \mathbb{B}$ is a fibred SMCC, then a fibred functor

$$\begin{tikzcd}
\mathbb{E}^{\mathrm{op}} \times_{\mathbb{B}} \mathbb{E} \arrow[rr, "T"] \arrow[dr] & & \mathbb{E} \arrow[dl] \\
 & \mathbb{B} &
\end{tikzcd}$$

is called **strong fibred** if there exists a fibred natural transformation t from

$$!((=)^{(-)})\otimes!((=')^{(-')}) \text{ to } T(-,=')^{T(=,-')}$$

satisfying commutativity of the two diagrams of Definition 5.2 in each fibre. The natural transformation t is called the **strength** of the functor T.

The fibred strength t is a natural transformation between two fibred functors

$$\begin{tikzcd}
\mathbb{E} \times_{\mathbb{B}} \mathbb{E}^{\mathrm{op}} \times_{\mathbb{B}} \mathbb{E}^{\mathrm{op}} \times_{\mathbb{B}} \mathbb{E} \arrow[rr] \arrow[dr] & & \mathbb{E} \arrow[dl] \\
 & \mathbb{B} &
\end{tikzcd}$$

For example, in the case of a PILL$_Y$-model, the interpretation of any inductively constructed type $\alpha, \beta \vdash \sigma(\alpha, \beta)$ with α occuring only negatively and β only positively induces a strong fibred functor, since as described in Section 3.8, for each such type we can define a term

$$- \vdash t\colon \prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to \sigma(\alpha,\beta) \multimap \sigma(\alpha',\beta')$$

The object part of the functor F is then defined as

$$F(\tau, \tau') = [\![\sigma(\tau, \tau')]\!]$$

and the strength s of the functor is defined as

$$(s_\Xi)_{\tau,\tau',\omega,\omega'} = [\![\Xi \vdash t\,\tau\,\tau'\,\omega\,\omega']\!].$$

The morphism part of the functor is induced by the strength.

In fact, an a sense these are the only fibred strong functors on PILL$_Y$-models.

Lemma 5.4. *Suppose*

$$\begin{tikzcd}
\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType} \arrow[rr, "F"] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

is a fibred strong functor on a PILL$_Y$-model. Then there exists a type $\alpha, \beta \vdash \sigma$ and a term

$$- \vdash t \colon \prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to \sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta')$$

in the internal language of $\mathbf{LinType} \to \mathbf{Kind}$ inducing F.

Proof. Let $T \colon \mathbf{LinType}_\Omega$ denote the generic object of the fibration $\mathbf{LinType} \to \mathbf{Kind}$. For each type $\tau \colon \mathbf{LinType}_\Xi$ there exists a map $\Xi \to \Omega$, which we will denote $\hat{\tau}$ such that $\hat{\tau}^* T = \tau$.

Define $\sigma = F([\![\alpha, \beta \vdash \alpha]\!], [\![\alpha, \beta \vdash \beta]\!])$. Then for any pair of types $(\tau, \tau') \in (\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})_\Xi$

$$F(\tau, \tau') = F(\langle \hat{\tau}, \hat{\tau'} \rangle^* ([\![\alpha, \beta \vdash \alpha]\!], [\![\alpha, \beta \vdash \beta]\!])) = \langle \hat{\tau}, \hat{\tau'} \rangle^* \sigma$$

since F is fibred. In the internal language $\Xi \vdash \sigma(\tau, \tau')$ is interpreted as $\langle \hat{\tau}, \hat{\tau'} \rangle^* \sigma$ and so indeed σ induces the action of F on objects.

Let s denote the strength of the fibred functor F. Consider the component

$$(s_{\Omega^4})_{[\![\alpha,\beta,\alpha',\beta' \vdash \alpha]\!], [\![\alpha,\beta,\alpha',\beta' \vdash \beta]\!], [\![\alpha,\beta,\alpha',\beta' \vdash \alpha']\!], [\![\alpha,\beta,\alpha',\beta' \vdash \beta']\!]}$$

and denote it by t'. In the internal language, t' is a term, and we can consider the polymorphic term

$$- \vdash t = \Lambda\alpha.\, \Lambda\beta.\, \Lambda\alpha'.\, \Lambda\beta'.\, t' \colon \prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to (\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta')).$$

We just need to show that the strength induced by the term t is in fact s, but

$$[\![\Xi \vdash t\,\tau\,\tau'\,\omega\,\omega']\!] = \langle \hat{\tau}, \hat{\tau'}, \hat{\omega}, \hat{\omega'} \rangle^* t' = (s_\Xi)_{\tau,\tau',\omega,\omega'}$$

since s is preserved by reindexing. □

Theorem 5.5. *In a parametric LAPL-structure, for any strong fibred functor F there exists a closed type τ such that $F(\tau, \tau) \cong \tau$ in $\mathbf{LinType}_1$ for 1 the terminal object of $\mathbf{Kind}$.*

Proof. Since by Lemma 5.4 we can express F in the internal language, we can copy the proofs from Sections 3.10, 3.11, 3.12. The functorial interpretation of types of Section 3.8 should be substituted by the polymorphic term provided by Lemma 5.4. Since internal equality implies external equality, we get the result in the “real world”. □

Remark 5.6. Since the functor F is fibred, we may reindex τ to get a family of objects $(!^*_\Xi \tau)_{\Xi \in \mathbf{Kind}}$ such that for each Ξ, $!^*_\Xi \tau$ satisfies $F(!^*_\Xi \tau, !^*_\Xi \tau) \cong !^*_\Xi \tau$ in the fibre, where $!_\Xi$ is the unique map $\Xi \to 1$ in $\mathbf{Kind}$.

Remark 5.7. Parametric LAPL-structures do not in general model recursive types, that is, we do not have for all types $\sigma \colon \mathbf{LinType}_\Omega$ a type τ such that $\sigma(\tau) \cong \tau$, since there may not be a functorial action corresponding to σ.

5.2 Parametrized recursive type equations

An inductively constructed type $\alpha_1, \beta_1, \ldots, \alpha_n, \beta_n \vdash \sigma$ with $2n$ free type variables, in which the variables $\vec{\alpha}$ occur only negatively and the variables $\vec{\beta}$ only positively induces a fibred functor

$$\begin{tikzcd}
(\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})^n \arrow[rr, "F"] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind}. &
\end{tikzcd}$$

On the other hand Definition 5.3 can easily be extended to define what it means that a functor F as above is strong fibred, and Lemma 5.4 extends to show that such strong fibred functors correspond to types σ as above and closed polymorphic terms of type

$$\prod \vec{\alpha}, \vec{\beta}, \vec{\alpha}', \vec{\beta}'. (\alpha_1' \multimap \alpha_1) \to (\beta_1 \multimap \beta_1') \to \ldots \to (\alpha_n' \multimap \alpha_n) \to (\beta_n \multimap \beta_n') \to \sigma(\vec{\alpha}, \vec{\beta}) \multimap \sigma(\vec{\alpha}', \vec{\beta}')$$

in the internal language. The following theorem is then the corresponding extension of Theorem 5.5 obtained using the analysis of Section 3.13.

Theorem 5.8. *In a parametric LAPL-structure, for any strong fibred functor*

$$\begin{tikzcd}
(\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})^{n+1} \arrow[rr, "F"] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind}. &
\end{tikzcd}$$

there exists a strong fibred functor

$$\begin{tikzcd}
(\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})^{n} \arrow[rr, "\mathit{Fix}F"] \arrow[dr] & & \mathbf{LinType} \arrow[dl] \\
& \mathbf{Kind}. &
\end{tikzcd}$$

and a fibred natural isomorphism

$$\begin{array}{ll} & F(X_1, Y_1, \ldots, X_n, Y_n, \mathit{Fix}F(Y_1, X_1, \ldots, Y_n, X_n), \mathit{Fix}F(X_1, Y_1, \ldots, X_n, Y_n)) \\ \cong & \mathit{Fix}F(X_1, Y_1, \ldots, X_n, Y_n). \end{array}$$

Moreover, if G is a strong fibred functor

$$\begin{tikzcd}
(\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})^{m} \arrow[rr, "G"] \arrow[dr] & & (\mathbf{LinType}^{\mathrm{op}} \times_{\mathbf{Kind}} \mathbf{LinType})^{n} \arrow[dl] \\
& \mathbf{Kind}. &
\end{tikzcd}$$

then $\mathit{Fix}(F \circ (G \times id)) = \mathit{Fix}F \circ G$.

6 Concrete Models

In this section we describe a parametric LAPL structure based on admissible pers over a universal domain as advocated by Plotkin [22]. Pers are known to model the typed λ-calculus, and admissible pers further facilitates a fixed point operator.

As noted in Section 4, to model PILL one has to provide a fibred symmetric monoidal adjunction. We do this by constructing a regular symmetric monoidal adjunction and then define the fibration pointwise.

The standard example is a lifting functor and a forgetful functor. This is also the case here albeit slightly obfuscated, as lifting is coded in the language of the universal domain. This is an adaptation of [13].

Finally, to be able to model polymorphism, the entire construction is done fibred. Parametricity is then ensured by a yet-to-be-described completion process, but first we present the "clean" version:

Let D be a reflexive cpo, i.e. a pointed ω-chain-complete partial order such that we have

$$\Phi\colon D \to [D \to D] \qquad \text{and} \qquad \Psi\colon [D \to D] \to D,$$

both Scott-continuous and satisfying

$$\Phi \circ \Psi = id_{[D\to D]}$$

where $[D \to D]$ denotes the cpo of continuous functions from D to D. We assume, without loss of generality, that both Φ and Ψ are strict. It is standard that there exists strict continuous functions

$$\langle\cdot,\cdot\rangle\colon D \times D \to D, \qquad \pi\colon D \to D \qquad \text{and} \qquad \pi'\colon D \to D,$$

such that for all $d, d' \in D$:

$$\pi\langle d, d'\rangle = d \qquad \text{and} \qquad \pi'\langle d, d'\rangle = d'.$$

We use $\imath$ to denote $\Psi(id_{[D\to D]})$. Notice that $\Phi(\imath) = id_{[D\to D]}$.

Definition 6.1. An *admissible partial equivalence relation on* D is a partial equivalence relation R on D satisfying

strict $\perp_D R \perp_D$,

ω**-chain complete** For $(x_n)_{n\in\omega}$ and $(y_n)_{n\in\omega}$ ω-chains in D:

$$(\forall n \in \omega . x_n \; R \; y_n) \Rightarrow \bigsqcup_{n\in\omega} x_n \; R \bigsqcup_{n\in\omega} y_n,$$

Definition 6.2. For R and S pers on D, define the set of **equivariant functions from** R **to** S as

$$\mathcal{F}(R,S) = \{f \in [D \to D] | x \; R \; y \Rightarrow f(x) \; S \; f(y)\}$$

and the set of **strict equivariant functions from** R **to** S as

$$\mathcal{F}(R,S)_\perp = \{f \in \mathcal{F}(R,S) | f(\perp_D) = \perp_D\}.$$

Note $\mathcal{F}(R,S)_\perp \subseteq \mathcal{F}(R,S)$.

Definition 6.3. For R and S pers on D, define on $\mathcal{F}(R,S)$ or $\mathcal{F}(R,S)_\perp$ the equivalence relation $\simeq_{R,S}$ by

$$f \simeq_{R,S} g \Leftrightarrow \forall d \in D.\; d \; R \; d \Rightarrow f(d) \; S \; g(d)$$

We write $\mathbf{PER}(D)$ for the category of partial equivalence relations over D. Recall that it has partial equivalence relations over D as objects and that a morphism $[f]\colon R \to S$ is an equivalence class in $\mathcal{F}(R,S)/\simeq_{R,S}$. Elements of $[f]$ are called **realizers** for $[f]$.

Definition 6.4. We define the category $\mathbf{AP}(D)$ of admissible partial equivalence relations over D as the full subcategory of $\mathbf{PER}(D)$ on the admissible pers.

Lemma 6.5. *There is a faithful functor $Classes\colon \mathbf{AP}(D) \to \mathbf{Set}$ mapping an admissible per to the set of equivalence classes and an equivalence class of realizers to the map of equivalence classes they induce.*

Proof. This is well-defined since to realizers are equivalent precisely when they define the same map of equivalence classes. □

Theorem 6.6. *The category $\mathbf{AP}(D)$ is a sub-cartesian closed category of $\mathbf{PER}(D)$.*

Proof. We recall the constructions. It is straightforward to verify that the resulting pers are admissible. The terminal object 1 is the admissible per defined by

$$d \; 1 \; d' \Leftrightarrow d = \bot_D = d'.$$

The binary product of R and S is

$$\begin{array}{c} \langle d_1, d_2 \rangle \; R \times S \; \langle d_1', d_2' \rangle \\ \Updownarrow \\ d_1 \; R \; d_1' \quad \wedge \quad d_2 \; S \; d_2' \end{array}$$

This is an exhaustive description, understood that only pairs are related in the product. The exponential of R and S, S^R, is given by

$$d \; S^R \; d' \quad \Leftrightarrow \quad \Phi(d) \simeq_{R,S} \Phi(d').$$

□

Definition 6.7. The category $\mathbf{AP}(D)_\bot$ of admissible pers and strict continuous functions is the full-on-objects subcategory of $\mathbf{AP}(D)$ with morphisms $[f]\colon R \to S$ equivalence classes in $\mathcal{F}(R,S)_\bot/\simeq_{R,S}$.

Note that in $\mathbf{AP}(D)_\bot$, morphisms are required to have a *strict* continuous realizer.

Theorem 6.8. *$\mathbf{AP}(D)_\bot$ is a cartesian sub-category of $\mathbf{AP}(D)$.*

Proof. Obvious using that π, π', and $\langle \cdot, \cdot \rangle$ are strict. □

Theorem 6.9. *The category $\mathbf{AP}(D)_\bot$ is symmetric monoidal closed.*

Proof. The tensor of R and S is

$$\begin{array}{c} \langle d_1, d_2 \rangle \; R \otimes S \; \langle d_1', d_2' \rangle \\ \Updownarrow \\ \langle d_1, d_2 \rangle \; R \times S \; \langle d_1', d_2' \rangle \\ \vee \\ \left(\begin{array}{c} d_1 \; R \; d_1 \quad \wedge \quad d_2 \; S \; d_2 \quad \wedge \quad d_1' \; R \; d_1' \quad \wedge \quad d_2' \; S \; d_2' \quad \wedge \\ (d_1 \; R \; \bot_D \quad \vee \quad d_2 \; S \; \bot_D) \quad \wedge \quad (d_1' \; R \; \bot_D \quad \vee \quad d_2' \; S \; \bot_D) \end{array} \right) \end{array}$$

This complicated looking definition is most easily understood through the functor $Classes$: The equivalence classes of the tensor product are those of the product with the modification that all pairs where one of the coordinates are related to $\perp_D$ has been gathered in one big equivalence class. It can thus be seen as a quotient of the product.

The unit of the tensor I is defined by

$$dId' \quad \Leftrightarrow \quad d = d' = \perp_D \vee d = d' = \langle \mathrm{1}, \perp_D \rangle .$$

This definition is not taken out of the blue. I is actually in the image of a lifting functor to be defined later. Notice the “if construct” on I, which will be available on all lifted relations:

$$\begin{array}{rcl} d\; I\; d & \Rightarrow & d = \perp_D \quad \vee \quad d = \langle \mathrm{1}, \perp_D \rangle \\ & \Rightarrow & \pi(d) = \perp_D \quad \vee \quad \pi(d) = \mathrm{1} \\ & \Rightarrow & \Phi(\pi(d)) = \perp_{[D \to D]} \quad \vee \quad \Phi(\pi(d)) = id_{[D \to D]} \end{array}$$

Thus for $d\ I\ d$, $\Phi(\pi(d))(d')$ can be read as “if $d \neq \perp_D$ then d' else $\perp_D$”. We will use this to construct realizers.

The exponential of R and S, $R \multimap S$, is given by

$$d\, R \multimap S\, d' \quad \Leftrightarrow \quad d\, S^R\, d' \;\wedge\; (d''\, R \perp_D \Rightarrow \Phi(d)(d'')\; S \perp_D S\; \Phi(d')(d''))$$

The proof consist of a series of straightforward verifications. □

For later use we shall mention how regular subobjects look in this category. We use $A \rightarrowtail R$ to express that A is a regular subobject of R, if R is an admissible per.

Lemma 6.10.

$$A \rightarrowtail R \Leftrightarrow Classes(A) \subseteq Classes(R) \wedge A \in obj(\mathbf{AP}(D)_\perp)$$

Proof. In $\mathbf{PER}(D)$ there is a standard way of constructing an equalizer out of a subset of the equivalence classes. This also works here, and the image of an equalizer is easily seen to be admissible. Thus all regular subobjects have a representative, which is tracked by the identity on D. □

We also need to know the following fact about admissible pers

Lemma 6.11. *If I is an arbitrary set, and for all $i \in I$, R_i is an admissible per over D then*

$$\bigcap_{i \in I} R_i$$

is an admissible per over D.

Proof. We intersect relations, which may be seen as sets of pairs. Thus we have the following equation

$$d \bigcap_{i \in I} R_i\; d' \Leftrightarrow \forall i \in I.\, d\; R_i\; d'$$

which makes the statement obvious, as all R_i are admissible. □

6.1 The connection to CUPERs

In [1] Amadio and Curien show how complete uniform pers over a universal domain allows on to solve domain equations on the per level. As we claim, the same is true for admissible pers, a comparison is natural.

There are, however, some technical issues which makes this a little difficult.

Cupers are defined over a universal solution to the domain equation

$$D = T(D) = (D \rightharpoonup D) + (D \times D)$$

In the category CPO^{ip} of pointed directed complete partial orders and injection-projection pairs. It is known that D is then the colimit of the ω^{op}-chain

$$\begin{tikzcd} \bot = 0 \arrow[r, "!"] & T0 \arrow[r, "T!"] & T^2 0 \arrow[r, "T^2!"] & \cdots \end{tikzcd}$$

Denoting $T^n 0$ by D_n we have by the cocone property fo each n the diagram:

$$\begin{tikzcd} D_n \arrow[r, bend right, "i_n"'] \arrow[r, phantom, "\lhd"] & D \arrow[l, bend right, "j_n"'] \end{tikzcd}$$

Defining $p_n = i_n \circ j_n \colon D \to D$ we can define a cuper on D as a relation $R \subseteq D \times D$ such that

- If $A \subseteq R$ and $A \subseteq_{dir} D \times D$ then $\bigsqcup A \in R$.
- If $d \; R \; e$ then for all n, $p_n(d) \; R \; p_n(e)$.

So apart from using directed-complete rather than chain-complete cpos and living on a universal domain solving a slightly different domain equation, cupers live on a domain with a known structure, and this structure appears in their definition.

Thus the only reasonable way to compare the two notions is to consider a suitably adapted notion of admissible pers over the above described D. We then find, that the cupers form a proper subset of the admissible pers.

It is noticeable, that cupers facilitate an ordering of the equivalence classes and thus allows one to solve recursive domain equations, while admissible pers achieve this by modeling polymorphic λ-calculus and calling upon parametricity[2]. Hence the two approaches are somewhat different.

6.2 Lifting

We now define a notion of lifting, to establish an adjunction between $\mathbf{AP}(D)$ and $\mathbf{AP}(D)_\bot$. Our notion of lifting is essentially the one in [13], specialized to the partial combinatory algebra defined by D, Φ and Ψ.

Let PD denote the power set of D. Define the map $L_0' \colon PD \to PD$ by

$$L_0'(A) = \{d \in D | \pi(\Phi(d)(\imath)) = \imath \wedge \pi'(\Phi(d)(\imath)) \in A\},$$

for $A \subseteq D$. And then the map $L_0 \colon \mathbf{AP}(D)_0 \to (\mathbf{AP}(D)_\bot)_0$ by

$$Classes(L_0(R)) = \{L_0'(K) | K \in D/R\} \cup \{\{\bot_D\}\}.$$

[2] And calling upon parametricity is, as far as we know, only possible after the deployment of a parametric completion process.

Notice the “if construct” available on a liftet relation: If R is an admissible per then

$$\begin{aligned} d\ L_0(R)\ d &\Rightarrow\ d = \bot_D \quad \vee \quad \pi(\Phi(d)(\mathsf{1})) = \mathsf{1} \\ &\Rightarrow\ \pi(\Phi(d)(\mathsf{1})) = \bot_D \quad \vee \quad \pi(\Phi(d)(\mathsf{1})) = \mathsf{1} \\ &\Rightarrow\ \Phi(\pi(\Phi(d)(\mathsf{1}))) = \bot_{[D\to D]} \quad \vee \quad \Phi(\pi(\Phi(d)(\mathsf{1}))) = id_{[D\to D]} \end{aligned}$$

Thus $\Phi(\pi(\Phi(d)(\mathsf{1})))(d')$ can be read “if $d \notin \bot_{L_0(R)}$ then d' else $\bot_D$”, where $\bot_D$ of course represents $\bot_S$ for any admissible per S.

We also have a “lift” and an “unlift”: If $d \in A$ then $\Psi(\lambda d' \in D.\langle \mathsf{1}, d\rangle) \in L_0(A)$ and if $d \in L_0(A)$ then $\pi'(\Phi(d)(\mathsf{1})) \in A$. This is convenient for constructing realizers.

Similarly define, for admissible pers R and S, the map $L_1'\colon \mathcal{F}(R,S) \to \mathcal{F}(L_0(R), L_0(S))_\bot$ by

$$L_1'(f) = \lambda d \in D.\Phi(\pi(\Phi(d)(\mathsf{1})))(\Psi(\lambda d' \in D.\langle \mathsf{1}, f(\pi'(\Phi(d)(\mathsf{1})))\rangle))$$

which reads “if $d \notin \bot_{L_0(R)}$ then lift(f(unlift d)) else $\bot_D$”.

And then the map $L_1\colon (\mathbf{AP}(D))_1 \to (\mathbf{AP}(D)_\bot)_1$ by

$$L_1(f) = [L_1'(t_f)]_{\simeq_{L_0(R),L_0(S)}}$$

for $f\colon R \to S$. A tedious, but straightforward, verification shows that the definitions of L_0', L_0, L_1' and L_1 all make sense, and that $L = (L_0, L_1)\colon \mathbf{AP}(D) \to \mathbf{AP}(D)_\bot$ defines a functor. There is an obvious forgetful functor $U\colon \mathbf{AP}(D)_\bot \to \mathbf{AP}(D)$.

Theorem 6.12. *There is a monoidal adjunction $L \dashv U$.*

Proof. One first shows that L is left adjoint to U in the ordinary sense. The unit of the adjunction is given by $(\eta_R\colon R \to UL(R))_{R\in APD_0}$, all tracked by

$$t_\eta = \lambda d \in D.\Psi(\lambda d' \in D.\langle \mathsf{1}, d\rangle).$$

For $f\colon R \to U(S)$ in $\mathbf{AP}(D)_\bot$, the required unique $h\colon L(R) \to S$ in $\mathbf{AP}(D)_\bot$, such that $U(h) \circ \eta_R = f$, is given by the realizer

$$t_h = \lambda d \in D.\Phi(\pi(\Phi(d)(\mathsf{1})))(t_f(\pi'(\Phi(d)(\mathsf{1})))),$$

where t_f is a realizer for f.

To show that the adjunction is monoidal it suffices by to show that the left adjoint L is a strong symmetric monoidal functor (see [17] for an explanation). To this end, we must exhibit an isomorphism $m_I\colon I \to L(1)$ and a natural isomorphism $m_{R,S}\colon L(R)\otimes L(S) \to L(R\times S)$. This is mostly straightforward; we just include the definition of $m_{R,S}\colon L(R) \otimes L(S) \to L(R \times S)$: it is the morphism tracked by the realizer

$$\begin{array}{l} \lambda d \in D. \\ \Phi(\pi(\Phi(\pi(d))(\mathsf{1})))(\\ \qquad \Phi(\pi(\Phi(\pi'(d))(\mathsf{1})))(\\ \qquad\qquad \Psi(\lambda d' \in D.\langle \mathsf{1}, \langle \pi'(\Phi(\pi(d))(\mathsf{1})), \pi'(\Phi(\pi'(d))(\mathsf{1}))\rangle\rangle) \\ \qquad) \\ ) \end{array}$$

which reads

```
“if $\pi(d) \neq \bot$ then
        if $\pi'(d) \neq \bot$ then
                lift of $\langle$unlift($\pi(d)$), unlift($\pi'(d)$)$\rangle$
        else $\bot_D$
 else $\bot_D$”.
```

Following a similar chain of thought, the inverse is tracked by

$$\begin{array}{l}
\lambda d \in D. \\
\Phi(\pi(\Phi(d)(\imath)))(\\
\qquad \langle \Psi(\lambda d' \in D. \langle \imath, \pi(\pi'(\Phi(d)(\imath)))\rangle), \Psi(\lambda d' \in D. \langle \imath, \pi'(\pi'(\Phi(d)(\imath)))\rangle)\rangle \\
)
\end{array}$$

which reads

```
“if $d \neq \bot$ then
        $\langle$lift of $\pi$(unlift($d$)), lift of $\pi'$(unlift($d$))$\rangle$
 else $\bot_D$”.
```

□

6.3 Going fibred

In order to model polymorphism, we do a fibred version of the adjunction presented in the last subsection, thus arriving at the PILL-model

$$\begin{tikzcd}
\mathbf{UFam}(\mathbf{AP}(D)_\bot) \arrow[rr, bend right=5, "U"'] \arrow[rr, phantom, "\perp"] \arrow[dr, "q"'] & & \mathbf{UFam}(\mathbf{AP}(D)) \arrow[ll, bend right=5, "L"'] \arrow[dl, "p"] \\
& \mathbf{Set}. &
\end{tikzcd} \tag{10}$$

Define the contravariant functor $P : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ by mapping set I to the category $P(I)$ with

objects: $(R_i)_{i \in I}$ where for all $i \in I$, R_i is an object of $\mathbf{AP}(D)$.

morphisms: $(\alpha_i)_{i \in I} \colon (R_i)_{i \in I} \to (S_i)_{i \in I}$, where, for all $i \in I$, $\alpha_i \in \mathbf{AP}(D)(R_i, S_i)$ and $\exists \alpha \in [D \to D]. \forall i \in I. \alpha_i = [\alpha]_{\simeq_{R_i,S_i}}$.

For a function $f \colon I \to J$, the reindexing functor $P(f)$ is simply given by composition with f.
Define the contravariant functor $Q : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ given by mapping set I to the category $Q(I)$ with

objects: $(R_i)_{i \in I}$ where for all $i \in I$, R_i is an object of $\mathbf{AP}(D)_\bot$.

morphisms: $(\alpha_i)_{i \in I} \colon (R_i)_{i \in I} \to (S_i)_{i \in I}$ where for all $i \in I$, $\alpha_i \in \mathbf{AP}(D)_\bot(R_i, S_i)$ and $\exists \alpha \in [D \to D]. \forall i \in I. \alpha_i = [\alpha]_{\simeq_{R_i,S_i}}$.

For a function $f\colon I \to J$, the reindexing functor $Q(f)$ is again simply given by composition with f.

That we have two contravariant functors is obvious. The Grothendieck construction then gives us two split fibrations, $p\colon \mathbf{UFam}(\mathbf{AP}(D)) \to \mathbf{Set}$ and $q\colon \mathbf{UFam}(\mathbf{AP}(D)_\perp) \to \mathbf{Set}$. The functors L and U easily lift to fibred functors between these two fibrations (we abuse notation and also denote the fibred functors by L and U). Explicitly, on objects $L(I, (R_i)_{i\in I}) = (I, (L(R_i))_{i\in I})$ and on vertical morphisms $L(I, (f_i)_{i\in I}) = (I, (L(f_i))_{i\in I})$. Likewise for U. These are not recursive definitions, they simply look so because of the reuse of letters.

Theorem 6.13. *L and U are split fibred functors and $L \dashv U$ is a split fibred strong monoidal adjunction*

Proof. It is obvious that L and U are split fibred functors; the second part follows immediately from Theorem 6.12. □

6.4 A domain-theoretic model of PILL

To show that (10) is a model of PILL it remains to be shown that q has a generic object and simple products.

Lemma 6.14. *The set $\Omega = \mathrm{Obj}(\mathbf{AP}(D)_\perp) = \mathrm{Obj}(\mathbf{AP}(D))$ is a split generic object of the fibration q. The fibration q has simple split Ω-products satisfying the Beck-Chevalley condition.*

Proof. The first part is obvious. For the second part, one uses the usual definition for uniform families of ordinary pers and verifies that it restricts to admissible pers: We recall from [12] that given any projection $\pi_A\colon A \times \Omega \to A$ in $\mathbf{Set}$, the right adjoint $\forall_A$ to π_A^* is given on objects by intersection:

$$\forall_A((R_{(a,\omega)})_{(a,\omega)\in A\times\Omega}) = (\bigcap_{\omega\in\Omega} R_{(a,\omega)})_{a\in A}.$$

By lemma 6.11 the resulting per is admissible. □

Theorem 6.15. *The diagram* (10) *constitutes a model of PILL$_Y$.*

Proof. Given the preceding results it only remains to verify that (1) the structure in the diagram models the polymorphic fixed point combinator and that (2) $\mathbf{UFam}(\mathbf{AP}(D))$ is equivalent to the category of products of free coalgebras of $\mathbf{UFam}(\mathbf{AP}(D))_\perp$.

For (1), the required follows, as expected, because the pers are strict and complete. In more detail, the reasoning is as follows: It is well-known that there is a Scott-continuous function $y \in [[D \to D] \to D]$ giving fixed-points through iterated application at $\perp_D$. Since realizers are Scott-continuous functions in $[D \to D]$, every realizer α has a fixed-point $y(\alpha)$ in D given by $\bigsqcup_n (\alpha^n)(\perp_D)$. If for some admissible per R, $\alpha \in \mathcal{F}(R, R)$, then, since R is strict, α respects R, and R is chain-complete, $y(\alpha)\ R\ y(\alpha)$. Thus the equivalence class of $y(\alpha)$ exists and is a fixed-point of the morphism represented by α. This is applicable both in $\mathbf{AP}(D)$ and $\mathbf{AP}(D)_\perp$, but is not so interesting on strict morphisms. It is, however, in $\mathbf{AP}(D)_\perp$ that we model the calculus, and thus here we want a fixed-point combinator — albeit only for some morphisms, namely those of type $!R \multimap R = R \to R$, corresponding to morphisms of $\mathbf{AP}(D)$. Intuitively, we wish to take such a morphism, transpose it, grab the fixed-point in $\mathbf{AP}(D)$ and call the whole process a morphism in $\mathbf{AP}(D)_\perp$. This is possible, since transposition cascades to the level of realizers. The function that transposes a morphism and returns the fixed-point of the result is continuous. The fixed-point function is in $\mathcal{F}(R \to R, R)$, for any R, and thus its code is a member of $\forall\alpha\mathsf{Type}.(\alpha \to \alpha) \multimap \alpha$. Precomposing with a uniform realizer for ϵ before taking the code, one easily obtains the polymorphic fixed-point combinator $Y\colon \forall\alpha\mathsf{Type}.(\alpha \to \alpha) \to \alpha$. Writing this out, one arrives at

$$\Psi(\lambda d \in D.y(\lambda d' \in D.\Phi(\Phi(\pi(\Phi(d)(\imath)))(\pi'(\Phi(d)(\imath))))(\langle \imath, d'\rangle))))$$

For (2), observe that by [17, Proposition 1.21] applied to Theorem 6.8 it suffices to show that $\mathbf{UFam}(\mathbf{AP}(D))$ is equivalent to the coKleisli category of the adjunction $L \dashv U$, but this follows from the fact that U is a forgetful functor. □

6.5 A parametric domain-theoretic model of PILL

In this section, we introduce a parametric version of the thus far constructed model. It is essentially obtained through a parametric completion process such as the one described in [5] for internal λ_2 models (as mentioned in the Introduction, we will generalize that completion process to produce parametric LAPL-structures in [16]).

We will arrive at the diagram

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[rr, bend right=8, "U"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[ll, bend right=8, "L"'] \arrow[dl] \\
& \mathbf{PAP}(D) &
\end{tikzcd} \tag{11}$$

Our construction is based on reflexive graphs and since our strategy is to obtain relational parametricity for admissible relations (to also model the fixed point combinator in the parametric model), we consider the set $\mathrm{RefGrph}$ of diagrams

$$A \rightarrowtail R \times S,$$

where A is a regular subobject of $R \times S$ in $\mathbf{AP}(D)_\perp$. (It is crucial that subobject is in the category with *strict* maps — it means that A will relate the equivalence class of $\perp$ in R to the equivalence class of $\perp$ in S.

Identifying $\mathrm{RefGrph}^n$ with n, we define the base category $\mathbf{PAP}(D)$ by

Objects: $n \in N$ — objects are natural numbers.

Morphisms: $f\colon n \to m$ is an m-tuple, $(f_1, \dots, f_m)$, where each f_i is a pair (f_i^p, f_i^r) satisfying

- f_i^p is a map of objects $(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n \to \mathrm{Obj}(\mathbf{AP}(D)_\perp)$
- f_i^r is a map, that to two vectors of objects of $\mathbf{AP}(D)_\perp$ associates maps of subobjects

  $$f_i^r \in \Pi_{\vec{R},\vec{S}\in(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n}\left(\Pi_{j\in\{1,\dots,n\}}\mathrm{RegSub}(R_j \times S_j) \to \mathrm{RegSub}(f_i^p(\vec{R}) \times f_i^p(\vec{S}))\right)$$

  satisfying

  $$\forall \vec{R} \in (\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n.f_i^r(\vec{R},\vec{R})(\vec{Eq_{R_j}}) = Eq_{f_i^p(\vec{R})},$$

where the regular subobjects are to be calculated in $\mathbf{AP}(D)_\perp$.

We now describe $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PAP}(D)$ and $\mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PAP}(D)$. As objects they basically contain an (indexed) per and an (indexed) relational interpretation of this per. As morphisms they have uniformly tracked morphisms that respect admissible relations. We wish to model admissible relations as regular subobjects in $\mathbf{AP}(D)_\perp$, so we introduce the notation $A \rightarrowtail R$ for $A \in \mathrm{Obj}(\mathrm{RegSub}_{\mathbf{AP}(D)_\perp}(R))$. We plan to use the Grothendieck construction, and so define indexed categories: $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ is defined with

Objects: $f\colon n \to 1$ is a morphism in $\mathbf{PAP}(D)$ from n to 1.

Morphisms: $\alpha\colon f \to g$ is a uniformly tracked family of morphisms $(\alpha_{\vec{R}})_{\vec{R}\in(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n}$ of $\mathbf{AP}(D)_\perp$ such that

$$\alpha_{\vec{R}}\colon f^p(\vec{R}) \to g^p(\vec{R}).$$

That α is uniformly tracked means that there is a strict continuous function $t_\alpha \in [D \to D]$ such that

$$\forall \vec{R} \in (\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n . \alpha_{\vec{R}} = {}_{f^p(\vec{R})}[t_\alpha]_{g^p(\vec{R})}.$$

Furthermore this α should respect relations:

$$\forall \vec{A} \rightarrowtail \vec{R} \times \vec{S}. \langle a, b\rangle \; f^r(\vec{R}, \vec{S}, \vec{A}) \; \langle a, b\rangle \Rightarrow \langle t_\alpha(a), t_\alpha(b)\rangle \; g^r(\vec{R}, \vec{S}, \vec{A}) \; \langle t_\alpha(a), t_\alpha(b)\rangle.$$

Quite similarly $(\mathbf{PFam}(\mathbf{AP}(D)))_n$ is defined as the category with

Objects: $f\colon n \to 1$ is a morphism in $\mathbf{PAP}(D)$ from some n to 1.

Morphisms: $\alpha\colon f \to g$ is a uniformly tracked family of morphisms $(\alpha_{\vec{R}})_{\vec{R}\in(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n}$ of $\mathbf{AP}(D)$ such that

$$\alpha_{\vec{R}}\colon U(f^p(\vec{R})) \to U(g^p(\vec{R}))$$

where $U\colon \mathbf{AP}(D)_\perp \to \mathbf{AP}(D)$ is the forgetful functor. That we now ask for morphisms of $\mathbf{AP}(D)$ removes the demand, that the uniform tracker be strict. Again this α should respect relations:

$$\forall \vec{A} \rightarrowtail \vec{R} \times \vec{S}. \langle a, b\rangle \; f^p(\vec{R}, \vec{S}, \vec{A}) \; \langle a, b\rangle \Rightarrow \langle t_\alpha(a), t_\alpha(b)\rangle \; g^p(\vec{R}, \vec{S}, \vec{A}) \; \langle t_\alpha(a), t_\alpha(b)\rangle$$

Here A is still a regular subobject in $\mathbf{AP}(D)_\perp$.

Note that the only difference between the two definitions is the choice of category in which the $\alpha_{\vec{R}}$ are required to be morphisms.

Definition 6.16. Define $\mathbb{L}\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PFam}(\mathbf{AP}(D)_\perp)$ on

objects by

$$\mathbb{L}((f^p, f^r)) = (F^p, F^r)$$

where

$$F^p(\vec{R}) = L(f^p(\vec{R}))$$

and

$$F^r((\vec{R}, \vec{S}, \vec{A})) = L(f^r(\vec{R}, \vec{S}, \vec{A}))$$

morphisms by

$$\mathbb{L}(\alpha\colon (f^p, f^r) \to (g^p, g^r))(R) = L(\alpha(R))$$

Define $\mathbb{U}\colon \mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PFam}(\mathbf{AP}(D))$ in a similar way using U instead of L.

Lemma 6.17. *If $A \rightarrowtail R \times S$, then $L(A) \rightarrowtail L(R) \times L(S)$.*

Lemma 6.18. *$\mathbb{L}\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PFam}(\mathbf{AP}(D)_\perp)$ and $\mathbb{U}\colon \mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PFam}(\mathbf{AP}(D))$ are both functors, and $\mathbb{L} \dashv \mathbb{U}$*

Proof. Easy given lemma 6.17 and the fact that for all admissible pers R, $L(Eq_R) = Eq_{L(R)}$. Lemma 6.17 ensures that the realizer t_η for the unit of $L \dashv U$ also defines a natural transformation $id \Rightarrow \mathbb{UL}$ with the required universal property. □

By an easy extension of Theorem 6.6, we have:

Theorem 6.19. *$\mathbf{PFam}(\mathbf{AP}(D))$ is fibred cartesian closed.*

Proof. It turns out to be easy, since the product of two regular subobjects turns out to be a regular subobject of the product, and the exponent of two regular subobjects turns out to be a regular subobject of the exponent. Since the adjunction works on the level of realizers and realizers are uniform, the adjunction holds. □

Theorem 6.20. *$\mathbf{PFam}(\mathbf{AP}(D)_\perp)$ is fibred cartesian and fibred symmetric monoidal closed.*

Proof. We just present the SMCC structure: The tensor product of (f^p, f^r) and (g^p, g^r) in the fibre

$$(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n,$$

is denoted by $(f^p, f^r) \otimes (g^p, g^r)$ and defined by

$$(f^p, f^r) \otimes (g^p, g^r) = (f^p \otimes g^p, f^r \otimes g^r),$$

where

$$(f^p \otimes g^p)(\vec{R}) = f^p(\vec{R}) \otimes g^p(\vec{R})$$

and $(f^r \otimes g^r)(\vec{R}, \vec{S})(\vec{A})$ is defined as the image of the map $f^r(\vec{R}, \vec{S})(\vec{A}) \otimes g^r(\vec{R}, \vec{S})(\vec{A}) \to f^p(\vec{R}) \otimes g^p(\vec{R}) \times f^p(\vec{S}) \otimes g^p(\vec{S})$ tracked by

$$t_t = \lambda d \in D. \langle \langle \pi\pi d, \pi\pi' d \rangle, \langle \pi'\pi d, \pi'\pi' d \rangle \rangle$$

which on pairs of pairs have the following behavior:

$$\langle \langle r_f, s_f \rangle, \langle r_g, s_g \rangle \rangle \mapsto \langle \langle r_f, r_g \rangle, \langle s_f, s_g \rangle \rangle.$$

The unit pI of the tensor is given by the object $(\vec{R} \mapsto I, (\vec{R}, \vec{S}) \mapsto Eq_I)$.

The exponential of (f^p, f^r) and (g^p, g^r) in $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$, is $(f^p, f^r) \multimap (g^p, g^r)$ defined by

$$(f^p, f^r) \multimap (g^p, g^r) = (f^p \multimap g^p, f^r \multimap g^r)$$

where

$$(f^p \multimap g^p)(\vec{R}) = f^p(\vec{R}) \multimap g^p(\vec{R})$$

and $(f^r \multimap g^r)(\vec{R}, \vec{S})(\vec{A})$ is defined by

$$\begin{array}{c} \langle d_r, d_s \rangle \; (f^r \multimap g^r)(\vec{R}, \vec{S})(\vec{A}) \; \langle d'_r, d'_s \rangle \\ \Updownarrow \\ \langle r, s \rangle \; f^r(\vec{R}, \vec{S})(\vec{A}) \; \langle r', s' \rangle \Rightarrow \langle \Phi(d_r)(r), \Phi(d_s)(s) \rangle \; g^r(\vec{R}, \vec{S})(\vec{A}) \; \langle \Phi(d'_r)(r'), \Phi(d'_s)(s') \rangle \\ \wedge \\ \langle d_r, d_s \rangle \; (f^p \multimap g^p)(\vec{R}) \times (f^p \multimap g^p)(\vec{S}) \; \langle d'_r, d'_s \rangle \end{array}$$

This is an exhaustive description, in the sense that only pairs are ever related.

To verify the adjunction $(-) \otimes (f^p, f^r) \dashv (f^p, f^r) \multimap (-)$, we use that we know that it holds in the first component and then check that the bijection can be restricted to realizers that define morphisms in the second component; the latter is a direct consequence of the way the relational interpretations of $\otimes$ and $\multimap$ are defined. □

Lemma 6.21. $\mathbb{L} \dashv \mathbb{U}$ *is a fibred symmetric monoidal adjunction.*

Proof. This proceeds much as in the unfibred case. We show that $\mathbb{L}$ is a fibred strong symmetric monoidal functor. We must provide a morphism m_I and a natural transformation m, but we can simply use the same realizers as before, since everything has been defined coordinatewise and these realizers are independent of the specific pers, and hence are uniform realizers. □

Lemma 6.22. $\Omega = 1$ *is a split generic object of* $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PAP}(D)$.

Proof. Obvious. □

Lemma 6.23. *If* (f^p, f^r) *is an object of* $\mathbf{PFam}(\mathbf{AP}(D)_\perp)_{n+1}$*, then* $(\bigcap f^p, \bigcap f^r)$*, where*

$$(\textstyle\bigcap f^p)(R_1, \ldots, R_n)(x, y) \iff$$
$$\textstyle\bigcap_{R \in \mathrm{Obj}(\mathbf{AP}(D)_\perp)} f^p(R_1, \ldots, R_n, R)(x, y) \wedge \forall R, S, A \rightarrowtail R \times S.\, \langle x, y\rangle f^r(Eq_{R_1}, \ldots, Eq_{R_n}, A)\langle x, y\rangle$$

and

$$\langle x, y\rangle(\textstyle\bigcap f^r)(A_1 \rightarrowtail R_1 \times S_1, \ldots, A_n \rightarrowtail R_n \times S_n)\langle x', y'\rangle \iff$$
$$\forall R, S, A \rightarrowtail R \times S.\, \langle x, y\rangle f^p(A_1, \ldots, A_n, A)\langle x', y'\rangle \wedge (\textstyle\bigcap f^p)(\vec{R})(x, x') \wedge (\textstyle\bigcap f^p)(\vec{S})(y, y')$$

is an object of $\mathbf{PFam}(\mathbf{AP}(D)_\perp)_n$*.*

Lemma 6.24. $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$ *has simple* Ω*-products.*

Proof. The construction is as in [12, Section 8.4]. Given a projection $\pi\colon n + m \to n$, we must define a right adjoint to π^*. This is done by extending the construction of the previous lemma in an obvious way to a functor. □

Proposition 6.25. *The diagram (11) constitutes a PILL*$_Y$ *model.*

Proof. It only remains to verify that the structure models the fixed point combinator. Here we simply use the Y from Theorem 6.15, which works since relations are strict and chain complete. □

We now proceed to show that this new PILL$_Y$ model can be extended to an LAPL-structure. For this we need just two more fibrations, $q\colon \mathbf{Fam}(\mathbf{Set}) \to \mathbf{PAP}(D)$ and $r\colon \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set})$. The fibre of $\mathbf{Fam}(\mathbf{Set})$ over n has as

Objects maps $f : obj(\mathbf{AP}(D))^n \to \mathbf{Set}$.

Morphisms $t : f \to g$ is a family

$$(t_{\vec{R}} : f(\vec{R}) \to g(\vec{R}))_{\vec{R} \in obj(\mathbf{AP}(D))^n}$$

and reindexing is given by composition. The fibre of $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set}))$ over an object $f : obj(\mathbf{AP}(D))^n \to \mathbf{Set}$ is a preorder with

Objects maps $g : obj(\mathbf{AP}(D))^n \to \mathbf{Set}$, such that

$$\forall \vec{R} \in obj(\mathbf{AP}(D))^n.\ g(\vec{R}) \subseteq f(\vec{R}).$$

Morphisms There is a morphism $g \to g'$ if

$$\forall \vec{R} \in obj(\mathbf{AP}(D))^n.\ g(\vec{R}) \subseteq g'(\vec{R}).$$

Here reindexing is with respect to morphisms in $\mathbf{PAP}(D)$ is given by composition, whereas reindexing with respect to morphisms in $\mathbf{Fam}(\mathbf{Set})$ is given by inverse image.

Lemma 6.26. *q is a fibration with fibred products, and (r, q) is an indexed first-order logic fibration with simple Ω-products and -coproducts.*

Proof.

$$\begin{tikzcd}
\mathrm{Sub}(\mathbf{Set}) \arrow[d] \\
\mathbf{Set}
\end{tikzcd}$$

is a first-order logic fibration with generic object and all simple products and coproducts. By Lemma A.8 in [5] we can construct the pullback

$$\begin{tikzcd}
\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[r] \arrow[d] & \mathrm{Sub}(\mathbf{Set}) \arrow[d] \\
\mathbf{Set}^{\to} \arrow[r, "\mathrm{dom}"] & \mathbf{Set}
\end{tikzcd}$$

obtaining that $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \longrightarrow \mathbf{Set}^{\to} \xrightarrow{\mathrm{cod}} \mathbf{Set}$ is a composable fibration with the desired qualities. Yet this is not quite the right fibration. Fortunately we have

$$\begin{tikzcd}
\mathbf{Fam}(\mathbf{Set}) \arrow[rr, "\simeq"] \arrow[dr] & & \mathbf{Set}^{\to} \arrow[dl, "cod"] \\
& \mathbf{Set} &
\end{tikzcd}$$

by the isomorphism mapping $(U_x)_{x \in X}$ to $\coprod_{x \in X} U_x \to X$. And now

$$\begin{tikzcd}
\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[rr] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{Fam}(\mathbf{Set}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Fam}(\mathbf{Set}) \arrow[r, "\simeq"] \arrow[dr] & \mathbf{Set}^{\to} \arrow[d, "cod"] \\
\mathbf{PAP}(D) \arrow[rr] & & \mathbf{Set}
\end{tikzcd}$$

is a pullback. The bottom half is a pullback by definition, the map $\mathbf{PAP}(D) \to \mathbf{Set}$ operates as follows

$$n \ \mapsto \ obj(\mathbf{AP}(D))^n \qquad (\vec{f^p}, \vec{f^r}) \colon n \to m \ \mapsto \ \vec{f^p} \colon obj(\mathbf{AP}(D))^n \to obj(\mathbf{AP}(D))^m$$

And the top one easily is a pullback as well. As → preserves products, the leftmost composable fibration have the desired qualities. □

We can then define the functor $I\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{Fam}(\mathbf{Set})$ to be the fibred version of $Classes$.

Lemma 6.27. *I is a faithful and product-preserving map of fibrations.*

It is now time to define the contravariant map of fibrations

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{AP}(D)_\perp)^2 \arrow[rr, "U"] \arrow[dr] & & \mathbf{Fam}(\mathbf{Set}) \arrow[dl] \\
& \mathbf{PAP}(D) &
\end{tikzcd}$$

This is defined at index n on

Objects by $U(f,g) = \vec{R} \mapsto P(I(f^p(\vec{R})) \times I(g^p(\vec{R})))$, where $P(-)$ denotes powerset,

Morphisms by $U(\alpha : f \to f', \beta : g \to g') = \vec{R} \mapsto$

$$A \subseteq I(f'^p(\vec{R})) \times I(g'^p(\vec{R})) \mapsto \{\, (x,y) \in I(f^p(\vec{R})) \times I(g^p(\vec{R})) \mid (I(\alpha)(x), I(\beta)(y)) \in A \,\}$$

Lemma 6.28. *U is a contravariant map of fibrations.*

Proof. U can be equivalently defined as

$$(\sigma, \tau) \mapsto 2^{I(\sigma) \times I(\tau)},$$

which makes the statement clear. □

We can then define a family of bijections $(\chi_n)_{n \in \mathrm{Obj}(\mathbf{PAP}(D))}$ such that for all $f, g \in (\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ and $M \in (\mathbf{Fam}(\mathbf{Set}))_n$

$$\chi_n \colon \mathbf{Fam}(\mathbf{Set})(M, U_n(f,g))_n \to \mathrm{Obj}(\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set}))_{M \times I_n(\mathbb{U}_n(f) \times \mathbb{U}_n(g))})$$

by

$$\chi_n(h) = \{(m,(a,b)) | (a,b) \in h(m)\}$$

Lemma 6.29. *χ is a bijection, which is natural in the domain variable, is natural in f,g, and which commutes with reindexing functors.*

We have now proved:

Proposition 6.30. *The diagram*

$$\begin{tikzcd}
& & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[r, phantom, "\perp"] \arrow[r, bend right, "U"'] \arrow[dr] & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[l, bend right, "L"'] \arrow[r, hook] \arrow[d] & \mathbf{Fam}(\mathbf{Set}) \arrow[dl] \\
& \mathbf{PAP}(D) &
\end{tikzcd} \tag{12}$$

constitutes a pre-LAPL structure.

Now we define a subfunctor V of U on

Objects by $V(f, g) = \vec{R} \mapsto \{ I(A) \mid A \rightarrowtail_{\mathbf{AP}(D)_\perp} (f^p(\vec{R}) \times g^p(\vec{R})) \}$,

One can now show that V is closed under all the constructions performable on admissible relations and that it contains all graph relations.

Lemma 6.31. *The structure in diagram (12) and V model admissible relations.*

Proof. We refer to figure 4 and provide only a part of a formula to hint at which construction we are debating:

eq_σ: Equality on a type σ is modeled as the diagonal subobject of $[\![\sigma]\!] \times [\![\sigma]\!]$. This corresponds to an admissible relation because it is isomorphic to $[\![\sigma]\!]$ by the continuous functions $\lambda d \in D.\, \langle d, d\rangle$ and π.

$\rho(t\ x, u\ y)$: Reindexing an admissible relation by a strict continuous function (i.e. (t, u)) is bound to give an admissible relation. We consider chain-completeness: Given two index-wise related chains in $(t, u)^{-1}(\rho)$, (t, u) taken on these gives us two index-wise related chains in ρ. Since ρ is chain-complete their limits are related in ρ, and since (t, u) is continuous the limits of the original chains is in the inverse image of the limit in ρ.

$\rho(x, y) \wedge \rho'(x, y)$: Conjunction is modeled by intersection, under which admissible relations by lemma 6.11 are stable.

$(x\colon \tau, y\colon \sigma).\, \rho(y, x)$: swapping the abscissa and ordinal axis does not break admissibility.

$!\rho$: This is simply the usual lift of relations.

$\top$: This is all classes. This is admissible.

$\phi \supset \rho(x, y)$: If ϕ does not hold we get all classes. If ϕ does hold we get ρ which is admissible.

Quantifications: All quantifications are modeled through intersections and are thus taken care of by lemma 6.11.

□

Having come so far, we move on to describe $\mathbf{LinAdmRelations}$ and $\mathbf{AdmRelCtx}$ from Section 4. Recall that $\mathbf{AdmRelCtx}$ is defined as the pullback

$$\begin{tikzcd}
\mathbf{AdmRelCtx} \arrow[r] \arrow[d, "\langle \partial_0, \partial_1 \rangle"'] & \mathbf{Fam}(\mathbf{Set}) \arrow[d] \\
\mathbf{PAP}(D) \times \mathbf{PAP}(D) \arrow[r, "\times"] & \mathbf{PAP}(D)
\end{tikzcd}$$

which means that $\mathbf{AdmRelCtx}$ has as

Objects triples (n, m, Φ) where $\Phi\colon obj(\mathbf{AP}(D))^{n+m} \to \mathbf{Set}$, assigns a set to a vector of admissible pers.

Morphisms triples $(f, g, \rho)\colon (n, m, \Phi) \to (n', m', \Phi')$ where $f\colon n \to n'$ and $g\colon m \to m'$ are morphisms in $\mathbf{PAP}(D)$ and ρ is an indexed family of maps

$$\rho = (\rho_{\vec{R},\vec{S}}\colon \Phi(\vec{R}, \vec{S}) \to \Phi'(\vec{f^p}(\vec{R}), \vec{g^p}(\vec{S})))_{\vec{R} \in obj(\mathbf{AP}(D))^n, \vec{S} \in obj(\mathbf{AP}(D))^m}$$

where Φ and Φ' are evaluated on the combined lists of admissible pers.

In this concrete case $\mathbf{LinAdmRelations}$ can be described as follows: Given an object (n, m, Φ) over (n, m), the fibre of $\mathbf{LinAdmRelations}$ over (n, m, Φ) has as

Objects triples (ϕ, f, g) such that f and g are objects of $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$ over n and m respectively and ϕ is an indexed family of maps

$$\phi = (\phi_{\vec{R},\vec{S}} \colon \Phi(\vec{R}, \vec{S}) \to \{ A \mid A \rightarrowtail f^p(\vec{R}) \times g^p(\vec{S}) \})_{\vec{R} \in obj(\mathbf{AP}(D))^n, \vec{S} \in obj(\mathbf{AP}(D))^m}$$

Morphisms A morphism $(\phi, f, g) \to (\psi, f', g')$ is a pair of morphisms

$$(t \colon f \to f', u \colon g \to g')$$

in $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ and $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_m$, respectively, such that

$$\forall \vec{R} \in obj(\mathbf{AP}(D))^n, \vec{S} \in obj(\mathbf{AP}(D))^m . \forall z \in \Phi(\vec{R}, \vec{S}).$$
$$\langle x, y \rangle \; \phi(z) \; \langle x, y \rangle \Rightarrow \langle t(x), u(y) \rangle \; \psi(z) \; \langle t(x), u(y) \rangle$$

Note that we now have two obvious projections ∂_0 and ∂_1.

Finally we can define the required functor J.

$$\begin{tikzcd} \mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[d] & & \mathbf{LinAdmRelations} \arrow[d] \\ \mathbf{PAP}(D) & {} \arrow[r] & \mathbf{AdmRelCtx} \end{tikzcd}$$

For the base categories, J is defined on

Objects by $n \mapsto (n, n, (\prod_i \{ A \mid A \rightarrowtail R_i \times S_i \})_{\vec{R},\vec{S} \in \mathbf{AP}(D)^n})$

Morphisms by $f \mapsto (f, f, \prod_i f_i^r)$

and for the total categories, J is defined on

Objects by $(f^p, f^r) \mapsto (f^r, f, f)$

Morphisms by $\alpha \mapsto (\alpha, \alpha)$.

This definition on morphisms is legal because α preserves relations.

In order to show that J preserves tensor products, we need the following lemma

Lemma 6.32. *The tensor product in* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *can be described as*

$$(\rho, f, g) \otimes (\rho', f', g') = (\rho \otimes \rho', f \otimes f', g \otimes g')$$

where $\rho \otimes \rho'$ *is calculated pointwise, i.e for* $z \in \phi(\vec{R}, \vec{S})$

$$(\rho \otimes \rho')_{\vec{R},\vec{S}}(z) = \rho(z) \otimes \rho'(z)$$

Proof. We argue that this construction defines a left adjoint to $\multimap$. The standard curry-uncurry-adjunction holds, on the level of realizers even, which is not hard to show. $\square$

Lemma 6.33. *J is a map of linear λ_2-fibrations.*

Proof. We must show that J preserves $\multimap$, $\otimes$, $\prod$, I and !.

The constructions in the two categories are virtually identical except for $\otimes$ until application of lemma 6.32.

To check the case of ! we consider the logical expression for $!\rho\colon \mathsf{AdmRel}(\sigma, \tau)$:

$$(x : !\sigma, y : !\tau). x \downarrow \iff y \downarrow \quad \wedge \quad x \downarrow \supset \rho(\epsilon x, \epsilon y)$$

The expression $x \downarrow$ equates $x \neq [\perp]$. Hence $x \downarrow \iff y \downarrow$ express the fact that no lifted class is related to $[\perp]$ in $!\rho$.

Further since ϵ provides us with unlifted versions of its argument, $x \downarrow \supset \rho(\epsilon x, \epsilon y)$ states that liftet classes are related in $!\rho$ only if their unlifted versions are related in ρ.

This is an exact description of the lifting performed by the functor L. □

It is easy to see that $\partial_0 J = id$ and $\partial_1 J = id$.

Theorem 6.34. *The diagram in* (12) *constitutes a parametric LAPL-structure.*

Proof. By the preceding results it is clear that it is an LAPL-structure; it only remains to show that it is a parametric such. Extensionality holds since the logic is essentially given by regular subobjects, which means that we have very strong equality [12], and thus also extensionality. The parametricity schema is easily verified to hold. □

Example 6.35. To ease notation in this example we shall write $(x, y) \in A$ for $\langle x, y\rangle A\langle x, y\rangle$ for regular subobjects $A \rightarrowtail R \times S$, as we do in LAPL. We will also leave Ψ, Φ implicit, and simply write $f\ x$ for $\Phi(f)(x)$.

We consider the type $\mathrm{Nat} = [\![\prod \alpha. (\alpha \multimap \alpha) \to \alpha \multimap \alpha]\!]$. By definition

$$d(\mathrm{Nat}^p)d'$$

iff for all R, S pers and all regular subobjects $A \rightarrowtail R \times S$, $(f, g) \in (A \multimap A)$ and $(x, y) \in A$

$$(d\ f\ x, d'\ g\ y) \in A.$$

The domain of Nat contains the elements $\perp = \lambda f \lambda x. \perp$ and $\underline{n} = \lambda f. \lambda x. f^n(x)$, in particular $\underline{0} = \lambda f \lambda x. x$.

Lemma 6.36. *Suppose $\underline{n} \leq \underline{m}$. Then $n = m$.*

Proof. Consider the two functions $f, g\colon D \to D$ given by $f(d) = \langle d, \iota\rangle$, where ι is the code of the identity function, and g being the first projection. Both are continuous and since $g \circ f = id\ f$ is injective. Define the sequence of elements $x_n = f^n(\perp)$. This sequence is strictly increasing.

Now, if $\underline{n} \leq \underline{m}$ then

$$x_n = \underline{n}\ f\ \perp \leq \underline{m}\ f\ \perp = x_m$$

so $n \leq m$. Further,

$$x_{m-n} = \underline{n}\ g\ x_m \leq \underline{m}\ g\ x_m = \perp$$

so $m = n$. □

Lemma 6.37. *The per*

$$\{\{\bot\}\} \cup \{\,\{\underline{n}\} \mid n\}$$

is a admissible.

Proof. Direct consequence of the lemma above. □

Proposition 6.38. *Suppose* $d(\mathrm{Nat}^p)d$*. Then either* $d = \bot$ *or* $d = \underline{n}$*.*

Proof. Consider the discrete admissible per D:

$$\{\{d\} \mid d \in D\}$$

Then given f, x consider the regular subobject $A \rightarrowtail \mathrm{Nat} \times D$ given by

$$(\bot, \bot) \in A, \qquad \forall n. \, (\underline{n}, f^n(x)) \in A.$$

A is admissible, simply because it contains no interesting increasing chains. Clearly $(\mathit{succ}, f) \in A \multimap A$, so

$$(d\, \mathit{succ}\, 0, d\ f\ x) \in A,$$

i.e., if $d\, \mathit{succ}\ 0 = \bot$, then $d\ f\ x = \bot$ for all f, x and so $d = \bot$, and if $d\, \mathit{succ}\ 0 = \underline{n}$ for some n, then $d\ f\ x = f^n(x)$, for all f, x, so $d = \underline{n}$. As we have seen, there are no other possibilities for $d\, \mathit{succ}\ 0$. □

Proposition 6.39. *Suppose* $d(\mathrm{Nat}^p)d'$*, then* $d = d'$*.*

Proof. Analyzing the above proof we see that

$$d = d\, \mathit{succ}\ 0$$

By considering the regular subobject $A \rightarrowtail \mathrm{Nat} \times \mathrm{Nat}$ given by

$$(\bot, \bot) \in A, \qquad \forall n. \, (\underline{n}, \underline{n}) \in A$$

we conclude

$$d\, \mathit{succ}\ 0 = d'\ \mathit{succ}\ 0.$$

□

Acknowledgments

We gratefully acknowledge discussions with Milly Maietti, Gordon Plotkin, John Reynolds, Pino Rosolini and Alex Simpson.

References

[1] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1998. 6.1

[2] A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, Edinburgh University, 1997. 2.1, 3.8, 4, 4.2

[3] P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical report, University of Cambridge, 1995. 1.1

[4] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal*, volume 41 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 2000. 1

[5] L. Birkedal and R. Møgelberg. Categorical models for Abadi-Plotkin's Logic for parametricity. *Mathematical Structures in Computer Science*, 2005. To Appear (Accepted for publication). 1, 3.2, 4, 1, 4.1, 6.5, 6.5

[6] M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996. 1

[7] P.J. Freyd. Algebraically complete categories. In A. Carboni, M. C. Pedicchio, and G. Rosolini, editors, *Category Theory. Proceedings, Como 1990*, volume 1488 of *Lecture Notes in Mathematics*, pages 95–104. Springer-Verlag, 1990. 3.12

[8] P.J. Freyd. Recursive types reduced to inductive types. In *Proceedings of the fifth IEEE Conference on Logic in Computer Science*, pages 498–507, 1990. 3.12

[9] P.J. Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P.T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1991. 3.12

[10] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'Etat, Université Paris VII, 1972. 1

[11] H. Huwig and A. Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73:101–112, 1990. 1

[12] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 2.1, 4, 6.4, 6.5, 6.5

[13] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Math. Struct. in Comp. Science*, 11, 1996. 6, 6.2

[14] M. Maietti, P. Maneggia, and E. Ritter. Relating categorical semantics for intuitionistic linear logic. *Applied Categorical Structures*, 2004. To Appear. 1.1

[15] Maria E Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. Technical Report CSR-01-7, University of Birmingham, School of Computer Science, August 2001. 4

[16] R. E. Møgelberg. Parametric completion for models of polymorphic intuitionistic / linear lambda calculus. Technical Report TR-2005-60, IT University of Copenhagen, February 2005. 1, 3.3, 6.5

[17] R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005. 1, 4, 4, 6.2, 6.4

[18] R. E. Møgelberg, L. Birkedal, and G. Rosolini. Synthetic domain theory and models of linear Abadi & Plotkin logic. Technical Report TR-2005-59, IT University of Copenhagen, February 2005. 1

[19] R.L. Petersen and J. Thamsborg. Polymorphism and linearity all in one pill. Student Project, 2003. 4

[20] B.C. Pierce. *Types and Programming Languages*. MIT Press, 2002. 1

[21] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000. 1

[22] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1, 3.12, 6

[23] Gordon Plotkin and Martín Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, volume 664 of *Lecture Notes in Comput. Sci.*, pages 361–375. Springer, Berlin, 1993. 1, 2, 2.2.3, 2.2.6, 3.8

[24] J.C. Reynolds. Towards a theory of type structure. In *Colloquium sur La Programmation*, volume 19 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1974. 1

[25] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983. 1

[26] J.C. Reynolds. Private communication, June 2000. 1

[27] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. 1

[28] Izumi Takeuti. An axiomatic system of parametricity. *Fund. Inform.*, 33(4):397–432, 1998. Typed lambda-calculi and applications (Nancy, 1997). 2.2

[29] P. Wadler. The Girard-Reynolds isomorphism (second edition). Manuscript, March 2004. 2.2

Categorical Models of PILL

Rasmus Ejlers Møgelberg
Lars Birkedal
Rasmus Lerchedahl Petersen

Abstract

We review the theory of adjunctions and comonads in the 2-category of symmetric monoidal adjunctions. This leads to the definitions of linear adjunctions, linear categories and models of DILL as in [1, 6, 7]. This theory is generalized to the fibred case, and we define models of PILL and PILL$_Y$ and morphisms between them.

Contents

1 Models of DILL

1.1 The 2-category of symmetric monoidal categories

In the following SMC stands for symmetric monoidal category.

Definition 1.1. *A functor of SMC's from $\mathbb{C}$ to $\mathbb{C}'$ is a functor F plus natural transformation*

$$m\colon F(-)\otimes F(=)\Rightarrow F(-\otimes =)$$

and map $m_I\colon I\to F(I)$ satisfying the following commutative diagrams

$$\begin{tikzcd}
(F(-)\otimes F(=))\otimes F(\equiv) \arrow[r, "\cong"] \arrow[d, "m\otimes id"'] & F(-)\otimes(F(=)\otimes F(\equiv)) \arrow[d, "id\otimes m"] \\
F((-)\otimes(=))\otimes F(\equiv) \arrow[d, "m"'] & F(-)\otimes F((=)\otimes(\equiv)) \arrow[d, "m"] \\
F(((-)\otimes(=))\otimes(\equiv)) \arrow[r, "\cong"] & F((-)\otimes((=)\otimes(\equiv)))
\end{tikzcd}$$

$$\begin{tikzcd}
I\otimes F(-) \arrow[r, "\cong"] \arrow[d, "m_I\otimes id"'] & F(-) \arrow[d, "\cong"] \\
F(I)\otimes F(-) \arrow[r, "m"] & F(I\otimes(-))
\end{tikzcd}
\qquad
\begin{tikzcd}
F(-)\otimes F(=) \arrow[r, "\cong"] \arrow[d, "m"'] & F(=)\otimes F(-) \arrow[d, "m"] \\
F((-)\otimes(=)) \arrow[r, "\cong"] & F((=)\otimes(-))
\end{tikzcd}$$

The functor F is called ***strong*** *if the transformations m, m_I are isomorphisms and* ***strict*** *if they are identities. The composite of symmetric monoidal functors*

$$(F, m^F, m^F_I)\colon \mathbb{C}\to\mathbb{C}' \text{ and } (G, m^G, m^G_I)\colon \mathbb{C}'\to\mathbb{C}''$$

is $(GF, G(m^F)\circ m^G, G(m^F_I)\circ m^G_I)$, where

$$GF(-)\otimes GF(=) \xRightarrow{m^G} G(F(-)\otimes F(=)) \xRightarrow{G(m^F_I)} GF(-\otimes =)$$
$$I \xrightarrow{m^G_I} G(I) \xrightarrow{G(m^F_I)} GF(I).$$

Definition 1.2. *A symmetric monoidal transformation between symmetric monoidal functors (F, m^F, m^F_I) and (G, m^G, m^G_I) is a natural transformation $\phi\colon F\Rightarrow G$ in the usual sense satisfying*

$$\begin{tikzcd}
F(-)\otimes F(=) \arrow[r, "m^F"] \arrow[d, "\phi\otimes\phi"'] & F((-)\otimes(=)) \arrow[d, "\phi"] \\
G(-)\otimes G(=) \arrow[r, "m^G"] & G((-)\otimes(=))
\end{tikzcd}
\qquad
\begin{tikzcd}
& I \arrow[dl, "m^F_I"'] \arrow[dr, "m^G_I"] & \\
F(I) \arrow[rr, "\phi_I"] & & G(I).
\end{tikzcd}$$

The above defines the 2-category of SMC's. In this 2-category one can define adjunctions, monads, comonads etc. as usual. In the following we write out some of these definitions in detail.

Definition 1.3. *A pair of functors of SMC's*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, bend right, "G"'] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right, "F"']
\end{tikzcd}$$

constitute a symmetric monoidal adjunction (with F left adjoint), if $F \dashv G$ as usual and both the unit $\eta\colon id_{\mathbb{D}} \Rightarrow GF$ and the counit $\epsilon\colon FG \Rightarrow id_{\mathbb{C}}$ are symmetric monoidal transformations. This means that the following diagrams commute:

$$\begin{tikzcd}
FG(-) \otimes FG(=) \arrow[r, "m^F"] \arrow[d, "\epsilon\otimes\epsilon"'] & F(G(-) \otimes G(=)) \arrow[d, "F(m^G)"] \\
(-) \otimes (=) & FG((-) \otimes (=)) \arrow[l, "\epsilon"]
\end{tikzcd}
\qquad
\begin{tikzcd}
I \arrow[d, "m_I^F"'] & \\
F(I) \arrow[r, "F(m_I^G)"'] & FG(I) \arrow[ul, "\epsilon"']
\end{tikzcd}$$

$$\begin{tikzcd}
(-) \otimes (=) \arrow[r, "\eta"] \arrow[d, "\eta\otimes\eta"'] & GF((-) \otimes (=)) \\
GF(-) \otimes GF(=) \arrow[r, "m^G"] & G(F(-) \otimes F(=)) \arrow[u, "G(m^F)"']
\end{tikzcd}
\qquad
\begin{tikzcd}
I \arrow[r, "m_I^G"] \arrow[d, "\eta"'] & G(I) \arrow[dl, "G(m_I^F)"] \\
GF(I) &
\end{tikzcd}$$

The following theorem is originally from [5].

Theorem 1.4. *An adjunction $F \dashv G$ between symmetric monoidal categories is a symmetric monoidal adjunction iff F is a strong symmetric monoidal functor.*

For a proof we refer to [8, 2]. We just note that if (F, m^F, m_I^F) is strong symmetric, then the natural transformations $m^G\colon G(-) \otimes G(=) \to G((-) \otimes (=))$ is given as the adjoint correspondent to the composition

$$F(G(-) \otimes G(=)) \xrightarrow{(m^F)^{-1}} FG(-) \otimes FG(=) \xrightarrow{\epsilon\otimes\epsilon} (-) \otimes (=)$$

and the natural transformation m_I^G is given as the adjoint correspondent to

$$(m_I^F)^{-1}\colon FI \to I.$$

A symmetric monoidal comonad on an SMC $\mathbb{C}$ is a vector $((F, m, m_I), \epsilon, \delta)$ such that (F, m, m_I) is a SMC epifunctor, (F, ϵ, δ) is a comonad and ϵ, δ are symmetric monoidal transformations. Since the usual construction of a comonad from an adjunction can be carried out inside any 2-category, we obtain:

Lemma 1.5. *Any symmetric monoidal adjunction*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, bend right, "G"'] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right, "F"']
\end{tikzcd}$$

gives rise to a symmetric monoidal comonad on $\mathbb{C}$.

Suppose we are given a functor $F\colon \mathbb{C} \to \mathbb{D}$ between symmetric monoidal *closed* categories. Then there exists a natural transformation $n\colon F((-) \multimap (=)) \Rightarrow F(-) \multimap F(=)$ defined as

$$F((-) \multimap (=)) \longrightarrow F(-) \multimap F((-) \multimap (=)) \otimes F(-) \xrightarrow{id \multimap m}$$
$$F(-) \multimap F(((-) \multimap (=)) \otimes (-)) \xrightarrow{id \multimap F(ev)} F(-) \multimap F(=),$$

where the first map is the unit of the adjunction.

Definition 1.6. *A morphism of SMCC's is simply a morphism of SMC's. A strong map of SMCC's is a strong map of SMC's where the transformation n above is an isomorphism. The map is strict if it is a strict map of SMC's and the transformation n is the identity.*

1.2 The co-Kleisli category and the Eilenberg-Moore category of a comonad

Suppose we are given an SMC $\mathbb{C}$ and a symmetric monoidal comonad (T, ϵ, δ) on it. We can then form the co-Kleisli category of the comonad as usual:

Objects: are the objects of $\mathbb{C}$.

Morphisms: A morphism from X to Y is a morphism in $\mathbb{C}$ from TX to Y.

Composition: Composition of maps $f\colon X \to Y$ and $g\colon Y \to Z$ is given as

$$\begin{tikzcd} TX \arrow[r, "\delta_X"] & T^2X \arrow[r, "Tf"] & TY \arrow[r, "g"] & Z. \end{tikzcd}$$

The natural transformation ϵ plays the role of the identity.

We denote the co-Kleisli category by $\mathbb{C}_T$.

We can also form the Eilenberg-Moore category of the comonad as

Objects: Coalgebras for the comonad, i.e., maps $h\colon X \to TX$ satisfying

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "h"'] & TX \arrow[d, "\delta"] \\ TX \arrow[r, "Th"] & T^2X \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "h"] \arrow[dr, "id"'] & TX \arrow[d, "\epsilon"] \\ & X \end{tikzcd}$$

Morphisms: Morphisms of coalgebras

We denote the Eilenberg-Moore category by $\mathbb{C}^T$.

Lemma 1.7. *The co-Kleisli category of a comonad is isomorphic to the full subcategory of the Eilenberg-Moore category on the free coalgebras for the comonad, i.e., the coalgebras of the form $\delta_X\colon T(X) \to T^2(X)$.*

Proof. There is clearly a bijective correspondence between objects. We need to check that this correspondence extends to morphisms. Suppose $h\colon TX \to Y$ is a morphism in the co-Kleisli category from X to Y. Then $Th \circ \delta_X$ defines a morphism of coalgebras since

$$\begin{tikzcd} T^2X \arrow[r, "T\delta"] & T^3X \arrow[r, "T^2h"] & T^2Y \\ TX \arrow[u, "\delta_X"] \arrow[r, "\delta_X"] & T^2X \arrow[u, "\delta_{TX}"'] \arrow[r, "Th"] & TY, \arrow[u, "\delta_Y"'] \end{tikzcd}$$

where the square to the left commutes by the definition of comonad, and the diagram to the right commutes by naturality of δ. To check that this defines a functor from the co-Kleisli category to the Eilenberg-Moore category, suppose $h\colon TX \to Y$ and $h'\colon TY \to Z$ in $\mathbb{C}$. If we first use the functor and then compose, we obtain $(Th')\circ\delta_Y\circ(Th)\circ\delta_X$. If we first compose and then apply the functor, we obtain $T(h'\circ Th\circ\delta_X)\circ\delta_X$, which by definition of comonad is $Th'\circ T^2h\circ\delta_{TX}\circ\delta_X$. By naturality of δ, we conclude that the functor commutes with composition. Clearly ϵ is mapped to the identity.

Suppose on the other hand that $f\colon TX \to TY$ defines a map of coalgebras, i.e., $Tf\circ\delta_X = \delta_Y\circ f$. Then we can define the map $\epsilon_Y\circ f\colon TX \to Y$, which is a map in the co-Kleisli category from X to Y. Again we need to check that this defines a functor. Suppose $f'\colon TY \to TZ$ is another map of coalgebras. Composing first and the applying the functor gives $\epsilon_Z\circ f'\circ f$. Applying the functor first and then composing gives $\epsilon_Z\circ f'\circ T(\epsilon_Y\circ f)\circ\delta_X = \epsilon_Z\circ f'\circ T(\epsilon_Y)\circ\delta_Y\circ f$, since f is a map of coalgebras. We now use $T(\epsilon_Y)\circ\delta_Y = id_Y$ by one of the equations for comonads to conclude that the functor commutes with composition. Clearly the identity is mapped to ϵ.

We need to check that the two functors are inverses of each other. Suppose we start with a map in the co-Kleisli category, i.e., a map $h\colon TX \to Y$. Applying the two functors to this gives $\epsilon_Y\circ Th\circ\delta_X = h\circ\epsilon_{TX}\circ\delta_X = h$. If we start with a map of coalgebras $f\colon TX \to TY$, applying the two functors gives $T(\epsilon_Y)\circ T(f)\circ\delta_X = T(\epsilon_Y)\circ\delta_Y\circ f = f$. □

We have the usual adjunctions between $\mathbb{C}$ and $\mathbb{C}^T$ and $\mathbb{C}$ and $\mathbb{C}_T$. We can illustrate these as

$$\begin{tikzcd}
\mathbb{C} \arrow[r, bend right, "F^T"'] \arrow[r, phantom, "\perp"] \arrow[d, bend left, "F_T"] \arrow[d, phantom, "\dashv"] & \mathbb{C}^T \arrow[l, bend right, "U^T"'] \\
\mathbb{C}_T, \arrow[u, bend left, "U_T"] \arrow[ur, "i"'] &
\end{tikzcd}$$

where i is the inclusion. We know that $iF_T = F^T, U^Ti = U_T$. Without further assumptions, neither $\mathbb{C}_T$ nor $\mathbb{C}^T$ have a natural SMC structure, so it does not make sense to ask for the adjunctions to be symmetric monoidal.

Definition 1.8. *A* linear adjunction *is a symmetric monoidal adjunction*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, bend right, "G"'] \arrow[r, phantom, "\perp"] & \mathbb{D}, \arrow[l, bend right, "F"']
\end{tikzcd}$$

where the SMC-structure on $\mathbb{D}$ is in fact a cartesian structure, and $\mathbb{C}$ is SMCC.

A morphism of linear adjunctions from $\begin{tikzcd} \mathbb{C} \arrow[r, bend right, "G"'] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right, "F"'] \end{tikzcd}$ *to* $\begin{tikzcd} \mathbb{C}' \arrow[r, bend right, "G'"'] \arrow[r, phantom, "\perp"] & \mathbb{D}' \arrow[l, bend right, "F'"'] \end{tikzcd}$ *is a pair of functors H, K where H is a strict map of symmetric monoidal closed categories, and K is a strong map of symmetric monoidal categories such that the diagrams*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, "G"] \arrow[d, "H"] & \mathbb{D} \arrow[r, "F"] \arrow[d, "K"] & \mathbb{C} \arrow[d, "H"] \\
\mathbb{C}' \arrow[r, "F'"] & \mathbb{D}' \arrow[r, "G'"] & \mathbb{C}'
\end{tikzcd}$$

commute up to isomorphism. *Furthermore, H is required to commute with the comonads induced by the adjunctions, i.e., $HFG = F'G'H$, $H\epsilon = \epsilon' H$ and $H\delta = \delta' H$, where δ, δ' are the comultiplications of the comonad induced by the adjunctions.*

A natural transformation from (H, K) to (H, K') (notice that the first components of the two functors are equal) is a natural transformation from K to K'.

The definition of natural transformation may seem a bit unintuitive, in particular the fact that natural transformations are always identity on the SMCC components of a functor. We have chosen this definition because we want a fairly restrictive notion of equivalence between linear adjunctions.

It is well-known that DILL can be interpreted soundly and completely in any linear adjunction [1]

Remark 1.9. *An LNL-model is a linear adjunction in which the cartesian category is closed.*

Definition 1.10. *A linear category is an SMCC $\mathbb{C}$ with a symmetric monoidal comonad $((!, m, m_I), \epsilon, \delta)$ and symmetric monoidal natural transformations $e\colon\ !(-) \to I$, $d\colon\ !(-) \to !(-)\otimes!(-)$, such that*

- *For each object A, $(!A, e_A, d_A)$ is a commutative comonoid, i.e.,*

$$\begin{tikzcd}
!A \arrow[r, "d_A"] \arrow[dr, "\cong"'] & !A\otimes!A \arrow[d, "id\otimes e_A"] & !A \arrow[r, "d_A"] \arrow[dr, "d_A"'] & !A\otimes!A \arrow[d, "s"] \\
& !A \otimes I & & !A\otimes!A
\end{tikzcd}$$

$$\begin{tikzcd}
!A \arrow[r, "d_A"] \arrow[d, "d_A"'] & !A\otimes!A \arrow[r, "id\otimes d_A"] & !A \otimes (!A\otimes!A) \arrow[d, "\cong"] \\
!A\otimes!A \arrow[rr, "d_A\otimes id"] & & (!A\otimes!A)\otimes!A,
\end{tikzcd}$$

  where s is the natural transformation $(-) \otimes (=) \Rightarrow (=) \otimes (-)$,

- *e_A, d_A define coalgebra maps from $\delta_A\colon\ !A \to !!A$ to the coalgebras $m_I\colon I \to !I$ and*

$$!A\otimes!A \xrightarrow{\delta_A\otimes\delta_A} !!A\otimes!!A \xrightarrow{m} !(!A\otimes!A)$$

- *All coalgebra maps between free coalgebras preserve the comonoid structure, i.e., if $f\colon\ !A \to !B$ is such that*

$$\begin{tikzcd}
!!A \arrow[r, "!f"] & !!B \\
!A \arrow[u, "\delta_A"] \arrow[r, "f"] & !B \arrow[u, "\delta_B"']
\end{tikzcd}$$

  then

$$\begin{tikzcd}
!A \arrow[rr, "f"] \arrow[dr, "e_A"'] & & !B \arrow[dl, "e_B"] & !A \arrow[r, "f"] \arrow[d, "d_A"'] & !B \arrow[d, "d_B"] \\
& I & & !A\otimes!A \arrow[r, "f\otimes f"] & !B\otimes!B.
\end{tikzcd}$$

Linear categories model Intuitionistic Linear Logic (ILL). In ILL, types of the form $!A$ behave intuitionistically, and intuitively, one should think of e as providing weakening for these types, and d as providing contraction.

Lemma 1.11. *In Definition 1.10, the last condition can be replaced by the condition that δ preserves comonoid structure.*

Proof. From the definition of comonads, we see that δ is a coalgebra map, and thus the new condition is a special case of the old.

For the other implication, suppose that $f\colon\ !A \to !B$ is a map of coalgebras. Then

$$e_B \circ f = e_{!B} \circ \delta_B \circ f = e_{!B} \circ (!f) \circ \delta_A = e_{!A} \circ \delta_A = e_A$$

which proves commutativity of the first diagram. For the second notice first that

$$f =!\epsilon_B \circ \delta_B \circ f =!\epsilon_B \circ (!f) \circ \delta_A.$$

The result now follows from the following commutative diagram:

$$\begin{tikzcd}
!A \arrow[r, "\delta"] \arrow[d, "d_A"'] & !!A \arrow[r, "!f"] \arrow[d, "d_{!A}"] & !!B \arrow[r, "!\epsilon"] \arrow[d, "d_{!B}"] & !B \arrow[d, "d_B"] \\
!A \otimes !A \arrow[r, "\delta\otimes\delta"] & !!A \otimes !!A \arrow[r, "!f\otimes !f"] & !!B \otimes !!B \arrow[r, "!\epsilon\otimes !\epsilon"] & !B \otimes !B.
\end{tikzcd}$$

□

Definition 1.12. *A morphism of linear categories from $(\mathbb{C}, !, d, e)$ to $(\mathbb{C}', !', d', e')$ is a strong symmetric monoidal closed functor F preserving all the comonad structure on the nose, i.e., $!'F = F!, \epsilon'F = F\epsilon, \delta'F = F\delta$. If the functor F is strict, we call this a strict functor of linear categories.*

Lemma 1.13. *For a linear category, the associated Eilenberg-Moore category is cartesian.*

Proof. The product of two coalgebras $h_A\colon A \to !A, h_B\colon B \to !B$ is

$$A \otimes B \xrightarrow{h_A \otimes h_B} !A \otimes !B \xrightarrow{m} !(A \otimes B)$$

with projection given by

$$A \otimes B \xrightarrow{id \otimes h_B} A \otimes !B \xrightarrow{id \otimes e_B} A \otimes I \xrightarrow{\cong} A$$

and diagonal Δ_A given by

$$A \xrightarrow{h_A} !A \xrightarrow{d_A} !A \otimes !A \xrightarrow{\epsilon_A \otimes \epsilon_A} A \otimes A.$$

Having defined the diagonal, pairing of functions $f\colon A \to B, g\colon A \to C$ is defined as usual as $\langle f, g\rangle = f \otimes g \circ \Delta_A$.

The terminal object is $m_I\colon I \to !I$. □

Proposition 1.14. *Each linear adjunction*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, bend right, "G"'] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right, "F"']
\end{tikzcd}$$

gives rise to a linear category whose comonad is $! = FG$. This extends to a functor from the category of linear adjunctions to the category of linear categories with strict morphisms.

Proof. Recall first that in a linear adjunction, the left adjoint is strong by Theorem 1.4, i.e., m, m_I are isomorphisms.

The map e_A is the composition

$$FGA \xrightarrow{F(\star)} F(1) \xrightarrow{m_I^{-1}} I$$

and d_A is

$$FGA \xrightarrow{F(\Delta)} F(GA \times GA) \xrightarrow{m^{-1}} FGA \otimes FGA.$$

For the details of this proof, we refer to [2].

The last part of the proposition is obvious. □

1.3 The category of products of free coalgebras

Given a linear category $(\mathbb{C}, !, e, d)$ we define $\mathbb{C}_!^\star$ to have as objects finite vectors of objects of $\mathbb{C}$ and as morphisms from (A_i) to (B_j) morphisms of $\mathbb{C}^!$ from $\prod \delta_{A_i}$ to $\prod \delta_{B_j}$. This category is equivalent to the full subcategory of $\mathbb{C}^!$ on products of objects of $\mathbb{C}_!$. We call $\mathbb{C}_!^\star$ the category of products of free coalgebras and we will often denote an object of $\mathbb{C}_!^\star$ simply as $\prod \delta_{A_i}$ instead of (A_i).

Lemma 1.15. *Given a linear category* $(\mathbb{C}, !, e, d)$*, there is a symmetric monoidal adjunction*

$$\begin{tikzcd} \mathbb{C} \arrow[r, bend right, "F_!^\star"'] \arrow[r, phantom, "\perp"] & \mathbb{C}_!^\star \arrow[l, bend right, "U_!^\star"'] \end{tikzcd},$$

i.e., a linear adjunction whose associated linear category (Proposition 1.14) is isomorphic to $(\mathbb{C}, !, e, d)$.

Proof. This is basically the restriction of the adjunction between $\mathbb{C}^!$ and $\mathbb{C}$. To show that the adjunction is symmetric monoidal, it suffices to show that $U_!^\star$ is a strong symmetric monoidal functor. But

$$U_!^\star((A_i) \times (B_j)) = U_!^\star((A_1, \dots A_n, B_1 \dots B_m) = !A_1 \otimes \dots !A_n \otimes !B_1 \otimes \dots B_m$$

and

$$U_!^\star(A_i) \otimes U_!^\star(B_j) = (\otimes_i !A_i) \otimes (\otimes_j !B_j)$$

so $U_!^\star$ is clearly a strong symmetric monoidal functor. □

Lemma 1.16. *The construction of Lemma 1.15 extends to a functor from the category of linear categories with strict maps to the category of linear adjunctions. This functor is right inverse to the functor of Proposition 1.14.*

Proof. Suppose $K\colon (\mathbb{C}, !) \to (\mathbb{D}!')$ is a map of linear categories. We define $H\colon \mathbb{C}_!^\star \to \mathbb{D}_{!'}^\star$ by $H(A_i) = (KA_i)$ and on morphisms

$$H(h\colon \otimes !A_i \to \otimes !B_j) = K(h)\colon \otimes !KA_i = K(\otimes !A_i) \to K(\otimes !B_j) = \otimes !KB_j.$$

The reader may verify that because K is strict and commutes with δ, this defines a map of coalgebras. Clearly H is a strict map of SMC's and the two required diagrams commute on the nose. □

Definition 1.17. *We define the category of DILL models to be the full subcategory of the category of linear adjunctions on the objects equivalent to the objects induced by linear categories as in Lemma 1.15*

If we write out the definition above, then a DILL model is a linear adjunction $\begin{tikzcd} \mathbb{C} \arrow[r, bend right, "F"'] \arrow[r, phantom, "\perp"] & \mathbb{D} \arrow[l, bend right, "G"'] \end{tikzcd}$ such that there exists maps of SMC's H, K as in

$$\begin{tikzcd}
 & \mathbb{C}^\star_{GF} \arrow[dl, bend right, "U^\star_{GF}"'] \arrow[d, shift left, "H"] \\
\mathbb{C} \arrow[ur, bend right, "F^\star_{GF}"'] \arrow[r, bend right, "F"'] & \mathbb{D} \arrow[l, bend right, "G"'] \arrow[u, shift left, "K"]
\end{tikzcd}$$

such that H, K is an equivalence of categories and such that

$$G \cong U^\star_{GF} K, \quad F^\star_{GF} \cong KF, \quad GH \cong U^\star_{GF}, \quad HF^\star_{GF} \cong F.$$

Notice that out of these four equations, the first two are equivalent to the last two using the assumption that (H, K) is an equivalence of categories.

Clearly DILL-models provide sound models of DILL, but they are in fact also complete [6].

Remark 1.18. *In [6] the category of DILL-models is defined by requiring that the cartesian category* is $\mathbb{C}^\star_!$, *and not just is equivalent to it. The authors of [6] then argue that DILL provides the internal language of the DILL-models meaning that the category of DILL models is equivalent to the category of DILL theories with translations as morphisms. With our definition of DILL model, we still have a functor constructing the internal language of a model and a functor constructing the classifying model of a theory. For any theory, the internal language of the classifying model is isomorphic to the original theory, and for any model, the classifying model of the internal language is equivalent to the original model.*

Proposition 1.19. *Given two DILL-models and a morphism between the two corresponding linear categories, there exists an extension of this morphism to a morphism of DILL-models. This extension is unique up to isomorphism.*

Proof. The map is up to equivalence given by Lemma 1.16. □

We now give two examples of DILL-models. The first is a practical reformulation of the category $\mathbb{C}^\star_!$ and the second (Proposition 1.21) handles a special case in which $\mathbb{C}_!$ is equivalent to $\mathbb{C}^\star_!$.

We now give a different definition of the category $\mathbb{C}^\star_!$.

Objects: Finite vectors of objects from $\mathbb{C}$.

Morphisms: A morphism from $(A_i)_i$ to $(B_j)_j$ is a family of morphisms $(f_j \colon \otimes_i !A_i \to B_j)_j$.

Composition: The composite of $(f_j)_j \colon (A_i)_i \to (B_j)_j$ and $(g_k)_k \colon (B_j)_j \to (C_k)_k$ is

$$\otimes_i !A_i \xrightarrow{\otimes_i \delta_{A_i}} \otimes_i !!A_i \xrightarrow{m} !(\otimes_i !A_i) \xrightarrow{\langle !f_j \rangle_j} \otimes_j (!B_j) \xrightarrow{(g_k)_k} (C_k)_k,$$

where $\langle !f_j \rangle_j$ is the pairing of the functions $!f_j$ defined as

$$!(\otimes_i !A_i) \xrightarrow{d} \otimes_j !(\otimes_i !A_i) \xrightarrow{\otimes_j !f_j} \otimes_j !B_j.$$

Identity: The identity on (A_i) is

$$(\ \otimes_i !A_i \xrightarrow{\pi_i} !A_i \xrightarrow{\epsilon} A_i\ )_i,$$

where $\pi_{i_0} \colon \otimes_i !A_i \to A_{i_0}$ is defined as

$$\otimes_i !A_i \xrightarrow{\otimes_{i \neq i_0} e_{A_i} \otimes id} (\otimes_{i \neq i_0} I) \otimes A_{i_0} \xrightarrow{\cong} A_{i_0}.$$

Lemma 1.20. *The description above describes a category. This category is isomorphic to $\mathbb{C}^{\star}_{!}$.*

Proof. To be able to distinguish the two definitions, for the remainder of this proof we denote by $\mathbb{D}$ the definition just above. We prove that there are bijective correspondences between objects and morphisms of $\mathbb{D}$ and $\mathbb{C}^{\star}_{!}$ and that these bijections preserve composition and identity. This way we prove both statements of the lemma simultaneously.

Objects of both $\mathbb{D}$ and $\mathbb{C}^{\star}_{!}$ correspond to finite vectors of objects of $\mathbb{C}$. The correspondence on morphisms is given by

$$\mathrm{Hom}_{\mathbb{D}}((A_i)_i, (B_j)_j) \cong \prod_j \mathrm{Hom}_{\mathbb{D}}((A_i)_i, B_j) \cong \prod_j \mathrm{Hom}_{\mathbb{C}}(\otimes_i !A_i, B_j) \cong \\ \prod_j \mathrm{Hom}_{\mathbb{C}^!}(\prod_i \delta_{A_i}, \delta_{B_j}) \cong \mathrm{Hom}_{\mathbb{C}^!}(\prod_i \delta_{A_i}, \prod_j \delta_{B_j}).$$

In one direction, this correspondence maps a map in $\mathbb{D}$:

$$(f_j \colon \otimes_i !A_i \to B_j)_j$$

to

$$\otimes_i !A_i \xrightarrow{\otimes_i \delta_{A_i}} \otimes_i !!A_i \xrightarrow{m} !\otimes_i !A_i \xrightarrow{d} \otimes_j !\otimes_i !A_i \xrightarrow{\otimes_j !f_j} \otimes_j !B_j$$

as a map in $\mathbb{C}^!$. Going the other way, given a map of coalgebras $f \colon \otimes_i !A_i \to \otimes_j !B_j$, this map corresponds to $(\epsilon \circ \pi_j \circ f)_j$ in $\mathbb{D}$.

Since these processes are inverses of each other, we know for example that

$$\otimes_i !A_i \xrightarrow{\otimes_i \delta_{A_i}} \otimes_i !!A_i \xrightarrow{m} !\otimes_i !A_i \xrightarrow{d} \otimes_j !\otimes_i !A_i \xrightarrow{\otimes_j !f_j} \otimes_j !B_j \xrightarrow{\pi_j} !B_j \xrightarrow{\epsilon} B_j$$

is simply f_j. This allows us to conclude that if we start with two maps $(f_j) \colon (A_i) \to (B_j), (g_k) \colon (B_j) \to (C_k)$ in $\mathbb{D}$, take the corresponding maps in $\mathbb{C}^{\star}_{!}$, compose these and then go back into $\mathbb{D}$, we get exactly the composite of (f_j) and (g_k) as defined in $\mathbb{D}$. This proves that the isomorphism preserves composition. It is clear that the isomorphism preserves identity. □

Proposition 1.21. *Suppose the linear category $(\mathbb{C}, !, e, d)$ has products. Then $\mathbb{C}_!$ is equivalent in the category of SMC's to $\mathbb{C}^{\star}_{!}$ and the usual adjunction between $\mathbb{C}$ and $\mathbb{C}_!$ is a DILL-model.*

Proof. Notice first that there exists a natural isomorphism $\otimes_i !A_i \cong !(\prod_i A_i)$ which is a map of coalgebras, i.e.,

$$\begin{tikzcd}
!(\prod_i A_i) \arrow[rr, "\delta_{\prod A_i}"] \arrow[d, "\cong"] & & !!(\prod_i A_i) \arrow[d, "!(\cong)"] \\
\otimes_i !A_i \arrow[r, "\otimes_i \delta_{A_i}"] & \otimes_i !!A_i \arrow[r, "m"] & !(\otimes_i !A_i)
\end{tikzcd}$$

is commutative. This follows from the Yoneda Lemma and the natural isomorphisms

$$\mathrm{Hom}_{\mathbb{C}^!}(h, \textstyle\prod \delta_{A_i}) \cong \prod \mathrm{Hom}_{\mathbb{C}^!}(h, \delta_{A_i}) \cong \prod \mathrm{Hom}_{\mathbb{C}}(B, A_i) \cong \\ \mathrm{Hom}_{\mathbb{C}}(B, \textstyle\prod A_i) \cong \mathrm{Hom}_{\mathbb{C}^!}(h, \delta_{\prod A_i}),$$

for $h\colon B \to !B$.

We need to check that the adjunction between $\mathbb{C}$ and $\mathbb{C}_!$ is an SMC adjunction. The SMC structure on $\mathbb{C}_!$ is defined by the product $\delta_A \times \delta_B = \delta_{A\times B}$, and we need to check that the functor $U_!\colon \mathbb{C}_! \to \mathbb{C}$ is strong. But

$$U_!(\delta_A \times \delta_B) = U_!(\delta_{A\times B}) = !(A \times B) \cong !A \otimes !B = U_!(\delta_A) \otimes U_!(\delta_B).$$

The equivalence is given by the obvious inclusion of $\mathbb{C}_!$ into $\mathbb{C}_!^\star$, and the map, that maps $\prod_i \delta_{A_i}$ to $\delta_{\prod A_i}$. Clearly the composition starting at $\mathbb{C}_!$ is the identity. The isomorphism between the other composition and the identity is the isomorphism $\delta_{\prod A_i} \cong \prod \delta_{A_i}$ described above. We need to check that the two functors are in fact strong morphisms of SMC's, and that the two equivalences make the right triangles commute as in the text after Definition 1.17.

The inclusion of $\mathbb{C}_!$ into $\mathbb{C}_!^\star$ is strong by the isomorphism constructed above, and the functor the other way is strong, because $(\prod \delta_{A_i}) \times (\prod \delta_{B_j})$ maps to $\delta_{(\prod A_i)\times(\prod B_j)}$ which is isomorphic to the product of the images (in fact equal up to arrangement of parentheses).

Finally, we will check that the triangles mentioned after Definition 1.17 commute up to isomorphism. By the same remark, it suffices to prove that half the triangles commute, so let us only consider the ones for the inclusion $\mathbb{C}_! \to \mathbb{C}_!^\star$. It is easily seen that these triangles commute. □

2 PILL models

Definition 2.1 (The 2-category of fibred SMC's). *A fibred symmetric monoidal category is a fibration together with a fibred functor*

$$\begin{tikzcd}
E \times_B E \arrow[rr, "\otimes"] \arrow[dr] & & E \arrow[dl] \\
& B &
\end{tikzcd}$$

and fibred vertical natural transformations making each fibre into an SMC.

A fibred symmetric monoidal functor is a map of fibrations (F, K)*:*

$$\begin{tikzcd}
\mathbb{E} \arrow[r, "F"] \arrow[d] & \mathbb{E}' \arrow[d] \\
\mathbb{B} \arrow[r, "K"] & \mathbb{B}'
\end{tikzcd}$$

together with vertical fibred natural transformations m, m_I*, such that for each object* Ξ *in* $\mathbb{B}$ *the functor* $(F_\Xi, m_\Xi, (m_I)_\Xi)$ *is an SMC functor. We say that* F *is strong (strict) if* m, m_I *are fibred isomorphisms (identities).*

A fibred symmetric monoidal natural transformation from (H, K) *to* (H', K') *is a natural transformation* *of fibred functors* $(\alpha,\ \beta)$ *as usual, as in*

$$\begin{tikzcd}
E \arrow[r, bend left, "H"] \arrow[r, bend right, "H'"'] \arrow[r, phantom, "\Downarrow \alpha"] \arrow[d, "p"'] & E' \arrow[d, "p'"] \\
B \arrow[r, bend left, "K"] \arrow[r, bend right, "K'"'] \arrow[r, phantom, "\Downarrow \beta"] & B'
\end{tikzcd}$$

such that the usual diagrams are commutative. Notice that these diagrams need not be vertical, for example, the diagram

$$\begin{tikzcd}
H(-) \otimes H(=) \arrow[r, "m^H"] \arrow[d, "\alpha\otimes\alpha"'] & H((-) \otimes (=)) \arrow[d, "\alpha"] \\
H'(-) \otimes H'(=) \arrow[r, "m^{H'}"] & H'((-) \otimes (=))
\end{tikzcd}$$

projects via p' *to*

$$\begin{tikzcd}
Kp(-) \arrow[r, "id"] \arrow[d, "\beta_{(-)}"] & Kp(-) \arrow[d, "\beta_{(-)}"] \\
K'p(-) \arrow[r, "id"] & K'p(-)
\end{tikzcd}$$

since $p(-) = p(=)$ *(so the vertical maps are not vertical . . .)*

Having defined what the 2-category of fibred SMC's is, we can derive the notion of a fibred symmetric monoidal adjunction. We focus on the case of a fibred symmetric adjunction over a specific base category.

The pair of fibred functors F, G in

$$\begin{tikzcd}
\mathbb{D} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{E} \arrow[ll, bend right, "F"'] \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

is called a fibred symmetric monoidal adjunction if

- the two fibrations are fibred symmetric monoidal,
- the two functors F, G are fibred symmetric monoidal,
- there exist fibred vertical symmetric monoidal natural transformations $\epsilon\colon FG \Rightarrow id_{\mathbb{D}}, \eta\colon id_{\mathbb{E}} \Rightarrow GF$ such that in each fibre over $\Xi \in \mathbb{B}$, these are counit and unit of the adjunction $F_\Xi \dashv G_\Xi$.

There is a fibred version of Theorem 1.4.

Theorem 2.2. *A fibred adjunction*

$$\begin{tikzcd}
\mathbb{D} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{E} \arrow[ll, bend right, "F"'] \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

between fibred symmetric monoidal fibrations is a symmetric monoidal fibred adjunction iff F *is strong.*

Proof. The left adjoint of a fibred symmetric monoidal adjunction is strong since it is strong in each fibre. For the other direction, we notice that the constructions of m^G, m^G_I as described after Theorem 1.4 give us fibred natural transformations, which satisfy the desired properties, since they satisfy them in each fibre. □

Definition 2.3. *A fibred linear adjunction is a fibred symmetric monoidal adjunction*

$$\begin{tikzcd}
\mathbb{D} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{E} \arrow[ll, bend right, "F"'] \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

where $\mathbb{D}$ is fibred SMCC and the fibred tensor-product on $\mathbb{E}$ is a fibred cartesian product.

A map of fibred linear adjunctions from

$$\begin{tikzcd}
\mathbb{D} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{E} \arrow[ll, bend right, "F"'] \arrow[dl] & \text{to} & \mathbb{D}' \arrow[rr, bend right, "G'"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{E}' \arrow[ll, bend right, "F'"'] \arrow[dl] \\
& \mathbb{B} & & & & \mathbb{B}' &
\end{tikzcd}$$

is a pair of fibred maps $(H, L)\colon (\mathbb{D} \to \mathbb{B}) \to (\mathbb{D}' \to \mathbb{B}')$ and $(K, L)\colon (\mathbb{E} \to \mathbb{B}) \to (\mathbb{E}' \to \mathbb{B}')$ (over the same map in the base categories) such that (H, L) is a strict map of fibred SMCC's preserving the induced comonad on the nose, and (K, L) is a strong fibred map of SMC's such that the diagrams

$$\begin{tikzcd}
\mathbb{D} \arrow[r, "G"] \arrow[d, "H"] & \mathbb{E} \arrow[r, "F"] \arrow[d, "K"] & \mathbb{D} \arrow[d, "H"] \\
\mathbb{D}' \arrow[r, "G'"] & \mathbb{E}' \arrow[r, "F'"] & \mathbb{D}'
\end{tikzcd}$$

commute up to vertical isomorphism.

A natural transformation of fibred linear adjunctions from

$$((H, L), (K, L)) \text{ to } ((H, L), (K', L))$$

(notice that the (H, L) components of the two maps of fibred linear adjunctions are the same) is a vertical natural transformation from K to K' over L.

Again, this may seem a strange definition of natural transformations, but we have chosen this definition to give us a restrictive notion of equivalence of fibred linear adjunctions.

Definition 2.4. *A fibred linear category is a fibred SMCC with a fibred symmetric monoidal comonad, and fibred symmetric monoidal natural transformations e, d such that for each fibre, the restriction of the data mentioned constitutes a linear category.*

A morphism of fibred linear categories is a strong fibred morphism of SMCC's preserving the comonad structure on the nose as in Definition 1.10. It is called strict, if the functor is a strict fibred symmetric monoidal functor.

Proposition 2.5. *There is a forgetful functor from the category of fibred linear adjunctions to the category of fibred linear categories with strict morphisms.*

Proof. The proof of Proposition 1.14 clearly generalizes. □

On the other hand, suppose we are given a fibred linear category $\mathbb{C} \to \mathbb{B}$ with comonad $!$. We can construct the category of coalgebras for the comonad $\mathbb{C}^!$ as having as objects *vertical* maps $A \to !A$ and the rest of the construction as usual. This gives a fibration $\mathbb{C}^! \to \mathbb{B}$. Likewise, we can construct the co-Kleisli fibration $\mathbb{C}_! \to \mathbb{B}$ by taking each fibre to be the co-Kleisli category of the restriction of the comonad, and letting reindexing be the obvious choice. Finally, we can construct $\mathbb{C}^\star_! \to \mathbb{B}$ fibrewise as we did in the Section 1.3.

Lemma 2.6. *Given a fibred linear category $\mathbb{C}$ with comonad $!$, the fibred adjunction*

$$\begin{tikzcd}
\mathbb{C} \arrow[rr, bend right, "F^\star_!"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{C}^\star_! \arrow[ll, bend right, "U^\star_!"'] \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

is a fibred linear adjunction. This construction extends to a functor which is right inverse to the forgetful functor of Proposition 2.5.

Definition 2.7. *A PILL-model is a fibred linear adjunction*

$$\begin{tikzcd}
\mathbb{C} \arrow[rr, bend right, "F"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbb{D} \arrow[ll, bend right, "G"'] \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

equivalent to (in the category of fibred linear adjunctions) the fibred linear adjunction induced by the comonad GF as in Lemma 2.6 and such that further

- *the category $\mathbb{B}$ is cartesian*
- *the fibration $\mathbb{C} \to \mathbb{B}$ has a generic object projecting to Ω in $\mathbb{B}$, and products with respect to projections $\Xi \times \Omega \to \Xi$ in $\mathbb{B}$.*

The condition of the fibred linear adjunction being equivalent to the fibred linear adjunction induced by the comonad means that there exists maps H, K fibred over $\mathbb{B}$ as in

$$\begin{tikzcd}
& \mathbb{C}^\star_{GF} \arrow[dl, bend right, "U^\star_{GF}"'] \arrow[d, shift left, "H"] \\
\mathbb{C} \arrow[ur, "F^\star_{GF}"'] \arrow[r, bend right, "F"'] & \mathbb{D} \arrow[l, bend right, "G"'] \arrow[u, shift left, "K"]
\end{tikzcd}$$

such that H, K are strong maps of fibred SMC's and constitute a fibred equivalence, and such that the obvious four triangles commute up to vertical isomorphisms.

Definition 2.8. *A morphism of PILL-models is a morphism of fibred linear adjunctions such that the SMCC part of the functor preserves generic object, products in the base and products in the fibration.*

Definition 2.9. *A PILL$_Y$-model is a PILL model with a polymorphic fixed point combinator*

$$Y\colon \prod \alpha\colon \mathsf{Type}.\, (\alpha \to \alpha) \to \alpha.$$

A morphism of PILL$_Y$-models is a morphism of PILL-models preserving Y.

The following Proposition is a trivial generalization of Proposition 1.19.

Proposition 2.10. *Given two PILL-models and a morphism of fibred linear categories between the corresponding fibred linear categories preserving generic object, products in the base and the simple products, there exists an extension of this map to a map of PILL-models. The extension is unique up to vertical isomorphism.*

Lemma 2.11. *The fibration $\mathbb{C}^\star_! \to \mathbb{B}$ is isomorphic to the fibration obtained by defining each fibre as in Lemma 1.20 and defining reindexing to be the obvious choice.*

Proof. The two fibrations are fibrewise isomorphic by Lemma 1.20, and we just need to check that the isomorphism commutes with reindexing, which is obvious. □

Proposition 2.12. *Suppose the linear fibration $\mathbb{C} \to \mathbb{B}$ with comonad $!$ has fibrewise products. Then the usual fibred adjunction between $\mathbb{C}$ and $\mathbb{C}_!$ is a fibred linear adjunction, and there exists an equivalence of fibred linear adjunctions between this and the fibred adjunction between $\mathbb{C}$ and $\mathbb{C}^\star_!$.*

Proof. The proof of Proposition 1.21 clearly generalizes. □

References

[1] A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, Edinburgh University, 1997. (document), 1.2

[2] P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical report, University of Cambridge, 1995. 1.1, 1.2

[3] Masahito Hasegawa. Categorical glueing and logical predicates for models of linear logic. 1999.

[4] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999.

[5] G. M. Kelly. Doctrinal adjunction. In *Category Seminar (Proc. Sem., Sydney, 1972/1973)*, pages 257–280. Lecture Notes in Math., Vol. 420. Springer, Berlin, 1974. 1.1

[6] Maria E Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. Technical Report CSR-01-7, University of Birmingham, School of Computer Science, August 2001. (document), 1.3, 1.18

[7] Paola Maneggia. *Models of Linear Polymorphism*. PhD thesis, University of Birmingham, Feb. 2004. (document)

[8] Paul-André Melliès. Categorical models of linear logic revisited. *Theoretical Computer Science*. To appear. 1.1

Synthetic Domain Theory and Models of Linear Abadi & Plotkin Logic

Rasmus Ejlers Møgelberg
Lars Birkedal
Giuseppe Rosolini

Abstract

In a recent article [3] the first two authors and R.L. Petersen have defined a notion of parametric LAPL-structure. Such structures are parametric models of the equational theory PILL_Y, a polymorphic intuitionistic / linear type theory with fixed points, in which one can reason using parametricity and, for example, solve a large class of domain equations [3, 4].

Based on recent work by Simpson and Rosolini [14] we construct a family of parametric LAPL-structures using synthetic domain theory and use the results of *loc. cit.* and results about LAPL-structures to prove operational consequences of parametricity for a strict version of the Lily programming language. In particular we can show that one can solve domain equations in the strict version of Lily up to ground contextual equivalence.

Contents

1 Introduction

It was first realized by Plotkin [10, 9] that PILL_Y, a polymorphic type theory with linear as well as intuitionistic variables and fixed points, combined with relational parametricity has surprising power, in that one can define recursive types in the theory. This theory can be seen as an approach to axiomatic domain theory where the concept of linear and intuitionistic maps correspond to strict and non-strict continuous maps between domains. In this approach recursive domain equations are solved using polymorphism instead of the traditional limit-colimit construction.

In [10] Plotkin also sketched a logic for reasoning about parametricity for PILL_Y (the logic is a variant of Abadi & Plotkin's logic for parametricity [11]) and how to solve domain equations for PILL_Y and prove correctness of the solutions in the logic using parametricity.

Recently the first two authors together with R.L. Petersen have given a detailed presentation of the logic sketched by Plotkin and defined the categorical notion of parametric LAPL-structure (Linear Abadi-Plotkin Logic), which are models of the logic [3, 4]. Using Plotkin's constructions one can solve recursive domain equations in LAPL-structures. In *loc. cit.* a concrete domain theoretical LAPL-structure based on admissible pers on a reflexive domain is constructed, and in [7] a parametric completion process along the lines of [12] is presented constructing parametric LAPL-structures out of a large class of models of PILL_Y.

In recent work Simpson and Rosolini [14] have constructed an interpretation (or rather a family of interpretations) of $\text{Lily}_{\text{strict}}$ — a strict version of Lily [1] — based on Synthetic Domain Theory (SDT). The interpretation uses a class of domains in an intuitionistic set theory, and the type constructors are interpreted using simple set-theoretic constructions. It is a result of SDT that such a theory has models, and for each such model the construction of [14] gives an interpretation of $\text{Lily}_{\text{strict}}$, but one does not have to know the details of these models to use the interpretation.

Simpson and Rosolini further show how one can use the interpretation to prove operational properties of $\text{Lily}_{\text{strict}}$. In particular, they prove a version of the strictness theorem for Lily [1] for the new language $\text{Lily}_{\text{strict}}$. The strictness theorem states that the two versions of ground contextual equivalence obtained by observing termination at lifted types for a call-by-value and a call-by-name operational semantics coincide. They show that the interpretation is adequate with respect to this ground contextual equivalence.

In this paper we present a parametric LAPL-structure based on the interpretation of $\text{Lily}_{\text{strict}}$ of [14]. We have three motivations for this work. First of all, we would like to show that the concept of parametric LAPL-structure is general enough to incorporate many different models. As mentioned we have already constructed a concrete domain-theoretic parametric LAPL-structure and shown how to construct parametric LAPL-structures from PILL_Y-models using a parametric completion process. In a future paper we intend to construct a parametric LAPL-structure using operational semantics of Lily, showing that the parametric reasoning used in [1] can be presented as reasoning in an LAPL-structure.

Our second motivation is that the interpretation presented in [14] is parametric and thus one should be able to solve recursive domain equations in it. Proving that the interpretation gives rise to an LAPL-structure provides a formal proof of this.

Our third motivation is that we can use the LAPL-structure and the adequacy of the interpretation of $\text{Lily}_{\text{strict}}$ to show formally consequences of parametricity for $\text{Lily}_{\text{strict}}$. This builds upon the idea from [14] of giving denotational proofs of the theorems in [1], and extends it to prove properties not included in [1].

We assume that the reader is familiar with LAPL-structures but assume no knowledge of synthetic domain theory. In Section 2 we introduce synthetic domain theory as presented in [14], constructing a category of domains. In Sections 4-6 we present the LAPL-structure. We first present a model of PILL_Y based on the category of domains, then we create a parametric version of this model, and finally we construct the full parametric LAPL-structure.

In Section 7 we show how to use the parametric LAPL-structure to reason about Lily$_{\text{strict}}$. In particular, we show how to solve recursive domain equations in Lily$_{\text{strict}}$. First, however, we present the language and sketch the results of [14].

In Appendix A we address the following question: It is well known [10] that in PILL$_Y$, using parametricity, the type for tensor products can be expressed using the other constructions of the language:

$$\sigma \otimes \tau \cong \prod \alpha . \, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha .$$

Does this mean that if one leaves out tensor products of PILL$_Y$, then one obtains a language as expressive as the original PILL$_Y$? In particular, there could exist terms in PILL$_Y$, that could not be expressed without using let-expressions, which of course cannot exist in the language without tensor products. The answer to the question is yes, and the appendix is included here because this result is needed for solving recursive domain equations in Lily$_{\text{strict}}$.

Acknowledgments. We thank Alex Simpson for helpful discussions.

2 Synthetic Domain Theory

The idea of synthetic domain theory as originally conceived by Dana Scott is to consider domains as simply special sets, and maps between them as all set-theoretic maps. Of course, one of the points of classical domain theory is that all continuous maps have fixed points, and so all set-theoretic maps between domains should have fixed points. Classically this entails that all domains are trivial, since if X is a domain and $x, y \in X$, $x \neq y$ we can define the map $f \colon X \to X$ as

$$f(z) = \begin{cases} y & z = x \\ x & z \neq x \end{cases} .$$

Clearly this f cannot have a fixed point. This argument does not hold in intuitionistic set theory since the map f is not constructively definable. In fact, in some models of intuitionistic set theory interesting classes of sets with the property that all endofunctions have fixed points do exist.

In this section we recall the approach to synthetic domain theory (SDT) presented in [14]. In fact we follow [14] closely, but add some details and proofs to assist the reader. In their paper, Simpson and Rosolini work informally in an intuitionistic set theory, which formally may be taken to be IZF [15]. For later purposes, we will need to be a bit more refined, but we postpone this issue to Section 4. Simpson and Rosolini's axioms for SDT assume given a class of special sets called predomains satisfying certain conditions. Domains are then defined to be pointed pre-domains, and another axiom states that all endomaps on domains have fixed points.

We emphasize that there do exist models of SDT as presented here. For example, SDT can be modeled in any realizability topos satisfying the *strong completeness axiom* of [6], by taking predomains to be the well-complete objects.

In the rest of this paper we will use synthetic domain theory to construct an LAPL-structure. To be precise, the construction actually gives us a large family of LAPL-structures, since we get one for each model of SDT.

2.1 Pointed sets

Consider the powerset of the one-point set $1 = \{\emptyset\}$:

$$\Omega = P(1).$$

In intuitionistic set theory, Ω is not just the set $\{\emptyset, \{\emptyset\}\}$ as it is classically. In fact for every proposition p one can associate the element

$$\{\emptyset \mid p\} \in \Omega$$

and for each element $X \in \Omega$ one can associate the proposition $\emptyset \in X$, and these associations are each others inverses up to provable equivalence of propositions. Motivated by this, Ω is called *the set of truth values*.

Simpson and Rosolinis first axiom is that a set $\Sigma \subseteq \Omega$ of truth values is given. The set Σ is to be thought of as the set of truth values of propositions of the form "P terminates", for programs P.

Axiom 2.1. *The subset $\Sigma \subseteq \Omega$ is a dominance [13], i.e.,*

- $\top \in \Omega$.
- *If $p \in \Sigma, q \in \Omega$ and $p \supset (q \in \Sigma)$ then $p \wedge q \in \Sigma$.*

For each $p \in \Sigma$ consider the set

$$X^p = \{A \subseteq X \mid (\forall x, x' \in A.\, x = x') \wedge ((\exists x \in A) \supset\subset p)\},$$

i.e., X^p is the set of subsingleton subsets A of X which are inhabited (i.e., $\exists x \in A$) with truth value p. There is a canonical isomorphism $X \cong X^\top$.

Definition 2.2 ([14]). *A pointed set is a pair $(X, (r_p)_{p\in\Sigma})$ where X is a set and for each $p \in \Sigma$, $r_p \colon X^p \to X$ is a map such that*

- *for all $x \in X$, $r_\top(\{x\}) = x$*
- *for all $p, q \in \Sigma$, $e \in X^{p\wedge q}$,*
$$r_{p\wedge q}(e) = r_p(\{r_{p\wedge q}(e) \mid p\})$$

The definition above is a generalization of the classical concept of pointed set. In that case $\Sigma = \{\bot, \top\}$, and a pointed set is a set X with two functions $r_\top, r_\bot$. The first condition of Definition 2.2 tells us that $r_\top$ is the isomorphism $X^\top \cong X$ and since $X^\bot = \{\emptyset\}$, $r_\bot$ simply corresponds to a point in X.

A strict map is simply a map preserving the pointed structure.

Definition 2.3 ([14]). *A* strict *map from a pointed set $(X, (r_p)_{p\in\Sigma})$ to a pointed set $(Y, (s_p)_{p\in\Sigma})$ is a map $f \colon X \to Y$ such that for all $p \in \Sigma$ and $e \in X^p$*

$$f(r_p(e)) = s_p(\{f(x) \mid x \in e\})$$

Following [14], we will often leave the pointed structure implicit and simply write X for a pointed set $(X, (r_p)_{p\in\Sigma})$. Write $f \colon X \multimap Y$ to denote that f is a pointed map, and write $X \multimap Y$ for the set of pointed maps from X to Y. Simpson and Rosolini define a subset $X' \subset X$ to be a subpointed set of X if for all $p \in \Sigma$ and $e \in (X')^p$ the point $r_p(e)$ is in X', where $(r_p)_{p\in\Sigma}$ is the pointed structure on X.

Lemma 2.4. *For any pair of strict maps $f, g\colon X \multimap Y$, the equalizer of f and g*

$$\{x \in X \mid f(x) = g(x)\}$$

is a subpointed set of X.

Proof. Define $E = \{x \in X \mid f(x) = g(x)\}$. For any $p \in \Sigma$, $e \in E^p$ we must show that $r_p^X(e) \in E$. But

$$f(r_p^X(e)) = r_p^Y(\{f(x) \mid x \in e\}) = r_p^Y(\{g(x) \mid x \in e\}) = g(r_p^X(e)).$$

□

Lemma 2.5. *For all sets X and families of pointed sets $(Y_x, (r_p^{Y_x})_{p\in\Sigma})_{x\in X}$, the family of maps*

$$r_p^{\prod_{x\in X} Y_x}(e) = (r_p^{Y_x}(\{\pi_x(z) \mid z \in e\}))_{x\in X}$$

where $\pi_x\colon \prod_{x\in X} Y_x \to Y_x$ denotes the projection, define a pointed structure on $\prod_{x\in X} Y_x$. This defines a categorical product in the category of pointed sets and strict maps.

Proof. Suppose $e \in (\prod_{x\in X} Y_x)^{p\wedge q}$ for some $p, q \in \Sigma$. Then

$$\begin{gathered} r_{p\wedge q}^{\prod_x Y_x}(e) = (r_{p\wedge q}^{Y_x}(\{\pi_x(z) \mid z \in e\}))_{x\in X} = \\ (r_p^{Y_x}(\{r_{p\wedge q}^{Y_x}(\{\pi_x(z) \mid z \in e\}) \mid p\}))_{x\in X} = r_p^{\prod_x Y_x}(\{r_{p\wedge q}^{\prod_x Y_x}(e) \mid p\}). \end{gathered}$$

This proves that the maps define a pointed structure. Since any pairing of strict maps is strict, this defines a product in the category of pointed sets and strict maps. □

Lemma 2.6. *For any set X and all pointed sets $(Y, r_p^Y)_{p\in\Sigma}$, the maps $r_p^{X\to Y}$ defined by*

$$r_p^{X\to Y}(e) = (x \mapsto r_p^Y(\{f(x) \mid f \in e\}))$$

define a pointed structure on $X \to Y$. If, moreover, X is pointed, then the set $X \multimap Y$ is a subpointed set of $X \to Y$.

Proof. As always $X \to Y \cong \prod_{x\in X} Y$ and the pointed structure defined above is just the product structure. We proceed to show that $X \multimap Y$ is a subpointed set of $X \to Y$.

Define the set $LX = \bigcup_{p\in\Sigma} X^p$. The set $X \multimap Y$ is the equalizer of the two maps

$$\phi, \psi\colon (X \to Y) \to (LX \to Y)$$

defined as

$$\begin{aligned} \phi(f)(e) &= r_p^Y(\{f(x) \mid x \in e\}) \\ \psi(f)(e) &= f(r_p^X(e)) \end{aligned}$$

for $e \in X^p$. By Lemma 2.4 it now suffices to show that ϕ, ψ are strict. Suppose $e \in X^p$ and $f \in (X \to Y)^q$. Then

$$\begin{gathered} \phi(r_q^{X\to Y}(f))(e) = r_p^Y(\{r_q^{X\to Y}(f)(x) \mid x \in e\}) = \\ r_p^Y(\{r_q^Y(\{g(x) \mid g \in f\}) \mid x \in e\}. \end{gathered}$$

Now, suppose $y \in \{r_q^Y(\{g(x) \mid g \in f\}) \mid x \in e\}$ then there exists an $x \in e$, i.e., p holds and $y = r_{p\wedge q}^Y(\{g(x) \mid g \in f\})$. On the other hand if

$$y \in \{r_{q\wedge p}^Y(\{g(x) \mid g \in f \wedge x \in e\}) \mid \exists x \in e\}$$

then p holds and $y = r^Y_q(\{g(x) \mid g \in f\})$. Thus

$$\{r^Y_q(\{g(x) \mid g \in f\}) \mid x \in e\} = \{r^Y_{p\wedge q}(\{g(x) \mid g \in f \wedge x \in e\} \mid \exists x \in e\}$$

and so

$$\phi(r^{X\to Y}_q(f))(e) = r^Y_p(\{r^Y_{p\wedge q}(\{g(x) \mid g \in f \wedge x \in e\} \mid \exists x \in e\}) = \\ r^Y_{p\wedge q}(\{g(x) \mid g \in f \wedge x \in e\}).$$

Likewise

$$r^{LX\to Y}_q(\{\phi(g) \mid g \in f\})(e) = r^Y_q(\{\phi(g)(e) \mid g \in f\}) = \\ r^Y_q(\{r^Y_p(\{g(x) \mid x \in e\}) \mid g \in f\}) = \\ r^Y_q(\{r^Y_{p\wedge q}(\{g(x) \mid x \in e \wedge g \in f\}) \mid \exists g \in f\}) = r^Y_{p\wedge q}(\{g(x) \mid x \in e \wedge g \in f\})$$

and so ϕ is strict.

To see that ψ is strict, compute

$$\psi(r^{X\to Y}_q(f))(e) = r^{X\to Y}_q(f)(r^X_p(e)) = \\ r^Y_q(\{g(r^X_p(e)) \mid g \in f\})$$

and

$$r^{LX\to Y}_q(\{\psi(g) \mid g \in f\})(e) = r^Y_q(\{\psi(g)(e) \mid g \in f\}) = \\ r^Y_q(\{g(r^Y_p(e)) \mid g \in f\}).$$

So ψ is strict, and we conclude that $X \multimap Y$ is an equalizer of strict maps from $X \to Y$ and so is subpointed. □

However, neither → nor $\multimap$ define a cartesian closed structure on the category of pointed sets and strict maps. Clearly → defines a cartesian closed structure on the category of pointed sets and all maps, and as will be shown later $\multimap$ is part of a symmetric monoidal closed structure on a category of domains.

Simpson and Rosolini introduce a free pointed structure $(LX, (\mu_p)_{p\in\Sigma})$ on any set X as

$$LX = \bigcup_{p\in\Sigma} X^p, \qquad \mu_p(E) = \bigcup E,$$

i.e., for E in $(LX)^p$, $\mu_p(E) = \{x \in e \mid e \in E\}$.

Lemma 2.7. *For all X, $(LX, (\mu_p)_{p\in\Sigma})$ is a pointed set.*

Proof. We first show that for all E, $\mu_p(E) \in LX$, i.e., $(\exists x \in \mu_p(E)) \in \Sigma$. The set $\mu_p(E)$ is inhabited iff there exists $e \in E$ and $x \in e$, but if $e \in E$, then $(\exists x \in e) \in \Sigma$. This means that setting $q = \exists x \in \mu_p(E)$, we have $p \supset (q \in \Sigma)$ and so by Σ being a dominance, we have $p \wedge q \in \Sigma$. Since $q \supset p$, $p \wedge q = q$.

We check that $(\mu_p)_p$ defines a pointed structure. Clearly $\mu_\top(\{e\}) = e$. If $E \in (LX)^{p\wedge q}$ consider the equation

$$\mu_{p\wedge q}(E) = \mu_p(\{\mu_{p\wedge q}(E) \mid p\}).$$

For any of the two sides to be inhabited $p \wedge q$ must hold, and in this case both sides reduce to E. □

For $f\colon X \to Y$, define

$$Lf\colon LX \to LY$$

by

$$Lf(e) = \{f(x) \mid x \in e\}.$$

Proposition 2.8. *The construction $L(-)$ defines a functor from the category of sets to the category of pointed sets with strict maps. This functor is left adjoint to the forgetful functor.*

Proof. For functoriality we simply check that $L(f)$ is pointed. Suppose $E \in (LX)^p$, then

$$\mu_p(\{L(f)(e) \mid e \in E\}) = \mu_p(\{\{f(x) \mid x \in e\} \mid e \in E\}) = \{f(x) \mid \exists e \in E.\, x \in e\}$$

and

$$L(f)(\mu_p(E)) = L(f)\{x \mid \exists e \in E.\, x \in e\} = \{f(x) \mid \exists e \in E.\, x \in e\}.$$

So $L(f)$ is pointed.

The adjoint correspondence associates to each strict map $f\colon LX \multimap Y$ the set theoretic map $x \mapsto f(\{x\})$. To prove that this association is injective, we prove that pointed maps out of LX are uniquely determined by their values on singletons. Suppose that $g\colon LX \to Y$ is pointed, and $e \in X^p$. Since

$$e = \mu_p(\{\{x\} \mid x \in e\})$$

we have

$$g(e) = g(\mu_p(\{\{x\} \mid x \in e\})) = r^Y_p(\{g(\{x\}) \mid x \in e\}).$$

To show that the correspondence is surjective, suppose $f\colon X \to Y$ is a set theoretic map, and consider $\hat{f}\colon LX \to Y$ given as $\hat{f}(e) = r^Y_p(\{f(x) \mid x \in e\})$. Clearly $\hat{f}(\{x\}) = f(x)$, and so we just need to show that $\hat{f}$ is pointed. Suppose $E \in (LX)^p$ and define $q = (\exists x \in \mu_p(E))$. Then

$$\hat{f}(\mu_p(e)) = \hat{f}(\{x \mid \exists e \in E.\, x \in e\}) = r^Y_q(\{f(x) \mid \exists e \in E.\, x \in e\}).$$

On the other hand

$$\begin{gathered} r^Y_p(\{\hat{f}(e) \mid e \in E\}) = r^Y_p(\{r_{\exists x \in e}(\{f(x) \mid x \in e\}) \mid e \in E\}) = \\ r^Y_p(\{r^Y_q(\{f(x) \mid \exists e \in E \wedge x \in e\}) \mid p\}). \end{gathered}$$

Since $q \supset p$, $p \wedge q = q$ and so by the last rule of Definition 2.2 the last part is simply

$$r^Y_q(\{f(x) \mid \exists e \in E \wedge x \in e\}),$$

which proves that $\hat{f}$ is pointed. □

Lemma 2.9. $\Sigma \cong L1$ *and so has a pointed structure.*

2.2 Domains and predomains

As said, one of Simpson and Rosolini's axioms states that a class of special sets called predomains is given.

Axiom 2.10. *There is a class of sets* $\mathbf{Predom}$ *called* predomains *such that*

- *If $A \cong B$ and A is a predomain, then so is B.*
- *For any set-indexed family of predomains $(A_x)_{x \in X}$ the product $\prod_{x \in X} A_x$ is a predomain.*
- *For any pair of functions $f, g\colon A \to B$ between predomains, the equalizer of f and g is a predomain.*
- *The set of natural numbers $\mathbb{N}$ is a predomain.*

- *If A is a predomain, so is LA.*

Lemma 2.11. *The sets 1 and Σ are predomains.*

Proof. The set 1 is the empty product and so is a predomain. The set Σ is isomorphic to $L1$. □

Since predomains will be used to model polymorphism it would be desirable to have a *set* of all predomains, such that products can be defined over this set. This is unfortunately too much to ask for, so instead Simpson and Rosolini ask for the existence of a set of predomains containing representatives of each isomorphism class of predomains.

Axiom 2.12. *There exists a* set *of predomains $\mathbf{P}$ such that for any predomain A there exists a predomain $B \in \mathbf{P}$ such that $A \cong B$.*

Definition 2.13 ([14]). *A **domain** is a pointed predomain. Denote by $\mathbf{Dom}_\perp$ the category of domains with strict maps and by $\mathbf{Dom}$ the category of domains with all maps. By $\mathbf{D}$ denote the set of pointed structures on objects of $\mathbf{P}$, i.e.,*

$$\mathbf{D} = \{(B, (r_p)_{p\in\Sigma}) \mid B \in \mathbf{P}, (r_p)_{p\in\Sigma} \textit{ is a pointed structure on } B\}$$

Clearly the set $\mathbf{D}$ has the property that for all $A \in \mathbf{Dom}_\perp$, there exists an element $B \in \mathbf{D}$ such that $A \cong B$ in the category $\mathbf{Dom}_\perp$.

Simpson and Rosolini's last axiom states that all endomaps on domains have fixed points.

Axiom 2.14. *For every domain A there is a function $\mathit{fix}_A \colon (A \to A) \to A$ such that*

- *fix_A gives fixed points, i.e., for any $f\colon A \to A$,*

$$f(\mathit{fix}_A(f)) = \mathit{fix}_A$$

- *The maps fix_A satisfy a uniformity property, i.e., for any domain B and any set of maps $f\colon A \to A$, $g\colon B \to B$, $h\colon A \multimap B$ such that*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"', multimap] & A \arrow[d, "h", multimap] \\ B \arrow[r, "g"] & B \end{tikzcd}$$

commutes, $h(\mathit{fix}_A(f)) = \mathit{fix}_B(g)$.

3 The category of domains

In this section, it is shown that $\mathbf{Dom}$ is cartesian closed, $\mathbf{Dom}_\perp$ is symmetric monoidal closed, and there is a symmetric monoidal adjunction

$$\begin{tikzcd} \mathbf{Dom}_\perp \arrow[r, bend right] \arrow[r, phantom, "\perp"] & \mathbf{Dom} \arrow[l, bend right] \end{tikzcd}$$

where the left adjoint is the lifting functor and the right adjoint is the forgetful functor. This will prove that there is a linear structure on $\mathbf{Dom}_\perp$. These results are basically taken from [14].

Lemma 3.1. *The category* $\mathbf{Dom}_\perp$ *is complete.*

Proof. This is immediate from Axiom 2.10 and Lemmas 2.5, 2.4. □

Lemma 3.2. *If* A, B *are predomains, then so is* $A \to B$*. If* A, B *are domains, then so is and* $A \multimap B$*.*

Proof. For the first part $A \to B \cong \prod_{x \in A} B$ and so $A \to B$ is a predomain by Axiom 2.10.
For the second part, notice that in the proof of Lemma 2.6 it is shown that $A \multimap B$ is the equalizer of pointed maps between domains $A \to B$ and $LA \to B$, and so by Lemma 3.1 is a domain. □

This lemma has two corollaries.

Corollary 3.3. *The category* $\mathbf{Dom}$ *is cartesian closed.*

Corollary 3.4. $(-) \multimap (=)$ *defines a functor* ${\mathbf{Dom}_\perp}^{\mathrm{op}} \times \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$*.*

Proof. It only remains to check that for each strict map $f \colon A \multimap A'$ between domains, and for each domain B, the maps $(f \multimap B) \colon (A' \multimap B) \to (A \multimap B)$ and $(B \multimap f) \colon (B \multimap A) \to (B \multimap A')$ are strict. This is an easy exercise. □

Definition 3.5 ([14]). *Suppose* $f \colon A \times B \to C$ *is a map between domains. Say that* f *is* strict in the first variable *if for all* $p \in \Sigma, e \in A^p$, $y \in B$

$$f(r^A_p(e), y) = r^C_p(\{f(x, y) \mid x \in e\}).$$

Likewise one can define what it means for f *to be strict in the second variable. The map* f *is called* bistrict *if it is strict in both variables.*

The $\otimes$ part of the symmetric monoidal structure on $\mathbf{Dom}_\perp$ (to be defined below) will satisfy the universal property that strict maps out of $A \otimes B$ correspond bijectively to bistrict maps out of $A \times B$.

Lemma 3.6. *Bistrict maps are strict.*

Proof. Suppose $f \colon A \times B \to C$ is bistrict. Using the isomorphism $(A \times B)^p \cong A^p \times B^p$ we must show that

$$f(r^p_A(e), r^p_A(e)) = r^p_C(\{f(x, y) \mid x \in e, y \in f\})$$

for any $p \in \Sigma$, $e \in A^p$, $f \in B^p$. We compute

$$f(r^A_p(e), r^B_p(g)) = r^C_p(\{f(x, r^B_p(g)) \mid x \in e\}) = r^C_p(\{r^C_p\{f(x, y) \mid y \in g\} \mid x \in e\})$$

Since

$$\{r^C_p\{f(x, y) \mid y \in g\} \mid x \in e\} = \{f(x, y) \mid x \in e, y \in g\}$$

we get

$$f(r^A_p(e), r^B_p(g)) = r^C_p(\{f(x, y) \mid x \in e, y \in g\})$$

which shows that f is strict. □

Strict maps are not necessarily bistrict, for example projections are in general strict but not bistrict.

Lemma 3.7. *If $f\colon A \times B \multimap C$ is bistrict and $a\colon A' \multimap A, b\colon B' \multimap B, c\colon C \multimap C'$ are strict maps, then*

$$c \circ f \circ (a \times b)$$

is bistrict.

Proof. This follows easily by direct calculation. □

Lemma 3.8. *Strict maps from A to $B \multimap C$ correspond by currying to bistrict maps $A \times B \multimap C$.*

Proof. First assume that $f\colon A \multimap (B \multimap C)$. We show that $\hat{f}\colon A \times B \to C$ is bistrict. If $e \in A^p, y \in B$ then

$$\hat{f}(r_p^A(e), y) = f(r_p^A(e))(y) = r_p^{B \multimap C}(\{f(x) \mid x \in e\})(y) = \\ r_p^C(\{f(x)(y) \mid x \in e\}) = r_p^C(\{\hat{f}(x,y) \mid x \in e\})$$

and if $e \in B^p, x \in A$,

$$\hat{f}(x, r_p^B(e)) = f(x)(r_p^B(e)) = r_p^C(\{f(x)(y) \mid y \in e\}) = r_p^C(\{\hat{f}(x,y) \mid y \in e\}),$$

using that $f(x)$ is strict.

Assume on the other hand that $\hat{f}$ is bistrict. We first show that for all $x \in A$, $f(x)$ is strict:

$$f(x)(r_p^B(e)) = \hat{f}(x, r_p^B(e)) = r_p^C(\{\hat{f}(x,y) \mid y \in e\}) = r_p^C(\{f(x)(y) \mid y \in e\}).$$

To show that f is strict, we must show that for $e \in A^p, y \in B$, $f(r_p^A(e))(y) = r_p^{B \multimap C}(\{f(x) \mid x \in e\})(y)$. But by definition of $r_p^{B \multimap C}$,

$$r_p^{B \multimap C}(\{f(x) \mid x \in e\})(y) = r_p^C(\{f(x)(y) \mid x \in e\}) = \\ r_p^C(\{\hat{f}(x,y) \mid x \in e\}) = \hat{f}(r_p^A(e), y) = f(r_p^A(e))(y).$$

□

Lemma 3.9. *There exists a functor $(-) \otimes (=)\colon \mathbf{Dom}_\perp \times \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$ and a domain I giving $\mathbf{Dom}_\perp$ an SMCC-structure.*

Proof. For each domain B the functor $B \multimap (-)$ defined on the category of domains preserves small limits. The existence of the set $\mathbf{D}$ tells us that the solution set condition is satisfied, and so by the Adjoint Functor Theorem, $B \multimap (-)$ has a left adjoint $B \otimes (-)$.

Using Lemma 3.8 we see that

$$A \multimap (B \multimap C) \cong B \multimap (A \multimap C)$$

and thus $A \otimes B \cong B \otimes A$. Thus we can define $(-) \otimes (=)$ as a functor in two variables.

The domain I is defined as $L(1)$ (which by the way is Σ). This defines a unit for the tensor since

$$I \otimes A \multimap B \cong I \multimap (A \multimap B) \cong 1 \to (A \multimap B) \cong A \multimap B.$$

□

Lemma 3.8 gives a correspondence between strict maps $A \otimes B \multimap C$ and bistrict maps $A \times B \multimap C$, natural in C. This correspondence is of course given by a universal map, which is the subject of the next lemma.

Lemma 3.10. *There exists a natural transformation $\eta\colon (-) \times (=) \to (-) \otimes (=)$ such that the correspondence between strict maps out of the tensor and bistrict maps out of the product is given by composition with this natural transformation. Each component of the natural transformation is bistrict and thus strict.*

Proof. The component of the unit of the adjunction $(-) \otimes B \dashv B \multimap (-)$ at A is a strict map from A to $B \multimap A \otimes B$, which corresponds to a bistrict map $\eta\colon A \times B \multimap A \otimes B$. This map induces the correspondence between bistrict maps out of $A \times B$ and strict maps out of $A \otimes B$, and we proceed to show that η is natural.

Naturality of the unit is the commutative diagram

$$\begin{tikzcd}
A \arrow[r, -o] \arrow[d, -o, "f"'] & (B \multimap (A \otimes B)) \arrow[d, -o, "B \multimap (f \otimes B)"] \\
A' \arrow[r, -o] & (B \multimap (A' \otimes B))
\end{tikzcd}$$

which gives naturality in the first variable of η. Naturality in the second variable follows by symmetry. □

Lemma 3.11. *The forgetful functor $U\colon \mathbf{Dom}_\perp \to \mathbf{Dom}$ is a symmetric monoidal functor with respect to the cartesian closed structure on $\mathbf{Dom}$.*

Proof. We need to construct the natural transformation $m^U\colon (-) \times (=) \to (-) \otimes (=)$ and the map $m^U_I\colon 1 \to I$ satisfying the requirements of [8, Definition 1.1]. Define the natural transformation m^U to be η of Lemma 3.10, and define m^U_I to be the unit of the adjunction of Proposition 2.8 at 1, i.e., $m^U_I\colon 1 \to I$ is the map giving the correspondence between strict maps out of I and general maps out of 1.

We need to check that these maps satisfy the requirements of [8, Definition 1.1]. So first we need to show that the compositions

$$(X \times Y) \times Z \xrightarrow{\eta \times id} (X \otimes Y) \times Z \xrightarrow{\eta} (X \otimes Y) \otimes Z \xrightarrow{\cong} X \otimes (Y \otimes Z)$$

and

$$(X \times Y) \times Z \xrightarrow{\cong} X \times (Y \times Z) \xrightarrow{id \times \eta} X \times (Y \otimes Z) \xrightarrow{\eta} X \otimes (Y \otimes Z)$$

are equal for all domains X, Y, Z. But both compositions induce the same bijective natural correspondence between maps

$$(X \times Y) \times Z \to W$$

strict in each variable and strict maps

$$X \otimes (Y \otimes Z) \multimap W$$

so these two maps are equal. For the diagrams

$$\begin{tikzcd}
1 \times X \arrow[r, "\cong"] \arrow[d, "m^U_I \times id"'] & X \arrow[d, "\cong"] & X \times Y \arrow[r, "\cong"] \arrow[d, "\eta"'] & Y \times X \arrow[d, "\eta"] \\
I \times X \arrow[r, "\eta"] & I \otimes X & X \otimes Y \arrow[r, "\cong"] & Y \otimes X
\end{tikzcd}$$

both directions in the first diagram induce the correspondence between maps out of $1 \times X$ strict in the second variable and strict maps out of $I \otimes X$. For the second diagram, both maps induce the correspondence between bistrict maps out of $X \times Y$ and strict maps out of $Y \otimes X$. □

Lemma 3.12. *The lifting functor $L\colon \mathbf{Dom} \to \mathbf{Dom}_\perp$ is a strong symmetric monoidal functor.*

Proof. For all domains A, B, C

$$\begin{aligned} L(A \times B) \multimap C \cong A \times B \to C \cong A \to B \to C \cong \\ LA \multimap LB \multimap C \cong LA \otimes LB \multimap C \end{aligned} \tag{1}$$

so that $LA \otimes LB \cong L(A \times B)$ and by definition $L1 \cong I$. This defines the natural transformation m and map m_I needed for L to be a strong symmetric monoidal functor. Now, of course one will have to show commutativity of the diagrams of [8, Definition 1.1], but this can be done exactly as in the proof of Lemma 3.11. □

Lemma 3.13. *The adjunction*

$$\begin{tikzcd} \mathbf{Dom}_\perp \arrow[r, phantom, "\perp"] & \mathbf{Dom} \arrow[l, bend right, "L"'] \\ \end{tikzcd}$$

$$\begin{tikzcd} \mathbf{Dom}_\perp \arrow[r, bend right, "U"'] \arrow[r, phantom, "\perp"] & \mathbf{Dom} \arrow[l, bend right, "L"'] \end{tikzcd}$$

is symmetric monoidal.

Proof. The functors of the adjunction are symmetric monoidal and the left adjoint is strong, so the lemma follows from [8, Theorem 1.4]. □

Lemma 3.14. *The functor $L\colon \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$ extends the SMCC structure on the category of domains to a linear category structure.*

Proof. This follows from Lemma 3.13 and [8, Proposition 1.14]. □

4 The domains fibration

In this section we construct a PILL$_Y$-model based on the linear structure of the category $\mathbf{Dom}_\perp$. A first attempt at such a model would model types with n free variables as maps $f\colon (\mathbf{Dom}_\perp)_0^n \to (\mathbf{Dom}_\perp)_0$ where $(\mathbf{Dom}_\perp)_0$ is the class of domains. But to be able to handle polymorphism, we change this model slightly, such that types become functors $f\colon (\mathbf{Dom}_\perp)_{\mathtt{iso}}^n \to \mathbf{Dom}_\perp$ where $(\mathbf{Dom}_\perp)_{\mathtt{iso}}$ is the restriction of $\mathbf{Dom}_\perp$ to isomorphisms. The idea here is basically that if $f\colon (\mathbf{Dom}_\perp)_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ is a type with one free variable, then the values of f are up to isomorphism determined by the values of f on the domains in $\mathbf{D}$, so we can define the product of all the $f(A)$'s with A ranging over all domains (i.e., we take the product over a proper class) as

$$\textstyle\prod_{d\in\mathbf{D}} f(d)$$

with projection onto $f(A)$ defined as

$$\textstyle\prod_{d\in\mathbf{D}} f(d) \xrightarrow{\pi_d} f(d) \xrightarrow{f(i)} f(A)$$

defined by taking $i\colon d \cong A$. The idea described here will be modified slightly to make the projection described above independent of the choice of d, i. The details are described in Lemma 4.2, and the idea of modelling polymorphism this way is due to [14].

The model described in this section will be modified to a parametric PILL$_Y$-model in Section 5.

Some of the constructions of the present section cannot be carried out in the set theoretic setting used in Section 2, since they involve constructions on classes. In particular, since $(\mathbf{Dom}_\perp)_0$ is a class and not a set, the collection of all class maps $(\mathbf{Dom}_\perp)_0 \to (\mathbf{Dom}_\perp)_0$ is not a class, and so since a category has a class of objects, we cannot use this collection to construct a category.

For the concerned reader, we sketch how these issues may be resolved. As the given model of SDT, we will assume that we have a category of classes satisfying the axioms of Joyal and Moerdijk's algebraic set theory [5] as refined in [17] with the notion of classic structure on a regular category with a universe and a small natural numbers object. Given such a setting, the categories $\mathbf{Dom}$ and $\mathbf{Dom}_\perp$ mentioned above are internal categories in the regular category of classes while the collection of all internal functors $(\mathbf{Dom}_\perp)_0 \to (\mathbf{Dom}_\perp)_0$ is a class in the external sense, since it is a subclass of the class of morphisms of the category of classes. Thus the fibrations in Lemma 4.7 are defined externally. The examples of realizability toposes mentioned in Section 2 still provide models as they embed into categories of classes as described in [18].

The reader should keep in mind that we really construct a family of parametric LAPL-structures. Since the LAPL-structure is constructed using SDT, we get a parametric LAPL-structure for each model of SDT.

We now begin the detailed description of the model. Consider the category $(\mathbf{Dom}_\perp)_{\mathtt{iso}}$ obtained from $\mathbf{Dom}_\perp$ by restricting to the isomorphisms. We will define the fibration

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$$

by defining the base category to have as objects natural numbers and as morphisms from n to m functors $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to (\mathbf{Dom}_\perp)^m_{\mathtt{iso}}$. Objects in $\mathbf{DFam}((\mathbf{Dom}_\perp)_{\mathtt{iso}})$ over n are functors $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ and morphisms are natural transformations. Reindexing is by composition.

Lemma 4.1. *The fibration*

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$$

has a fibred linear structure plus fibred products.

Proof. Suppose $f, g\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ are objects of $\mathbf{DFam}((\mathbf{Dom}_\perp)_{\mathtt{iso}})_n$, we define $f \otimes g$ by composing the pairing $\langle f, g\rangle$ with the functor $\otimes\colon \mathbf{Dom}_\perp \times \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$. Products are likewise defined pointwise, and the comonad is given by pointwise application of L. We define $(f \multimap g)(\vec{D}) = f(\vec{D}) \multimap g(\vec{D})$ and if $\vec{i}\colon \vec{D} \multimap \vec{D}'$ is a vector of isomorphisms, then $(f \multimap g)(\vec{i})(h\colon f(\vec{D}) \multimap g(\vec{D})) = g(\vec{i}) \circ h \circ f(\vec{i}^{-1})$.

Finally, we notice that the equations required for this to define a fibred linear structure hold, since they hold pointwise. □

Lemma 4.2. *There exists right Kan extensions for all functors $(\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ along projections $(\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to (\mathbf{Dom}_\perp)^n_{\mathtt{iso}}$.*

Proof. Suppose $g\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Dom}_\perp$. We define $\mathrm{RK}_\pi(g)\colon \mathbf{Dom}^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ as

$$\mathrm{RK}_\pi(g)(\vec{A}) = \{x \in \textstyle\prod_{D\in\mathbf{D}} g(\vec{A}, D) \mid \forall D, D' \in \mathbf{D}, i\colon D \multimap D' \text{ iso.}\, g(\vec{A}, i)x_D = x_{D'}\}$$

This is a domain since it is the limit of a diagram of domains, and the category of domains with strict maps is complete with limits as computed in sets.

The required adjoint correspondence is given as follows. Suppose $f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ and $t\colon \pi^* f \Rightarrow g$. The $\vec{A}$ component of the natural transformation $\bar{t}\colon f \Rightarrow \mathrm{RK}_\pi(g)$ is given by the family $(t_{\vec{A},D})_{D\in\mathbf{D}}$. We need to show that this map has image in the subset $\mathrm{RK}_\pi(g)(\vec{A})$, but this follows from the naturality diagram for t:

$$\begin{tikzcd}
(\pi^* f)(\vec{A}, D) \arrow[r, -o, "t_{\vec{A},D}"] \arrow[d, -o, "(\pi^* f)(id,i)=id"'] & g(\vec{A}, D) \arrow[d, -o, "g(\vec{A},i)"] \\
(\pi^* f)(\vec{A}, D') \arrow[r, -o, "t_{\vec{A},D'}"] & g(\vec{A}, D')
\end{tikzcd}$$

which commutes for each isomorphism $i\colon D \multimap D'$.

Suppose on the other hand that $t\colon f \Rightarrow \mathrm{RK}_\pi(g)$. Given any domain B, there exists $i\colon D \multimap B$ isomorphism, and we define $t_{\vec{A},B}\colon f(\vec{A}) \multimap g(\vec{A}, B)$ as the composition

$$\begin{tikzcd}
f(\vec{A}) \arrow[r, -o, "t_{\vec{A}}"] & \mathrm{RK}_\pi(g)(\vec{A}) \arrow[r, -o, "\pi_D"] & g(\vec{A}, D) \arrow[r, -o, "g(\vec{A},i)"] & g(\vec{A}, B)
\end{tikzcd}$$

where π_D is the projection onto the D'th coordinate. We show that this definition is independent of the choice of D, i. So suppose D', i' is another such choice, then we have a commutative diagram

$$\begin{tikzcd}
\mathrm{RK}_\pi(g)(\vec{A}) \arrow[r, -o, "\pi_D"] \arrow[dr, -o, "\pi_{D'}"'] & g(\vec{A}, D) \arrow[r, -o, "g(\vec{A},i)"] \arrow[d, -o, "g(\vec{A},(i')^{-1}\circ i)"] & g(\vec{A}, B) \\
 & g(\vec{A}, D') \arrow[ur, -o, "g(\vec{A},i')"'] &
\end{tikzcd}$$

where the first triangle commutes by definition of $\mathrm{RK}_\pi(g)$ and the second triangle commutes by g being a functor.

One may easily check that these two maps define a bijective correspondence between transformations $\pi^* f \Rightarrow g$ and transformations $f \Rightarrow \mathrm{RK}_\pi(g)$. It is clear that the correspondence is natural. □

Lemma 4.3. *The fibration*

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$$

has a generic object and simple products.

Proof. The generic object is simply the inclusion $(\mathbf{Dom}_\perp)_{\mathtt{iso}} \to \mathbf{Dom}_\perp$. This is a split generic object since all functors factorize through it.

Suppose $g\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Dom}_\perp$. We define the product $\prod g\colon \mathbf{Dom}^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ to be the $\mathrm{RK}_\pi(g)$. The universal property of Kan extensions then gives us the desired correspondence between maps

$$\begin{array}{c} \pi^* f \to g \\ \hline\hline f \to \prod g \end{array}$$

□

Remark 4.4. *From the proof of 4.2 we can extract the interpretation of type specialization. Suppose $x \in \prod g(\vec{A})$ and B is any domain. To specialize x to B, we choose $D \in \mathbf{D}$ and $i\colon D \multimap B$ and define*

$$x(B) = g(\vec{A}, i)(x_D)$$

where x_D is the D'th component of x. As we have proved, this definition is independent of the choice of D, i.

Consider the fibration $\mathbf{DFam}(\mathbf{Dom}) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$ defined to have as objects in the fiber over n functors $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}$ and as vertical maps natural transformations.

Lemma 4.5. *The fibration $\mathbf{DFam}(\mathbf{Dom}) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$ is equivalent to the fibration of finite products of free coalgebras for the comonad $!$ on $\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$. The maps of the equivalence together with the identity on $\mathbf{DFam}(\mathbf{Dom}_\perp)$ form a map of fibred adjunctions.*

Proof. The fibration $\mathbf{DFam}(\mathbf{Dom}) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$ is the coKleisli fibration corresponding to the fibred comonad on $\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$. Now apply Proposition 1.21 of [8]. □

Lemma 4.6. *The model*

$$\begin{tikzcd}
\mathbf{DFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{DFam}(\mathbf{Dom}) \arrow[ll, bend right=10] \arrow[dl] \\
& \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\} &
\end{tikzcd}$$

models Y.

Proof. We define $Y = (\mathit{fix}_D)_{D \in \mathbf{D}}$. Strictly speaking, this $(\mathit{fix}_D)_{D \in \mathbf{D}}$ is an element of the wrong set, since

$$(\mathit{fix}_D)_{D \in \mathbf{D}} \in \textstyle\prod_{D \in \mathbf{D}} (D \to D) \to D$$

and we need an element in the set $\prod_{D \in \mathbf{D}} L(LD \multimap D) \multimap D$. But these sets are isomorphic, and in the following we work with implicit isomorphisms between them. We need to check that $(\mathit{fix}_D)_{D \in \mathbf{D}}$ in fact defines an element in the type $[\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]$, i.e., the right Kan extension of the functor $D \mapsto [(D \to D) \to D]$. So we need to check that for all $i \colon D \multimap D'$ isomorphisms between elements $D, D' \in \mathbf{D}$

$$((i \to i) \to i)(\mathit{fix}_D) = \mathit{fix}_{D'}$$

But $((i \to i) \to i)(\mathit{fix}_D)$ is the map that maps a function $f \colon D' \to D'$ to $i(\mathit{fix}_D(i^{-1} \circ f \circ i))$ and since the diagram

$$\begin{tikzcd}
D \arrow[r, "i^{-1} \circ f \circ i"] \arrow[d, "i", multimap] & D \arrow[d, "i", multimap] \\
D' \arrow[r, "f"] & D'
\end{tikzcd}$$

commutes, uniformity of *fix* implies that for all $f \colon D' \to D'$

$$i(\mathit{fix}_D(i^{-1} \circ f \circ i)) = \mathit{fix}_{D'}(f).$$

We have proved that Y in fact defines an element of $[\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]$.

We need to check that $f\ !(Y\ A\ !f) = Y\ A\ !f$ for all domains A and all maps $f \colon A \to A$. As explained in Remark 4.4, the term $Y\ A$ is modeled by choosing an isomorphism $i \colon D \to A$ for some domain $D \in \mathbf{D}$ and setting $[\![Y\ A]\!] = ((i \to i) \to i)\mathit{fix}_D$, which as we saw before, by uniformity, simply is fix_A. Now, to interpret $[\![Y\ A\ (!f)]\!] = \mathit{fix}_A(f)$ we should strictly speaking apply the element of $L(LA \multimap A) \multimap A$ corresponding to fix_A to $\{\bar{f}\}$ where $\bar{f} \colon LA \multimap A$ is the strict map corresponding to $f \colon A \to A$, but this just gives $\mathit{fix}_A(f)$ as one would expect. Likewise $[\![f\ !(Y\ A\ (!f))]\!] = f(\mathit{fix}_A\ f)$, which is equal to $\mathit{fix}_A(f)$. □

We sum the above to the following

Proposition 4.7.

$$\begin{tikzcd}
\mathbf{DFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=10] \arrow[dr] & \perp & \mathbf{DFam}(\mathbf{Dom}) \arrow[ll, bend right=10] \arrow[dl] \\
& \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\} &
\end{tikzcd}$$

is a PILL$_Y$-model.

5 The parametric fibration

In this section, we basically apply a parametric completion process as in [12, 2] to the model of the last section. Types in the resulting model will be types in the old model with a relational interpretation mapping identity relations to identity relations, i.e., satisfying the identity extension schema. First we discuss two notions of relations.

By a relation R between domains A, B we mean a subset of $A \times B$ and we write $\mathbf{Rel}(A, B)$ for the set of relations from A to B. By an admissible relation between domains A, B we mean a subdomain of $A \times B$ and we write $\mathbf{AdmRel}(A, B)$ for the set of admissible relations from A to B. This is also the notion of admissible relations used in [14]. We shall often write $R(x, y)$ for $(x, y) \in R$.

Lemma 5.1. *Admissible relations are closed under reindexing by strict maps and arbitrary intersections, i.e., if $R\colon \mathbf{AdmRel}(A, B)$ and $f\colon A' \multimap A, g\colon B' \multimap B$ are strict maps between domains then*

$$\{(x, y)\colon A' \times B' \mid R(f(x), g(y))\}$$

is an admissible relation, and if $(R_x\colon \mathbf{AdmRel}(A, B))_{x\in X}$ is a set-indexed family of admissible relations, then

$$\{(y, z)\colon A \times B \mid \forall x\colon X.\, R_x(y, z)\}$$

is admissible.

Proof. Reindexing is given by pullbacks

$$\begin{tikzcd}
\{(x, y)\colon A' \times B' \mid R(f(x), g(y))\} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & R \arrow[d] \\
A' \times B' \arrow[r, "f\times g"] & A \times B
\end{tikzcd}$$

and intersections are limits, so the lemma follows from $\mathbf{Dom}_\perp$ being complete. □

Consider the category $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ whose objects are admissible relations on domains, and whose morphisms are pairs of strict maps preserving relations, i.e., mapping related elements to related elements. We denote by $\mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\text{iso}}$ the restriction of $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ to isomorphisms, i.e., morphisms in this category are pairs of isomorphisms (f, g) such that (f, g) as well as (f^{-1}, g^{-1}) preserve relations.

We have canonical reflexive graphs of functors:

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\text{iso}} \arrow[r, shift left=2] \arrow[r, shift right=2] & (\mathbf{Dom}_\perp)_{\mathtt{iso}} \arrow[l] & \mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{Dom}_\perp \arrow[l]
\end{tikzcd}$$

where in both graphs, the functors from left to right map relations to domain and codomain respectively and the functor going from right to left map a domain to the identity relation on the domain.

Lemma 5.2. *The category* $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ *has an SMCC-structure and products. The maps of the reflexive graph*

$$\begin{tikzcd} \mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{Dom}_\perp \arrow[l] \end{tikzcd}$$

commute with the products and the SMCC-structure.

Proof. For $R\colon \mathbf{AdmRel}(A,B)$, $S\colon \mathbf{AdmRel}(C,D)$ we define

$$\begin{gathered} R\times S\colon \mathbf{AdmRel}(A\times C, B\times D)\\ R\multimap S\colon \mathbf{AdmRel}(A\multimap C, B\multimap D) \end{gathered}$$

as

$$\{((x,y),(w,z))\colon (A\times C)\times(B\times D)\mid R(x,w)\wedge S(y,z)\}$$

and

$$\{(f,g)\colon (A\multimap C)\times(B\multimap D)\mid \forall x\colon A, y\colon B.\, R(x,y)\supset S(f(x),g(y))\}.$$

The relation $R\times S$ is easily seen to be admissible from Lemma 5.1. For each x, y

$$\begin{gathered} \{(f,g)\colon (A\multimap C)\times(B\multimap D)\mid R(x,y)\supset S(f(x),g(y))\} = \\ \textstyle\bigcap_{(x',y')\in R\cap\{(x,y)\}}\{(f,g)\colon (A\multimap C)\times(B\multimap D)\mid S(f(x'),g(y'))\} \end{gathered}$$

where the intersection is taken inside $(A\multimap C)\times(B\multimap D)$. And so $R\multimap S$ can be written as the intersection

$$\bigcap_{(x,y)\in A\times B}\ \bigcap_{(x',y')\in R\cap\{(x,y)\}}\{(f,g)\colon (A\multimap C)\times(B\multimap D)\mid S(f(x'),g(y'))\}$$

of admissible relations, and so is admissible by Lemma 5.1.

An admissible relation can be considered as a jointly monic span in the usual sense. For the definition of the tensor on relations, we will change notation a bit. We write $\bar{R}$ for the codomain of the maps of the span in the following, in order not to confuse this with the relation. The point is that the domain of the relation $R\otimes S$ will not necessarily be $\bar{R}\otimes\bar{S}$ as in the span

$$\begin{tikzcd} & \bar{R}\otimes\bar{S} \arrow[dl] \arrow[dr] & \\ A\otimes C & & B\otimes D, \end{tikzcd}$$

obtained by tensoring the two spans

$$\begin{tikzcd} & \bar{R} \arrow[dl] \arrow[dr] & & & \bar{S} \arrow[dl] \arrow[dr] & \\ A & & B & C & & D \end{tikzcd}$$

since we do not know that this is a jointly monic span. In stead we define $R\otimes S$ to be the intersection of all subdomains of $(A\otimes C)\times(B\otimes D)$ containing the image of this span. Now, for $T\colon \mathbf{AdmRel}(E,F)$ and $t\colon A\otimes C\multimap E, s\colon B\otimes D\multimap F$ the pair (t,s) preserves relations iff there exists a map r as making

$$\begin{tikzcd} & \bar{R}\otimes\bar{S} \arrow[rrr, "r"] \arrow[dl] \arrow[dr] & & & T \arrow[dl] \arrow[dr] & \\ A\otimes C \arrow[rrr, bend right, "t"'] & & B\otimes D \arrow[rrr, bend right, "s"'] & E & & F \end{tikzcd}$$

commute, because if the map r exists, then the pullback of T along $t \times s$ is a subdomain of $(A \otimes C) \times (B \otimes D)$ containing the image of the $\otimes$-span. On the other hand, if (t, s) preserve relations, then the map r can be defined by composition with $t \times s$.

Now, by naturality of η, the map r exists iff there exists a map u making

$$\begin{tikzcd}
 & \bar{R} \times \bar{S} \arrow[rrr, "u"] \arrow[dl] \arrow[dr] & & & T \arrow[dl] \arrow[dr] & \\
A \times C \arrow[rrr, bend right, "\hat{s}"'] & & B \times D \arrow[rrr, bend right, "\hat{t}"'] & E & & F
\end{tikzcd}$$

commute, where $\hat{t}, \hat{s}$ are the bistrict maps corresponding to s, t. So $(s, t)\colon R \otimes S \multimap T$ correspond bijectively to bistrict pairs $(\hat{s}, \hat{t})\colon R \times S \multimap T$, and these pairs correspond bijectively to maps from R to $S \multimap T$ showing that $(-) \otimes S$ is left adjoint to $S \multimap (-)$.

The neutral element for $\otimes$ is the identity relation on $I \in \mathbf{Dom}_\perp$. Maps $R \otimes eq_I \multimap S$ correspond to bistrict maps $R \times eq_I \multimap S$, which correspond to strict maps $R \multimap S$ so that $R \cong R \otimes I$.

The structure maps of the SMCC-structure on $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ such as the natural transformation

$$(-) \otimes ((=) \otimes (\equiv)) \multimap ((-) \otimes (=)) \otimes (\equiv)$$

are just given by pairing the corresponding maps in $\mathbf{Dom}_\perp$. Of course, one has to show that these maps preserve relations, but that is easy. Clearly the SMCC-structures om $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ and $\mathbf{Dom}_\perp$ commute with the domain and codomain maps. For the equality map, the only difficult thing to show is that $eq_A \otimes eq_B = eq_{A \otimes B}$.

Suppose R is any admissible relation between any pair of domains. Since R is itself simply a domain, we have the following equivalences

$$\begin{gathered}
\mathrm{Hom}_{\mathbf{AdmRel}(\mathbf{Dom}_\perp)}(eq_{A \otimes B}, R) \cong \mathrm{Hom}_{\mathbf{Dom}_\perp}(A \otimes B, R) \cong \\
\mathrm{Hom}_{\mathbf{Dom}_\perp}(A, B \multimap R) \cong \mathrm{Hom}_{\mathbf{AdmRel}(\mathbf{Dom}_\perp)}(eq_A, eq_B \multimap R) \cong \\
\mathrm{Hom}_{\mathbf{AdmRel}(\mathbf{Dom}_\perp)}(eq_A \otimes eq_B, R).
\end{gathered}$$

An easy check shows that this correspondence is given by the identity on the underlying pairs of maps, so by the Yoneda Lemma $eq_{A \otimes B}$ is isomorphic to $eq_A \otimes eq_B$ with isomorphism given by the pair $(id_{A \otimes B}, id_{A \otimes B})$. □

Lemma 5.3. *The category* $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ *has a linear category structure, commuting with the functors of*

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{Dom}_\perp \arrow[l] .
\end{tikzcd}$$

Proof. Suppose $R\colon \mathbf{AdmRel}(A, B)$. The relation can be considered as a jointly monic span

$$\begin{tikzcd}
 & R \arrow[dl] \arrow[dr] & \\
A & & B
\end{tikzcd}$$

in $\mathbf{Dom}_\perp$. We define the lifting of R to be the relation obtained by applying the functor ! to each map in the span. It is an easy exercise to show that the resulting span is jointly monic.

We need to check that this defines a comonad, and it suffices to check that the maps of the comonad on $\mathbf{Dom}_\perp$ preserve relations, which follows from naturality as in the diagram for ϵ:

$$\begin{tikzcd}[column sep=small]
& {!R} \arrow[rrr, "\epsilon"] \arrow[dl] \arrow[dr] & & & R \arrow[dl] \arrow[dr] & \\
{!A} \arrow[rrr, bend right, "\epsilon"'] & & {!B} \arrow[rrr, bend right, "\epsilon"'] & A & & B.
\end{tikzcd}$$

The same reasoning applies for the rest of the linear structure. For example, since d is the composition of $!\Delta$ with the isomorphism $!((-) \times (=)) \cong !(-) \otimes !(=)$, we see that d preserves relations from the following diagram

$$\begin{tikzcd}[column sep=small]
& {!R} \arrow[rrr, "!\Delta"] \arrow[dl] \arrow[dr] & & & {!(R \times R)} \arrow[rrr, "\cong"] \arrow[dl] \arrow[dr] & & & {!R \otimes !R} \arrow[dl] \arrow[dr] & \\
{!A} \arrow[rrr, bend right, "!\Delta"'] & & {!B} \arrow[rrr, bend right, "!\Delta"'] & {!(A \times A)} \arrow[rrr, bend right, "\cong"'] & & {!(B \times B)} \arrow[rrr, bend right, "\cong"'] & {!A \otimes !A} & & {!B \otimes !B.}
\end{tikzcd}$$

The span on the right actually represents the relation $!R \otimes !R$, because it is jointly monic (it is isomorphic to the span in the middle).

The proofs that δ, e, m, m_I preserve relations is done likewise. The commutative diagrams of [8, Definition 1.10, Lemma 1.11] commute since they commute in $\mathbf{Dom}_\perp$. □

We define the category $\mathbf{PDom}$ to have as objects natural numbers, and as morphisms from n to m pairs of functors making the diagram

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp)^n_{\text{iso}} \arrow[r] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{AdmRel}(\mathbf{Dom}_\perp)^m_{\text{iso}} \arrow[d, shift right=2] \arrow[d, shift left=2] \\
(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \arrow[r] \arrow[u] & (\mathbf{Dom}_\perp)^m_{\mathtt{iso}} \arrow[u]
\end{tikzcd}$$

commute.

We define the category $\mathbf{PFam}(\mathbf{Dom}_\perp)$ fibred over $\mathbf{PDom}$ to have as objects over n pairs of functors making the diagram

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp)^n_{\text{iso}} \arrow[r, "f^r"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[d, shift right=2] \arrow[d, shift left=2] \\
(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \arrow[r, "f^d"] \arrow[u] & \mathbf{Dom}_\perp \arrow[u]
\end{tikzcd}$$

commute. A vertical morphisms from (f^r, f^d) to (g^r, g^d) is a a pair of natural transformations $(s \colon f^r \Rightarrow g^r, t \colon f^d \Rightarrow g^d)$ making the obvious diagrams commute, i.e., for all $\vec{R} \colon \mathbf{AdmRel}(\vec{\alpha}, \vec{\beta})$,

$$\begin{array}{rcl}
\mathit{dom}(s_{\vec{R}}) & = & t_{\vec{\alpha}} \\
\mathit{codom}(s_{\vec{R}}) & = & t_{\vec{\beta}} \\
s_{eq_{\vec{\alpha}}} & = & (t_{\vec{\alpha}}, t_{\vec{\alpha}})
\end{array}$$

where $\mathit{dom}, \mathit{codom}$ denote the domain and codomain maps respectively. Since maps in $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ are given by pairs of maps in $\mathbf{Dom}_\perp$, clearly the equations determine s from t, so an alternative description of vertical morphisms would be natural transformations $t\colon f^d \Rightarrow g^d$ such that for all vectors of relations $\vec{R}\colon \mathbf{AdmRel}(\vec{\alpha},\vec{\beta})$, $(t_{\vec{\alpha}}, t_{\vec{\beta}})$ is a map of relations $f^r(\vec{R}) \to g^r(\vec{R})$.

Reindexing in the fibration $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ is by composition.

Lemma 5.4. *The fibration* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ *has a fibred linear structure and fibred products.*

Proof. The structure is defined pointwise, using Lemma 5.3, i.e., for example for $f = (f^r, f^d), g = (g^r, g^d)$ objects over n, we define

$$\begin{array}{rcl}(f \otimes g)^r(\vec{R}) & = & f^r(\vec{R}) \otimes g^r(\vec{R}) \\ (f \otimes g)^d(\vec{A}) & = & f^d(\vec{A}) \otimes g^d(\vec{A}).\end{array}$$

Of course, as in the proof of Lemma 4.1 since $(-) \multimap (=)$ is contravariant in the first variable, to define $f \multimap g$ for covariant functors f, g as a covariant functor, we must use that the domain of the functors f, g is a category in which all arrows are invertible, so that we can define $(f \multimap g)^d(i) = f^d(i^{-1}) \multimap g^d(i)$ and likewise for $(f \multimap g)^r$.

The needed natural transformations are defined using the corresponding natural transformations in $\mathbf{Dom}_\perp$ and $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$. For example ϵ is defined as $(\epsilon\colon {!f^r} \multimap f^r, \epsilon\colon {!f^d} \multimap f^d)$, and the equations needed hold, since they hold in $\mathbf{AdmRel}(\mathbf{Dom}_\perp)$ and $\mathbf{Dom}_\perp$. Since the requirement of $\multimap$ and $\otimes$ being adjoint can be expressed 2-categorically, the same argument can be used to show this. □

Lemma 5.5. *The fibration* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ *has a generic object and simple products.*

Proof. The generic object is the inclusion

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\text{iso}} \arrow[r] \arrow[d, shift left=2] \arrow[d, shift right=2] & \mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[d, shift left=2] \arrow[d, shift right=2] \\
(\mathbf{Dom}_\perp)_{\mathtt{iso}} \arrow[r] \arrow[u] & \mathbf{Dom}_\perp \arrow[u]
\end{tikzcd}$$

For the simple products, we define for $f^d\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ the product $(\prod f)^d\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ by defining $(\prod f)^d(\vec{A})$ to be

$$\{x \in \textstyle\prod_{D \in \mathbf{D}} f^d(\vec{A}, D) \mid \forall D, D' \in \mathbf{D}.\, \forall R \in \mathbf{AdmRel}(D, D').\, f^r(eq_{\vec{A}}, R)(x_D, x_{D'})\}$$

where we write x_D for $\pi_D(x)$. We define the relational interpretation as

$$(\textstyle\prod f)^r(\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B}))(x, y)$$

for $x \in (\prod f)^d(\vec{A}), y \in (\prod f)^d(\vec{B})$ iff

$$\forall D, D' \in \mathbf{D}.\, \forall R' \in \mathbf{AdmRel}(D, D') f^r(\vec{R}, R')(x_D, y_{D'}).$$

Since this is an intersection of admissible relations it is admissible by Lemma 5.1.

We show that $\prod f^r(eq_{\vec{A}}) = eq_{f^d(\vec{A})}$, proving that $(\prod f^r, \prod f^d)$ actually defines an object of $\mathbf{PFam}(\mathbf{Dom}_\perp)$. Suppose first that $(x, y) \in \prod f^r(eq_{\vec{A}})$. By definition $(x_D, y_D) \in f^r(eq_{\vec{A}}, eq_D) = eq_{f^d(\vec{A}, D)}$,i.e., $x_D = y_D$ and so we have proved $\prod f^r(eq_{\vec{A}}) \subset eq_{f^d(\vec{A})}$. Suppose on the other hand $x \in \prod f^d(\vec{A})$. We must prove that $(x, x) \in \prod f^r(eq_{\vec{A}})$, i.e. that for all $D, D' \in \mathbf{D}, R \in \mathbf{AdmRel}(D, D')$ we have

$$(x_D, x_{D'}) \in f^r(eq_{\vec{A}}, R)$$

which is exactly the definition of $x \in \prod f^d(\vec{A})$.

We will define the bijective correspondence between maps $(\pi^*g)^d \to f^d$ and maps $g^d \to (\prod f)^d$ basically as in the proof of Lemma 4.3. We need to show that in this correspondence maps preserving relations correspond to maps preserving relations.

If $t\colon (\pi^*g)^d \to f^d$ such that $(t,t)\colon (\pi^*g)^r \multimap f^r$ we define $\hat{t}\colon g^d \to (\prod f)^d$ as $\hat{t}_{\vec{A}}(x) = (t_{\vec{A},D}(x))_{D \in \mathbf{D}}$. We show that this defines an element in $(\prod f)^d(\vec{A})$. Suppose $D, D' \in \mathbf{D}, R\colon \mathbf{AdmRel}(D, D')$. Since $x \in g^d(\vec{A})$, and $(x,x) \in (\pi^*g)^r(eq_{\vec{A}}, R) = eq_{g^d(\vec{A})}$, the fact that t preserves relations show that

$$(t_{\vec{A},D}(x), t_{\vec{A},D'}(x)) \in f^r(eq_{\vec{A}}, R)$$

as desired. It is clear that if t preserves relations, so does $\hat{t}$.

Suppose $u\colon g^d \to (\prod f)^d$. We show that $\hat{u}\colon \pi^*g^d \to f^d$ defined as in the proof of Lemma 4.2 also preserves relations. So suppose we have admissible relations $\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})$ and $R\colon \mathbf{AdmRel}(A, B)$ and that $g^r(\vec{R})(x,y)$. Pick $D, D' \in \mathbf{D}$ and isomorphisms $i\colon D \multimap A, i'\colon D' \multimap B$, then by definition

$$\hat{u}_{\vec{A},A}(x) = f^d(id_{\vec{A}}, i) \circ \pi_D \circ u_{\vec{A}}(x) \qquad \hat{u}_{\vec{B},B}(y) = f^d(id_{\vec{B}}, i') \circ \pi_{D'} \circ u_{\vec{B}}(y). \tag{2}$$

Since $(i,i')^*R' \in \mathbf{AdmRel}(D, D')$, and since u preserves relations, we must have

$$(\pi_D \circ u_{\vec{A}}(x), \pi_{D'} \circ u_{\vec{B}}(y)) \in f^r(\vec{R}, (i,i')^*R') \tag{3}$$

by definition of $(\prod f)^r(\vec{R})$. Since $(i,i')\colon (i,i')^*R \multimap R$ preserve relations and f^r is a functor,

$$(f^d(id_{\vec{A}}, i), f^d(id_{\vec{B}}, i'))\colon f^r(\vec{R}, (i,i')^*R) \multimap f^r(\vec{R}, R)$$

preserve relations, which together with (2) and (3) means that

$$(\hat{u}_{\vec{A},A}(x), \hat{u}_{\vec{B},B}(y)) \in f^r(\vec{R}, R)$$

as desired. □

We define the category $\mathbf{PFam}(\mathbf{Dom})$ fibred over $\mathbf{PDom}$ to have the same objects as $\mathbf{PFam}(\mathbf{Dom}_\perp)$. A vertical morphisms from (f^r, f^d) to (g^r, g^d) is a natural transformation $t\colon f^d \Rightarrow g^d$ whose components are not required to be strict as they are in $\mathbf{PFam}(\mathbf{Dom}_\perp)$, but still required to preserve relations, i.e., if $\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})$, then the pair $(t_{\vec{A}}, t_{\vec{B}})$ is a map of relations $f^r(\vec{R}) \to g^r(\vec{R})$. Reindexing in the fibration $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{PDom}$ is given by composition.

Lemma 5.6. *The fibration* $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{PDom}$ *is equivalent to the fibration of finite products of free coalgebras for the fibred comonad* $!$ *on* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$*. The maps of the equivalence together with the identity on* $\mathbf{PFam}(\mathbf{Dom}_\perp)$ *form a map of fibred adjunctions.*

Proof. It is easy to see that $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{PDom}$ is the fibred co-Kleisli category for $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$, since maps preserving relations out of

$$\begin{tikzcd}
 & R \arrow[dl] \arrow[dr] & \\
A & & B
\end{tikzcd}$$

correspond to strict maps preserving relations out of

$$\begin{tikzcd}
 & !R \arrow[dl] \arrow[dr] & \\
!A & & !B
\end{tikzcd}$$

Since $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ has fibred products we may appeal to [8, Proposition 1.21]. □

Lemma 5.7. *The model*

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{PFam}(\mathbf{Dom}) \arrow[ll, bend right=10] \arrow[dl] \\
 & \mathbf{PDom} &
\end{tikzcd}$$

models Y

Proof. We have a Y-combinator in the fibration

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{((\mathbf{Dom}_\perp)_{\mathtt{iso}})^n \mid n\}$$

given by the family $(\mathit{fix}_D)_{D \in \mathbf{D}}$. We show that this element defines a term in $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$, for which we basically need to show that $(\mathit{fix}_D)_{D \in \mathbf{D}}$ is in the relational interpretation of the type $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$.

So we need to show that

$$(\mathit{fix}_D)_{D \in \mathbf{D}} (\textstyle\prod \alpha.\, (\alpha \to \alpha) \to \alpha) (\mathit{fix}_D)_{D \in \mathbf{D}},$$

i.e., that

$$\begin{gathered} \forall D, D' \in \mathbf{D}.\, \forall R \colon \mathbf{AdmRel}(D, D').\, \forall f \colon D \to D, g \colon D' \to D'. \\ (R \to R)(f, g) \supset R(\mathit{fix}_D f, \mathit{fix}_{D'} g). \end{gathered}$$

So suppose we are given $D, D' \in \mathbf{D}$. An admissible relation from D to D' is given by an inclusion of a subdomain

$$R \multimap D \times D'$$

and so $(R \to R)(f, g)$ means that the restriction of $f \times g$ to R factors through R, i.e., we have a commutative diagram

$$\begin{tikzcd}
R \arrow[r, "(f \times g)|_R"] \arrow[d, -o] & R \arrow[d, -o] \\
D \times D' \arrow[r, "f \times g"] & D \times D'.
\end{tikzcd}$$

From uniformity of fixed points we deduce that $\mathit{fix}_{D \times D'}(f \times g) = \mathit{fix}_R(f \times g)|_R$ and therefore $\mathit{fix}_{D \times D'}(f \times g) \in R$. But using naturality on the commutative square

$$\begin{tikzcd}
D \times D' \arrow[r, "f \times g"] \arrow[d, -o] & D \times D' \arrow[d, -o] \\
D \arrow[r, "f"] & D
\end{tikzcd}$$

(and likewise for the other projection) we see that

$$(\mathit{fix}_D f, \mathit{fix}_{D'} g) = \mathit{fix}_{D \times D'}(f \times g)$$

and so $(\mathit{fix}_D f, \mathit{fix}_{D'} g) \in R$.

Proving that $(\mathit{fix}_D)_{D \in \mathbf{D}}$ satisfies the required equations is done as in the proof of Lemma 4.6. □

Proposition 5.8.

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=8] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{PFam}(\mathbf{Dom}) \arrow[ll, bend right=8] \arrow[dl] \\
& \mathbf{PDom} &
\end{tikzcd} \tag{4}$$

is a PILL$_Y$-model.

Proof. This is the collected statement of the above lemmas. □

6 The LAPL-structure

In this section we show that the PILL$_Y$-model (4) is parametric by constructing a parametric LAPL-structure around it. Even though types in this model are pairs (f^r, f^d), when reasoning about parametricity, we will just consider the f^d part of a type. We can consider f^r as a relational interpretation of the type (f^r, f^d) since for each vector of relations $\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})$ we have $f^r(\vec{R})\colon \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B}))$. Notice also, that since terms from (f^r, f^d) to (g^r, g^d) are natural transformations $t\colon f^d \Rightarrow g^d$, so forgetting the f^r-part of a type represents a faithful functor.

Since the category of contexts should contain all functors $f^d\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}$ and types for all relations between them, a natural choice is to have this category contain all functors $f^d\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$. We will use set theoretic logic to reason about the model, so the category $\mathbf{Prop}$ should contain subfunctors of the functors in $\mathbf{Ctx}$.

The pre-LAPL-structure will be given by the diagram

$$\begin{tikzcd}
& & \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[r, shift right] \arrow[drr] & \mathbf{PFam}(\mathbf{Dom}) \arrow[l, shift right] \arrow[r] \arrow[dr] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
& & \mathbf{PDom}.
\end{tikzcd} \tag{5}$$

The category $\mathbf{DFam}(\mathbf{Set})$ is fibred over $\mathbf{PDom}$. Its fibre over n has as objects functors

$$(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set},$$

and reindexing along a morphism from m to n in $\mathbf{PDom}$ is by composition with the functor

$$((\mathbf{Dom}_\perp)_{\mathtt{iso}})^m \to ((\mathbf{Dom}_\perp)_{\mathtt{iso}})^n.$$

The category $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))$ is a fibred partial order over $\mathbf{DFam}(\mathbf{Set})$ and has as objects over

$$f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$$

subfunctors of f ordered by inclusion. The map $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{DFam}(\mathbf{Set})$ is given by the inclusion of $\mathbf{Dom}$ into $\mathbf{Set}$.

Lemma 6.1. *The fibration* $\mathbf{DFam}(\mathbf{Set}) \to \mathbf{PDom}$ *has fibred products and products in the base.*

Proof. The fibred products are given pointwise. □

Lemma 6.2. *The fibred functor*

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}) \arrow[r] \arrow[dr] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
 & \mathbf{PDom}
\end{tikzcd}$$

given by $(f^r, f^d) \mapsto i \circ f^d$*, where* $i\colon \mathbf{Dom} \to \mathbf{Set}$ *is the inclusion, preserves fibred products and is faithful.*

Lemma 6.3. *The composite fibration* $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{DFam}(\mathbf{Set}) \to \mathbf{PDom}$ *is a fibred first-order logic fibration with products with respect to projections in* $\mathbf{PDom}$*.*

Proof. The fibred first-order logic structure is defined pointwise using the first-order logic structure of $\mathrm{Sub}(\mathbf{Set}) \to \mathbf{Set}$.

We should show that for any projection $\pi\colon n+1 \to n$ in $\mathbf{PDom}$ and any $f \in \mathbf{DFam}(\mathbf{Set})_n$ we have a right adjoint to

$$(\bar{\pi})^*\colon \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))_f \to \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))_{\pi^* f}.$$

To be more precise, suppose $f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ is an object of $\mathbf{DFam}(\mathbf{Set})_n$ and $h\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Set}$ is a subfunctor of $\pi^* f = f \circ \pi$. We must define $(\prod h)\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ a subfunctor of f and prove that for any other subfunctor g of f

$$\forall \vec{A}.\, g(\vec{A}) \subseteq (\textstyle\prod h)(\vec{A}) \quad \text{iff} \quad \forall \vec{A}, B.\, g(\vec{A}) \subseteq h(\vec{A}, B). \tag{6}$$

Moreover, we must prove that $\prod$ is a functor, i.e. if $h' \subseteq h''$ then $\prod h' \subseteq \prod h''$, and that the Beck-Chevalley conditions are satisfied.

Define

$$(\textstyle\prod h)(\vec{A}) = \displaystyle\bigcap_{D \in \mathbf{D}} h(\vec{A}, D).$$

Clearly, the right to left implication of (6) holds. Suppose on the other hand that

$$\forall \vec{A}.\, g(\vec{A}) \subseteq (\textstyle\prod h)(\vec{A}).$$

If $\vec{A}, B$ are domains, we must show that $g(\vec{A}) \subseteq h(\vec{A}, B)$. We know that there exists $D \in \mathbf{D}$ and isomorphism $i\colon B \cong D$. Since $h(\vec{A}, i)\colon h(\vec{A}, B) \to h(\vec{A}, D)$ is an isomorphism of subobjects of $f(\vec{A})$ we must have $h(\vec{A}, B) = h(\vec{A}, D)$, so since clearly $g(\vec{A}) \subseteq h(\vec{A}, D)$, also $g(\vec{A}) \subseteq h(\vec{A}, B)$ as desired.

It is clear that $\prod(-)$ defines a functor, i.e. preserves order of subobjects of f. Concerning the Beck-Chevalley conditions, we must show that $\prod(-)$ commutes with reindexing in $\mathbf{PDom}$, which holds since reindexing commutes with taking intersections of indexed sets. For the other Beck-Chevalley condition suppose we have a pullback diagram in $\mathbf{DFam}(\mathbf{Set})$:

$$\begin{tikzcd}
\pi^* f \arrow[r, "\bar{\pi}"] \arrow[d, "\pi^* t"'] & f \arrow[d, "t"] \\
\pi^* g \arrow[r, "\bar{\pi}"] & g
\end{tikzcd}$$

for $f, g\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ and t vertical, and suppose also we have a subobject $h\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Set}$ of $\pi^* g$. We can then compute

$$(t^*(\textstyle\prod h))_{\vec{A}} = (t^*_{\vec{A}}(\textstyle\bigcap_{D\in\mathbf{D}} h(\vec{A}, D)))_{\vec{A}} = (\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in \textstyle\bigcap_{D\in\mathbf{D}} h(\vec{A}, D)\})_{\vec{A}}$$

and on the other hand

$$(\textstyle\prod((\pi^* t)^*(h)))_{\vec{A}} = (\textstyle\prod(\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in h(\vec{A}, B)\})_{\vec{A},B})_{\vec{A}} = \\ (\textstyle\bigcap_{D\in\mathbf{D}}\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in h(\vec{A}, D)\})_{\vec{A}}$$

Since these two are clearly equal, the Beck-Chevalley condition is satisfied. □

Lemma 6.4. *The diagram (5) is a pre-LAPL-structure.*

Proof. All that is missing in this proof is the definition of the fibred functor U

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \times_{\mathbf{PDom}} \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[r] \arrow[dr] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
& \mathbf{PDom}.
\end{tikzcd}$$

We define

$$U((f^r, f^d), (g^r, g^d))(\vec{A}) = \mathbf{Rel}(f^d(\vec{A}), g^d(\vec{A})).$$

We show that $U((f^r, f^d), (g^r, g^d))$ defines a functor $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ by defining for $\vec{i}\colon \vec{A} \to \vec{A}'$ the action

$$U((f^r, f^d), (g^r, g^d))(\vec{i})\colon U((f^r, f^d), (g^r, g^d))(\vec{A}) \to U((f^r, f^d), (g^r, g^d))(\vec{A}')$$

as $R \in U((f^r, f^d), (g^r, g^d))(\vec{A}) \mapsto (f^d(\vec{i}^{-1}), g^d(\vec{i}^{-1}))^* R$. The map U defines a contravariant fibred functor by reindexing, that is, if $t\colon (f^r, f^d) \to ((f')^r, (f')^d)$ and $t\colon (g^r, g^d) \to ((g')^r, (g')^d)$ are maps, then $U(t, u)$ is defined as

$$R\colon \mathbf{Rel}((f')^d(\vec{A}), (g')^d(\vec{A})) \mapsto (t_{\vec{A}}, u_{\vec{A}})^* R.$$

It is easy to see that U satisfies the requirements. □

Lemma 6.5. *The subfunctor of U given by*

$$V((f^r, f^d), (g^r, g^d))(\vec{A}) = \mathbf{AdmRel}(f^d(\vec{A}), g^d(\vec{A}))$$

defines a notion of admissible relations for the APL-structure (5).

Before we prove Lemma 6.5 we need a few lemmas. Recall that in the LAPL-logic [3], we have defined $x\downarrow$ as the proposition $\exists f\colon X \multimap \Sigma.\, f(x) = \top$, for $x\colon A$ and A a domain.

Lemma 6.6. *The map $\wedge\colon \Sigma \times \Sigma \to \Sigma$ given by $\wedge(e, f) = e \cap f$ is strict.*

Proof. Suppose $E \in (\Sigma \times \Sigma)^p$. Then

$$\wedge \circ r_p^{\Sigma\times\Sigma}(E) = \wedge(\bigcup_{(e,f)\in E} e, \bigcup_{(e,f)\in E} f) = \bigcup_{(e,f)\in E} e \cap f = \mu_p(\{\wedge(e, f) \mid (e, f) \in E\}).$$

□

Lemma 6.7. *Suppose* $e\colon LX$. *The propositions*

- $e\downarrow$
- $\exists x\colon X.\, e = \{x\}$
- $L(!)(e) = \top$

are equivalent, where ! *is the unique map* $X \to 1$.

Proof. Clearly $L(!)(e) = \top$ implies $e\downarrow$. Since

$$L(!)(e) = \{!x \mid x \in e\} = \{\star \mid \exists x \in e\}$$

the second and third proposition are equivalent.

For the last implication suppose $e\downarrow$, i.e., that there exists a map $f\colon LX \multimap \Sigma$ such that $f(e) = \top$. Define the map $g\colon LX \multimap \Sigma$ as $e' \mapsto f(e') \cap \{\emptyset \mid \exists x \in e'\}$. By Lemma 6.6 g is composed of strict maps and so is strict. But since pointed maps out of LX are uniquely determined by their values on singletons $g = f$, and so $f(e) = \top$ implies $\exists x \in e$. □

Proof of Lemma 6.5. For readability, we will assume that everything here takes place in the fiber over $0 \in \mathbf{PDom}$. The more general proof will be the same as below, with all sets replaced by indexed families of sets. Since all constructions used below are pointwise, the proof generalizes.

An admissible relation from domain A to domain B is simply a subdomain of $A \times B$. Equality is an admissible relation since it is given by the diagonal map, and reindexing preserves admissible relations by Lemma 5.1. That admissible relations are closed under conjunction and universal quantification is a consequence of the same lemma.

Suppose $\rho \subseteq A \times B$ is a subdomain. By Lemma 6.7

$$\begin{array}{rcl} !\rho & = & \{(e,f) \in LA \times LB \mid e\downarrow\supset\subset f\downarrow \wedge e\downarrow\supset (\epsilon e, \epsilon f) \in \rho\} \\ & = & \{(e,f) \in LA \times LB \mid \exists x \in e \supset\subset \exists y \in f \wedge \forall x \in e, y \in f.\, (x,y) \in \rho\} \end{array}$$

So $!\rho$ is given by the lift of the span $A \leftarrow \rho \rightarrow B$, and so is a subdomain of $LA \times LB$.

If ϕ is a proposition and ρ is an admissible relation, then

$$\{(x,y) \mid \phi \supset \rho(x,y)\} = \bigcap_{z \in \{0 \mid \phi\}} \{(x,y) \mid \rho(x,y)\}$$

which is an admissible relation by Lemma 5.1. So $(x,y).\, \phi \supset \rho(x,y)$ is an admissible relation.

Finally, we need to check that Rule 2.18 of [3] holds. Suppose $\rho\colon \mathbf{Rel}(LA, LB)$ and $\rho'\colon \mathbf{AdmRel}(LA, LB)$. We must show that if

$$\forall x\colon A, y\colon B.\, \rho(\{x\},\{y\}) \supset \rho'(\{x\},\{y\}) \tag{7}$$

then

$$\forall e\colon LA, f\colon LB.\, (e\downarrow\supset\subset f\downarrow) \supset \rho(e,f) \supset \rho'(e,f).$$

So assume (7) and that $e\colon LA, f\colon LB$ are such that $e\downarrow\supset\subset f\downarrow \wedge \rho(e,f)$. Denote by p the truth value $e\downarrow$. Now,

$$\{(e,f) \mid p\} \in (\rho')^p$$

since p implies $e = \{x\}$ and $f = \{y\}$ for some x, y and so the assumption $\rho(e, f)$, implies $\rho'(e, f)$ by (7). Since ρ' is a pointed subset of $LA \times LB$ we must have

$$r_p^{LA \times LB}(\{(e, f) \mid p\}) \in \rho'.$$

But $r_p^{LA\times LB}(\{(e, f) \mid p\}) = (r_p^{LA}(\{e \mid p\}), r_p^{LB}(\{f \mid p\}))$ and

$$r_p^{LA}(\{e \mid p\}) = \bigcup_{y \in \{e \mid p\}} y = \{x \in A \mid \exists y \in \{e \mid p\}.\, x \in y\} = e$$

and likewise $r_p^{B}(\{f \mid p\}) = f$, so $\rho'(e, f)$ as desired. □

Finally, to show that we have a full LAPL-structure we must show that all types have a relational interpretation. Of course, such a relational interpretation of a type (f^r, f^p) is f^r. We must check, however, that the linear structure on types defined in the model here agrees with the linear structure on $\mathbf{LinAdmRel} \to \mathbf{AdmRelCtx}$ defined abstractly in the LAPL-logic.

Theorem 6.8. *The pre-LAPL-structure (5) has a full LAPL-structure.*

Proof. The category $\mathbf{AdmRelCtx}$ has as objects triples (n, m, f) where n, m are natural numbers and f is an object of $\mathbf{DFam}(\mathbf{Set})_{n+m}$, i.e. a functor

$$(\mathbf{Dom}_\perp)_{\mathtt{iso}}^{n+m} \to \mathbf{Set}.$$

A morphisms from (n, m, f) to (n', m', f') is a pair of morphisms

$$(a^r, a^d) \colon n \to n', (b^r, b^d) \colon m \to m'$$

in $\mathbf{PDom}$ and a vertical morphism $t \colon f \to f' \circ (a^d \times b^d)$ in $\mathbf{DFam}(\mathbf{Set})_{n+m}$.

An object of $\mathbf{LinAdmRel}$ over (n, m, f) is a pair of objects $((g^r, g^d), (h^r, h^d)) \in \mathbf{PFam}(\mathbf{Dom}_\perp)_n \times \mathbf{PFam}(\mathbf{Dom}_\perp)_m$ plus a natural transformation

$$(k_{\vec{A},\vec{B}} \colon f^d(\vec{A}, \vec{B}) \to \mathbf{AdmRel}(g^d(\vec{A}), h^d(\vec{B})))_{(\vec{A},\vec{B}) \in (\mathbf{Dom}_\perp)_{\mathtt{iso}}^{n+m}}.$$

A vertical morphism in $\mathbf{LinAdmRel}$ from $((g^r, g^d), (h^r, h^d), k)$ to

$$(((g')^r, (g')^d), ((h')^r, (h')^d), (k'))$$

is a pair of morphisms

$$\begin{array}{l} t \colon (g^r, g^d) \to ((g')^r, (g')^d) \text{ in } \mathbf{PFam}(\mathbf{Dom}_\perp)_n \\ s \colon (h^r, h^d) \to ((h')^r, (h')^d) \text{ in } \mathbf{PFam}(\mathbf{Dom}_\perp)_m \end{array}$$

such that for all $\vec{A}, \vec{B}, x \in f^d(\vec{A}, \vec{B})$

$$\forall y, z.\, k_{\vec{A},\vec{B}}(x)(y, z) \supset k'_{\vec{A},\vec{B}}(x)(t_{\vec{A}}(y), s_{\vec{B}}(z))$$

We have a pair of maps of PILL_Y-models:

$$\begin{tikzcd} \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[d] & & \mathbf{LinAdmRel} \arrow[d] \arrow[ll] \\ \mathbf{PDom} & & \mathbf{AdmRelCtx} \arrow[ll] \end{tikzcd}$$

defined by mapping an object of $\mathbf{LinAdmRel}$, $((g^r, g^d), (h^r, h^d), k)$ to (g^r, g^d) and (h^r, h^d) respectively. We define the mapping Φ going the other way by first defining the map

$$\mathbf{PDom} \to \mathbf{AdmRelCtx}$$

to map an object n to $(n, n, \prod_{i \le n} V(\pi_i \circ \pi, \pi_i \circ \pi'))$ where π, π' are the first and second projections respectively $n + n \to n$ and $\pi_i \colon n \to 1$ is the i'th projection. One may also describe this object as the family

$$(\textstyle\prod_{i \le n} \mathbf{AdmRel}(A_i, B_i))_{\vec{A} \in \mathbf{Dom}^n, \vec{B} \in \mathbf{Dom}^n}$$

in the fibre $\mathbf{DFam}(\mathbf{Set})_{n+n}$.

Since objects in $\mathbf{PDom}$ are products of the generic object, if we are to define a map of PILL$_Y$-models, the action of the functor between the base categories on morphisms is completely determined by the action of the functor on the total categories, so we will describe the latter.

Suppose (f^d, f^r) is an object of $\mathbf{PFam}(\mathbf{Dom}_\perp)_n$. We map this to the object of $\mathbf{LinAdmRel}$ given by the pair of types $((f^d, f^r), (f^d, f^r))$ and the natural transformation

$$(\vec{R} \in \textstyle\prod_{i \le n} \mathbf{AdmRel}(A_i, B_i) \mapsto f^r(\vec{R}) \in \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B})))_{\vec{A}, \vec{B}}.$$

Given a map t from (f^d, f^r) to (g^d, g^r), that is, a natural transformation

$$(t_{\vec{A}} \colon f^d(\vec{A}) \multimap g^d(\vec{A}))_{\vec{A}}$$

preserving relations, we map it to the pair (t, t). To see that this defines a map from $\Phi(f^d, f^r)$ to $\Phi(g^d, g^r)$ we need to see that it preserves relations , which writing it out is the exact same condition as for t to preserve relations in the first place.

It is easy to see that Φ commutes with reindexing and therefore defines a map of fibrations. It is also evident that Φ together with the domain and codomain maps constitute a reflexive graph.

The generic object in $\mathbf{LinAdmRel} \to \mathbf{AdmRelCtx}$ is the object over

$$(1, 1, (\mathbf{AdmRel}(A, B))_{A,B})$$

in $\mathbf{AdmRelCtx}$ given by the pair of types $((id, id), (id, id))$ and the natural transformation

$$(id \colon \mathbf{AdmRel}(A, B) \to \mathbf{AdmRel}(A, B))_{A,B}.$$

It is clear that Φ preserves generic object. It is also clear that it preserves products in the base.

Let us show that Φ preserves !. Recall that applying ! in $\mathbf{PFam}(\mathbf{Dom}_\perp)$ maps a relation to the relation obtained by lifting both maps in the span. Logically this gives us that $\Phi(!(f^r, f^d))$ is

$$\vec{R} \mapsto \{(g, h) \in Lf^d(\vec{A}) \times Lf^d(\vec{B}) \mid \exists z \in Lf^r(\vec{R}).\, L(\pi)(z) = g \wedge L(\pi')(z) = h\}$$

But $\exists z \in Lf^r(\vec{R}).\, L(\pi)(z) = g \wedge L(\pi')(z) = h$ is equivalent to

$$(\exists x \in g \supset\subset \exists y \in h) \wedge \forall x, y \,.\, x \in g, y \in h \supset (x, y) \in f^r(\vec{R}) \tag{8}$$

If we apply the ! in $\mathbf{LinAdmRel}$ to $\Phi(f^r, f^d)$ we obtain

$$\vec{R} \mapsto \{(g, h) \in Lf^d(\vec{A}) \times Lf^d(\vec{B}) \mid (g \downarrow \supset\subset h \downarrow) \wedge g \downarrow \supset (\epsilon g, \epsilon h) \in f^r(\vec{R})\}$$

But since $g\downarrow \supset\subset \exists x\in g$ by Lemma 6.7 and $x\in g\supset \epsilon g = x$, this is the same as (8).

To see that the simple products are preserved, an easy calculation shows that both combination of simple products and Φ map (f^r, f^d) to the relation

$$\vec{R}\mapsto \{(x,y)\in \prod f^d(\vec{A})\times \prod f^d(\vec{B}) \mid \forall D, D'\in \mathbf{D}. \forall S\colon \mathbf{AdmRel}(D, D'). (x_D, x_{D'})\in f^r(\vec{R}, S)\}.$$

Likewise it is easily seen that Φ preserves $\multimap$.

Finally, we show that Φ preserves $\otimes$. Suppose $(f^r, f^d), (g^r, g^d)$ are types. Maps out of $\Phi((f^r, f^d)\otimes (g^r, g^d))$ in $\mathbf{LinAdmRel}$ are easily seen, using an argument as in Lemma 5.4, to correspond to pairs of bistrict maps out of $f^d\times g^d$ preserving $f^r\times g^r$. Since maps out of $\Phi(f^r, f^d)\otimes\Phi(g^r, g^d)$ satisfy the same universal property, we get that Φ preserves tensor. □

Theorem 6.9. *The LAPL-structure (5) is a parametric LAPL-structure, i.e. satisfies identity extension, extensionality and very strong equality.*

Proof. Let us first prove that (5) satisfies identity extension. Suppose we are given a type (f^d, f^r). The relational interpretation of this type is

$$(f^r\colon \textstyle\prod_{i\leq n}\mathbf{AdmRel}(A_i, B_i)\to \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B})))_{\vec{A},\vec{B}}.$$

Instantiating this at equality we obtain

$$[\![\vec{\alpha}\mid -\mid -\vdash (f^d, f^r)[eq_{\vec{\alpha}}]\colon \mathbf{AdmRel}((f^d, f^r)(\vec{\alpha}), (f^d, f^r)(\vec{\alpha}))]\!]$$

which is the element of

$$(\mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{A})))_{\vec{A}}$$

given as

$$(f^r(eq_{\vec{A}}))_{\vec{A}} = (eq_{f^d(\vec{A})})_{\vec{A}}$$

which is also

$$[\![\vec{\alpha}\mid -\mid -\vdash eq_{(f^d, f^r)}\colon \mathbf{AdmRel}((f^d, f^r)(\vec{\alpha}), (f^d, f^r)(\vec{\alpha}))]\!].$$

Very strong equality follows from very strong equality in the subobject fibration over $\mathbf{Set}$. Extensionality is a consequence of very strong equality. □

7 Proving consequences of parametricity for Lily$_{\text{strict}}$

In [14] a language, which we shall call Lily$_{\text{strict}}$ is introduced. This language is a modification of Lily [1], where the function type $\sigma \multimap \tau$ is interpreted as strict rather than linear functions. The reason for using strictness rather than linearity is that it is more general, i.e., gives types to more terms, and that it is exactly what is needed for call-by-value and call-by-name to give the same notion of ground contextual equivalence. This intuitively also corresponds more directly to strict functions in domain theory, since these are the functions that diverge if their input does.

Simpson and Rosolini define an interpretation of Lily$_{\text{strict}}$ into models of synthetic domain theory, and use this to prove that call-by-value and call-by-name give the same notion of contextual equivalence. This has been proved for Lily in [1] using operational methods, but Simpson and Rosolini give a different semantic proof. In [1] operational methods are also used for proving simple consequences of parametricity for Lily, and in this section, we show how to use the LAPL-structure (5) to prove more advanced parametricity results for $\text{Lily}_{\text{strict}}$.

The model of PILL_Y in (5) is based on the interpretation of $\text{Lily}_{\text{strict}}$ given in [14]. In this section we show that the two interpretations of PILL_Y and $\text{Lily}_{\text{strict}}$ are basically the same. The two languages are of course not the same, but since linear maps are strict, we can basically include PILL_Y into $\text{Lily}_{\text{strict}}$, and show that the interpretations agree up to this inclusion.

As mentioned earlier, the LAPL-structure we have constructed using synthetic domain theory is really a family of LAPL-structures, since we have one LAPL-structure for each model of synthetic domain theory.

In this section we will assume that we have chosen one such model which is also 1-consistent in the sense of [16, 18]: any sentence of the form $\exists n \colon \mathbb{N}.\, \phi(n)$, for ϕ a primitive recursive predicate,—a Σ^0_1-sentence—is true in the model iff there exists (in the external sense) a natural number n such that $\phi(n)$ is true. This is, for example, the case for a realizability topos satisfying the strong completeness axiom [6] where one takes predomains to be the well-complete objects. The reason for this assumption is that adequacy (Theorem 7.9 below), will be proved in the internal language of the model; it will hold in the real world only under the assumption of 1-consistency (and precisely when 1-consistency holds), as explained also in Section 8 of [14]. This technique was introduced in [16, 18].

We emphasize that the results about $\text{Lily}_{\text{strict}}$ (Theorems 7.18, 7.19) hold in general and independently of any model. Yet, to prove the results we need to refer to a model of SDT satisfying 1-consistency (which is known to exist).

7.1 The language Lily$_{\text{strict}}$

This subsection sums up some definitions and results from [14]. In particular we recall the language $\text{Lily}_{\text{strict}}$ with two operational semantics, a call-by-value and a call-by-name semantics. Each of these semantics gives rise to a concept of ground contextual equivalence corresponding to observing termination at !- types. We also recall the interpretation of the $\text{Lily}_{\text{strict}}$ into SDT defined in [14]. In *loc. cit.* it is also shown that the interpretation is adequate with respect to both notions of contextual equality, and using adequacy it is shown that the two ground contextual equivalences coincide.

The types of $\text{Lily}_{\text{strict}}$ are

$$\sigma, \tau ::= \alpha \mid \sigma \multimap \tau \mid !\sigma \mid \textstyle\prod \alpha.\, \sigma$$

where α ranges over an infinite set of type variables. Except for $\otimes, I$ these are exactly the types of PILL_Y. The notation $\vdash_\Xi \sigma \colon \mathsf{Type}$ means that σ is a well formed type with free type variables contained in Ξ.

Typing judgements of $\text{Lily}_{\text{strict}}$ are of the form

$$\Gamma \mid \delta \vdash_\Xi t \colon \sigma$$

where Γ is the context of free variables, i.e., an assignment of types to a finite set of variables usually written as $x_1 \colon \sigma_1, \dots, x_n \colon \sigma_n$ such that the free variables of t are contained in the domain of Γ, i.e., $\{x_1, \dots, x_n\}$. Ξ is a finite set of free type variables containing the free type variables of $\sigma_1, \dots, \sigma_n, \sigma$. The notation Ξ, α means $\Xi \cup \alpha$ *and* $\alpha \notin \Xi$. δ is a labeling of the variables in the domain of Γ, i.e., a map from $\{x_1, \dots, x_n\}$ to $\{0, 1\}$. Intuitively $\delta(x_i) = 1$ means that t is strict in x_i.

The notation $\vdash_\Xi \Gamma$ means that Γ is a well-formed context with free variables contained in Ξ.

Figure 1 recalls the term formation rules as defined in [14]. The notation $\Gamma \mid \delta, x :_i \sigma \vdash_\Xi t \colon \tau$ for $i = 0, 1$ is short for $\Gamma, x \colon \sigma \mid \delta[x \mapsto i] \vdash_\Xi t \colon \tau$, where $\delta[x \mapsto i]$ is the extension of δ to *dom*$(\delta) \cup \{x\}$ such that

$\delta(x) = i$. The notation $x :_{-} \sigma$ means that either $x :_0 \sigma$ or $x :_1 \sigma$. For δ, δ' labellings of the same set of variables, the notation $\delta \vee \delta'$ is the labeling mapping $x\colon \mathit{dom}(\delta)$ to $\max(\delta(x), \delta'(x))$. The constant zero labeling is denoted **0**.

$$\frac{}{\Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_\Xi x\colon \sigma} \qquad \frac{\Gamma \mid \delta, x :_1 \sigma \vdash_\Xi t\colon \tau}{\Gamma \mid \delta \vdash_\Xi \lambda x :_1 \sigma.\, t\colon \sigma \multimap \tau}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi s\colon \sigma \multimap \tau \qquad \Gamma \mid \delta' \vdash_\Xi t\colon \sigma}{\Gamma \mid \delta \vee \delta' \vdash_\Xi s(t)\colon \tau} \qquad \frac{\Gamma \mid \delta \vdash_\Xi t\colon \sigma}{\Gamma \mid \mathbf{0} \vdash_\Xi !t\colon !\sigma}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi s\colon !\sigma \qquad \Gamma \mid \delta', x :_{-} \sigma \vdash_\Xi t\colon \tau}{\Gamma \mid \delta \vee \delta' \vdash_\Xi \text{let } !x \text{ be } s \text{ in } t} \qquad \frac{\Gamma \mid \delta \vdash_{\Xi,\alpha} t\colon \sigma \qquad \vdash_\Xi \Gamma}{\Gamma \mid \delta \vdash_\Xi \Lambda\alpha.\, t\colon \prod \alpha.\, \sigma}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi t\colon \prod \alpha.\, \sigma \qquad \vdash_\Xi \tau\colon \mathsf{Type}}{\Gamma \mid \delta \vdash_\Xi t(\tau)\colon \sigma[\tau/\alpha]} \qquad \frac{\Gamma \mid \delta, x :_{-} \sigma \vdash_\Xi t\colon \sigma}{\Gamma \mid \delta \vdash_\Xi \text{rec } x\colon \sigma.\, t\colon \sigma}$$

Figure 1: Term formation rules for Lily$_{\text{strict}}$

Lemma 7.1. *If both* $\Gamma \mid \delta \vdash_\Xi t\colon \sigma$ *and* $\Gamma \mid \delta' \vdash_\Xi t\colon \sigma'$ *then* $\delta = \delta'$ *and* $\sigma = \sigma'$.

Lemma 7.2. *Suppose* $\Gamma \mid \delta \vdash_\Xi t\colon \tau$ *is typable in Lily*$_{\textit{strict}}$*, and* $\Xi \vdash \sigma\colon \mathsf{Type}$ *is a type in Lily*$_{\textit{strict}}$ *and* $x \notin \mathit{dom}(\Gamma)$*. Then* $\Gamma \mid \delta, x :_0 \sigma \vdash_\Xi t\colon \tau$ *is typable in Lily*$_{\textit{strict}}$*.*

Figure 2 recalls the two operational semantics for Lily$_{\text{strict}}$ as defined in [14]. Formally these are given as relations $t \Downarrow^s v$ and $t \Downarrow^n v$ between closed terms t of closed types and values v, where the set of values is the set of closed terms of closed types of the form

$$v ::= \lambda x\colon \sigma.\, t \mid !t \mid \Lambda\alpha.\, t.$$

$$\frac{}{\lambda x\colon \sigma.\, t \Downarrow \lambda x\colon \sigma.\, t} \qquad \frac{s \Downarrow^s \lambda x\colon \sigma.\, s' \qquad t \Downarrow^s v' \qquad s'[v'/x] \Downarrow^s v}{s(t) \Downarrow^s v}$$

$$\frac{s \Downarrow^n \lambda x\colon \sigma.\, s' \qquad s'[t/x] \Downarrow^n v}{s(t) \Downarrow^n v} \qquad \frac{}{!t \Downarrow !t}$$

$$\frac{s \Downarrow !s' \qquad t[s'/x] \Downarrow v}{\text{let } !x \text{ be } s \text{ in } t \Downarrow v} \qquad \frac{}{\Lambda\alpha.\, t \Downarrow \Lambda\alpha.\, t}$$

$$\frac{t \Downarrow \Lambda\alpha.\, t' \qquad t'[\sigma/\alpha] \Downarrow v}{t(\sigma) \Downarrow v} \qquad \frac{t[\text{rec } x\colon \sigma.\, t/x] \Downarrow v}{\text{rec } x\colon \sigma.\, t \Downarrow v}$$

Figure 2: Operational semantics of Lily$_{\text{strict}}$

In Figure 2 the notation $t \Downarrow v$ is used in some rules. This means that each of these rules exist both in the definition of the $\Downarrow^n$ and the $\Downarrow^s$ semantics. The notation $t \Downarrow^n$ is short for $\exists v.\, t \Downarrow^n v$ and likewise for $t \Downarrow^s$.

Lemma 7.3. *If* $t \Downarrow^n v$ *and* $t \Downarrow^n v'$ *then* $v = v'$*. Likewise for* $\Downarrow^s$*.*

Lemma 7.4. *If $t \Downarrow^n v$ or $t \Downarrow^s v$, then t, v have the same type.*

A ground σ-context is a term $x :_{-} \sigma \vdash C \colon !\tau$ for some type τ, and for closed $t \colon \sigma$ we write $C[t]$ for $C[t/x]$.

Definition 7.5. *For $t, t' \colon \sigma$ closed terms of closed types, we write $t \equiv^{s}_{gnd} t'$ if for all ground σ-contexts $C[-]$, $C[t] \Downarrow^s$ iff $C[t'] \Downarrow^s$. Likewise $t \equiv^{n}_{gnd} t'$ if for all ground σ-contexts $C[-]$, $C[t] \Downarrow^n$ iff $C[t'] \Downarrow^n$*

The idea of $\equiv^{\mathrm{n}}_{\mathrm{gnd}}, \equiv^{\mathrm{s}}_{\mathrm{gnd}}$ is that terms are considered equal if one can be substituted for the other in larger programs without observable difference in behavior of the resulting program. As in [14] and [1], here observable behavior refers to termination at !-types.

Lemma 7.6. *The relations $\equiv^{s}_{gnd}, \equiv^{n}_{gnd}$ are equivalence relations and congruences. The latter means that if $x :_{-} \sigma \vdash C \colon \tau$ is some term, and $t \equiv^{s}_{gnd} t'$ then $C[t/x] \equiv^{s}_{gnd} C[t'/x]$ and likewise for $\equiv^{n}_{gnd}$.*

We now recall the interpretation of Lily$_{\text{strict}}$ into SDT defined in [14]. Each type $\vdash_{\alpha_1, \ldots \alpha_n} \sigma$ has two interpretations. The first $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d$ is a $\mathbf{Dom}_0^n$ indexed family of domains, where $\mathbf{Dom}_0$ is the class of domains. We write the $\vec{D}$'th component of $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d$ as $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d_{\vec{D}}$. The second interpretation of a type is the relational interpretation $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^r$. This is a family of relations

$$(\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^r_{\vec{R}} \colon \mathbf{AdmRel}(\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d_{\vec{A}}, \llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d_{\vec{B}}))_{\vec{R} \colon \mathbf{AdmRel}(\vec{A}, \vec{B})}$$

indexed over $\mathbf{AdmRel}_0^n$ where $\mathbf{AdmRel}_0$ is the class of admissible relations on domains. The interpretation of types is defined in Figure 3.

$$\begin{array}{rcl}
\llparenthesis \vdash_{\vec{\alpha}} \alpha_i \rrparenthesis^d_{\vec{D}} & = & (D_i) \\
\llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{D}} & = & \llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d_{\vec{D}} \multimap \llparenthesis \vdash_{\vec{\alpha}} \tau \rrparenthesis^d \\
\llparenthesis \vdash_{\vec{\alpha}} !\sigma \rrparenthesis^d_{\vec{D}} & = & L\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis_{\vec{D}} \\
\llparenthesis \vdash_{\vec{\alpha}} \prod \alpha .\, \sigma \rrparenthesis^d_{\vec{D}} & = & \{\pi \in \prod_{D \in \mathbf{D}} \llparenthesis \vdash_{\Xi, \alpha} \sigma \rrparenthesis^d_{\vec{D}, D} \mid \forall D, D' \in \mathbf{D}. \\
 & & \forall R \colon \mathbf{AdmRel}(D, D').\, \llparenthesis \vdash_{\Xi, \alpha} \sigma \rrparenthesis^r_{\vec{eq}, R}(\pi_D, \pi_{D'})\}
\end{array}$$

$$\begin{array}{rcl}
\llparenthesis \vdash_{\vec{\alpha}} \alpha_i \rrparenthesis^r_{\vec{R} \colon \mathbf{AdmRel}(\vec{A}, \vec{B})} & = & R_i \\
\llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^r_{\vec{R} \colon \mathbf{AdmRel}(\vec{A}, \vec{B})} & = & \{(f, g) \in \llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{A}} \times \llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{B}} \mid \\
 & & \forall (x, y) \in \llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^r_{\vec{R}}.\, (f(x), g(y)) \in \llparenthesis \vdash_{\vec{\alpha}} \tau \rrparenthesis^r_{\vec{R}}\} \\
\llparenthesis \vdash_{\vec{\alpha}} !\sigma \rrparenthesis^r_{\vec{R} \colon \mathbf{AdmRel}(\vec{A}, \vec{B})} & = & \{(e, f) \colon L\llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}) \times L\llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}) \mid \\
 & & (\forall x \colon \llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}).\, x \in e \supset \exists y \in f \supset \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})(x, y)) \wedge \\
 & & (\forall y \colon \llbracket \Xi \mid \sigma \rrbracket^d(\vec{B}).\, y \in f \supset \exists x \in e \supset \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})(x, y))\} \\
\llparenthesis \vdash_{\vec{\alpha}} \prod \alpha .\, \sigma \rrparenthesis^r_{\vec{R} \colon \mathbf{AdmRel}(\vec{A}, \vec{B})} & = & \{(\pi, \pi') \in \llparenthesis \vdash_{\Xi} \prod \alpha .\, \sigma \rrparenthesis^d_{\vec{A}} \times \llparenthesis \vdash_{\Xi} \prod \alpha .\, \sigma \rrparenthesis^d_{\vec{B}} \mid \\
 & & \forall D, D' \in \mathbf{D}. \forall R \colon \mathbf{AdmRel}(D, D').\, \llparenthesis \vdash_{\Xi, \alpha} \sigma \rrparenthesis^r_{\vec{eq}, R}(\pi_D, \pi_{D'})\}
\end{array}$$

Figure 3: Interpretation of types

Suppose $f \colon A \multimap B$ is a pointed map. We write $\langle f \rangle \colon \mathbf{AdmRel}(A, B)$ for the graph of f.

Lemma 7.7 ([14]). *Suppose $(f_i \colon A_i \multimap B_i)_{i \leq n}$ are isomorphisms in $(\mathbf{Dom}_\perp)_{\mathtt{iso}}$ and $\vdash_{\alpha_1, \ldots \alpha_n} \sigma \colon$* Type *is a type in Lily$_{strict}$. Then there exists an isomorphism $\llbracket \sigma \rrbracket^d(\vec{f})$ in $\mathbf{Dom}_\perp$ such that $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^r_{(\langle f_i \rangle)_i}$ is the graph of $\llbracket \sigma \rrbracket^d(\vec{f})$. Moreover the correspondence $(f_i)_i \mapsto \llbracket \sigma \rrbracket^d(\vec{f})$ is functorial.*

To define the interpretation of terms

$$\Gamma \mid \delta \vdash_\Xi t \colon \sigma$$

define first the interpretation of the context Γ:

$$\llbracket \vdash_\Xi x_1 \colon \sigma_1, \dots, \sigma_n \colon \sigma_n \rrbracket^d = (\textstyle\prod_i \llbracket \vdash_\Xi \sigma_i \rrbracket_{\vec{D}})_{\vec{D}}.$$

Since the labeling does not play a role in the interpretation of terms and is uniquely determined by Ξ, Γ, t, we leave it out in the notation and write $\llbracket \Gamma \vdash_\Xi t \colon \sigma \rrbracket$ in stead of $\llbracket \Gamma \mid \delta \vdash_\Xi t \colon \sigma \rrbracket$.

For $\Xi = \alpha_1, \dots \alpha_n$, the interpretation $\llbracket \Gamma \vdash_\Xi t \colon \tau \rrbracket$ is a family of maps

$$(\llbracket \Gamma \vdash_\Xi t \colon \tau \rrbracket_{\vec{D}} \colon \llbracket \vdash_\Xi \Gamma \rrbracket^d_{\vec{D}} \to \llbracket \tau \rrbracket^d_{\vec{D}})_{\vec{D}}$$

The interpretation of terms is defined in Figure 4. In the definition of the interpretation of the let-expression the $(-)^*$-notation is used to denote the strict map

$$L\llbracket \vdash_\Xi \sigma \rrbracket \multimap \llbracket \vdash_\Xi \tau \rrbracket$$

corresponding to the set theoretic map

$$\llbracket \vdash_\Xi \sigma \rrbracket \to \llbracket \vdash_\Xi \tau \rrbracket$$

described. The notation

$$\llbracket \sigma \rrbracket^d(id_{\vec{D}}, i)$$

refers to the morphism of Lemma 7.7.

$$\begin{array}{rcl}
\llbracket \Gamma \vdash_\Xi \ x_i \colon \sigma_i \rrbracket_{\vec{D}}(\vec{x}) & = & x_i \\
\llbracket \Gamma \vdash_\Xi \lambda x \colon \sigma. t \colon \sigma \multimap \tau \rrbracket_{\vec{D}}(\vec{x}) & = & d \colon \llbracket \vdash_\Xi \sigma \rrbracket^d_{\vec{D}} \mapsto \llbracket \Gamma, x \colon \sigma \vdash_\Xi t \colon \tau \rrbracket_{\vec{D}}(\vec{x}, d) \\
\llbracket \Gamma \vdash_\Xi s(t) \colon \tau \rrbracket_{\vec{D}}(\vec{x}) & = & \llbracket \Gamma \vdash_\Xi s \colon \sigma \multimap \tau \rrbracket_{\vec{D}}(\vec{x})(\llbracket \Gamma \vdash_\Xi t \colon \sigma \rrbracket_{\vec{D}}(\vec{x})) \\
\llbracket \Gamma \vdash_\Xi !t \rrbracket_{\vec{D}}(\vec{x}) & = & \{\llbracket \Gamma \vdash_\Xi t \rrbracket_{\vec{D}}(\vec{x})\} \\
\llbracket \Gamma \vdash_\Xi \text{let } !x \text{ be } s \text{ in } t \rrbracket_{\vec{D}}(\vec{x}) & = & (d \colon \llbracket \vdash_\Xi \sigma \rrbracket \mapsto \llbracket \Gamma, x \colon \sigma \vdash_\Xi t \rrbracket_{\vec{D}}(\vec{x}, d))^* \\
 & & (\llbracket \Gamma \vdash_\Xi s \colon !\sigma \rrbracket_{\vec{D}}(\vec{x})) \\
\llbracket \Gamma \vdash_\Xi \Lambda\alpha. t \colon \prod \alpha. \sigma \rrbracket_{\vec{D}}(\vec{x}) & = & ((\llbracket \Gamma \vdash_{\Xi,\alpha} t \colon \sigma \rrbracket_{\vec{D},D}(\vec{x}))_{D \in \mathbf{D}} \\
\llbracket \Gamma \vdash_\Xi t(\tau) \colon \sigma[\tau/\alpha] \rrbracket_{\vec{D}}(\vec{x}) & = & \llbracket \vdash_\Xi \sigma \rrbracket^d(id_{\vec{D}}, i)((\llbracket \Gamma \vdash_\Xi t \colon \prod \alpha. \sigma \rrbracket_{\vec{D}}(\vec{x}))_D \\
 & & \text{where } D \in \mathbf{D} \text{ and } i \colon D \multimap \llbracket \vdash_\Xi \tau \rrbracket^d_{\vec{D}} \text{ is an iso} \\
\llbracket \Gamma \vdash_\Xi \text{rec}\, x \colon \sigma. t \colon \sigma \rrbracket_{\vec{D}}(\vec{x}) & = & \mathit{fix}(d \colon \llbracket \vdash_\Xi \sigma \rrbracket \mapsto \llbracket \Gamma, x \colon \sigma \vdash_\Xi t \colon \sigma \rrbracket_{\vec{D}}(\vec{x}, d))
\end{array}$$

Figure 4: The interpretation of terms

The definition of the interpretation of type application involves a choice, and so should be checked to be well-defined. This is the first condition of the next lemma. The proof of well-definedness can be done as in the proof of Lemma 4.2.

Lemma 7.8 ([14]). *Suppose* $\Gamma \vdash_\Xi t \colon \tau$. *Then*

- $\llbracket \Gamma \vdash_\Xi t \colon \tau \rrbracket$ *is well defined*
- *If* $\Gamma = x_1 \colon \sigma_1, \dots, x_n \colon \sigma_n$ *and* $\delta(x_i) = 1$ *then for each vector* $\vec{D}$ *of domains, the function* $\llbracket \Gamma \mid \delta \vdash_\Xi t \colon \tau \rrbracket_{\vec{D}}$ *is strict in the* i*'th variable.*

- *If $\vec{R}\colon \mathbf{AdmRel}(\vec{D}, \vec{D}')$ is a vector of admissible relations, and $x_1, \dots x_n$, $x'_1, \dots, x'_n$ are elements such that for all i,*
  $$(x_i, x'_i) \in (\!|\vdash_\Xi \sigma_i|\!)^r_{\vec{R}}$$
  then
  $$((\!|\Gamma \vdash_\Xi t|\!)_{\vec{D}}(\vec{x}), (\!|\Gamma \vdash_\Xi t|\!)_{\vec{D}'}(\vec{x}')) \in (\!|\tau|\!)^r_{\vec{R}}$$

The last property of Lemma 7.8 is the Logical Relations Lemma.

The following theorems are proved in [14].

Theorem 7.9 (Adequacy [14]). *Suppose $t\colon \tau, t'\colon \tau$ are closed terms of closed type. If $(\!|t|\!) = (\!|t'|\!)$ then $t \equiv^s_{gnd} t'$ and $t \equiv^n_{gnd} t'$.*

Theorem 7.10 (Strictness [14]). *If $t\colon !\sigma$ is a closed term of closed type, then $t \Downarrow^n$ iff $t \Downarrow^s$. In particular $\equiv^s_{gnd}$ and $\equiv^n_{gnd}$ coincide.*

Since $\equiv^{\mathrm{s}}_{\mathrm{gnd}}$ and $\equiv^{\mathrm{n}}_{\mathrm{gnd}}$ coincide we introduce the notation $\equiv_{\mathrm{gnd}}$ to stand for either of them.

Remark 7.11. *The assumption that the given model of SDT is 1-consistent stated in the beginning of this section is used in the proof of Theorem 7.9. In fact Theorem 7.9 is stated in [14] in a more general form, since there it is formulated in the internal language of the model. This technique for proving computational adequacy was developed in [16, 18], in which it is also proved that adequacy holds in the external sense stated here if and only if the given model of SDT is 1-consistent.*

7.2 Translating PILL$_Y$ into Lily

Consider the language PILL$_Y \setminus \otimes$ obtained by removing the type-constructors $\otimes$, I from PILL$_Y$ and removing the corresponding term constructors such as the corresponding let-expressions, $\star$, and $\otimes$ of terms.

The types of PILL$_Y \setminus \otimes$ and Lily$_{\mathrm{strict}}$ are the same, and the main difference between the two languages is that $\multimap$ in Lily$_{\mathrm{strict}}$ is interpreted as strict maps, and in PILL$_Y \setminus \otimes$ it is interpreted as linear maps. Since linear maps are strict, we can basically include PILL$_Y \setminus \otimes$ into Lily$_{\mathrm{strict}}$. Up to this inclusion of languages the interpretations $(\!|-|\!)$ of Lily$_{\mathrm{strict}}$ and $[\![-]\!]$ of PILL$_Y \setminus \otimes$ agree.

Theorem 7.12. *There exists an interpretation ϕ of PILL$_Y \setminus \otimes$ into Lily$_{strict}$ such that for all closed terms t of PILL$_Y$, $[\![t]\!] = (\!|\phi(t)|\!)$. This translation is the identity on types.*

The translation is functorial in the following sense: Suppose $u\colon \sigma \multimap \tau, t\colon \tau \multimap \omega$ are closed terms of closed types of PILL$_Y \setminus \otimes$. Then $\phi(t \circ u) = \phi(t) \circ \phi(u)$.

Before proving Theorem 7.12 we need a lemma. Recall that for any type $\vec{\alpha} \vdash \sigma$ of PILL$_Y \setminus \otimes$, $[\![\vec{\alpha} \vdash \sigma]\!]^d$ is a functor from $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}}$ to $\mathbf{Dom}_\perp$. Since $(\!|\vdash_{\vec{\alpha}} \sigma|\!)$ is an indexed family of domains, it does not make sense to ask if $(\!|\vdash_{\vec{\alpha}} \sigma|\!) = [\![\vec{\alpha} \vdash \sigma]\!]$, but we can still compare the values of $[\![\vec{\alpha} \vdash \sigma]\!]$ on objects with $(\!|\vdash_{\vec{\alpha}} \sigma|\!)$.

Lemma 7.13. *For all types $\Xi \vdash \sigma$ of PILL$_Y \setminus \otimes$, the object parts of $[\![-]\!]^d$ and $[\![-]\!]^r$ agree with $(\!|-|\!)^d$ and $(\!|-|\!)^r$. Moreover, for $\vec{i}$ a vector of strict isomorphism between domains, $[\![-]\!]^d(\vec{i})$ is equal to $(\!|-|\!)^d(\vec{i})$ as defined in Lemma 7.7.*

Proof. The proof is by induction on the structure of types.

Type variables are interpreted as projections by both interpretations. It is easy to see that $\multimap$ is interpreted the same way in both interpretations. Let us consider the interpretation of !. We clearly have

$$
\begin{array}{c}
[\![\Xi \mid !\sigma]\!]^d(\vec{A}) = L([\![\Xi \mid \sigma]\!]^d(\vec{A})) \\
\llparenthesis \Xi \mid !\sigma \rrparenthesis^d(\vec{A}) = L(\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}))
\end{array}
$$

For the relational interpretation, we have for $\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})$.

$$
\begin{array}{rcl}
\llparenthesis \Xi \vdash !\sigma \rrparenthesis^r(\vec{R}) & = & \{(e, f)\colon L\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}) \times L\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}) \mid \\
& & \forall x\colon \llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}).\, x \in e \supset \exists y \in f \supset \llparenthesis \Xi \vdash \sigma \rrparenthesis^r(\vec{R})(x, y) \wedge \\
& & \forall y\colon \llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{B}).\, y \in f \supset \exists x \in e \supset \llparenthesis \Xi \vdash \sigma \rrparenthesis^r(\vec{R})(x, y)\} \\
& = & \{(e, f)\colon L\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}) \times L\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A}) \mid \\
& & e \downarrow \supset\subset f \downarrow \wedge (x \in e \wedge y \in f \supset (x, y) \in \llparenthesis \Xi \vdash \sigma \rrparenthesis^r(\vec{R}))\}.
\end{array}
$$

On the other hand, $[\![\Xi \vdash !\sigma]\!]^r(\vec{R})$ is the image of the span obtained by applying the lifting functor L to both maps in the span

$$
\begin{tikzcd}
& {[\![\Xi \vdash \sigma]\!]^r(\vec{R})} \arrow[dl] \arrow[dr] & \\
{[\![\Xi \vdash \sigma]\!]^d(\vec{A})} & & {[\![\Xi \vdash \sigma]\!]^d(\vec{B}).}
\end{tikzcd}
$$

So $[\![\Xi \vdash !\sigma]\!]^r(\vec{R})$ consists of lifts of pairs from $[\![\Xi \vdash \sigma]\!]^r(\vec{R})$, i.e., pairs (e, f) such that $e \downarrow \supset\subset f \downarrow$ and $x \in e, y \in f \supset (x, y) \in [\![\Xi \vdash \sigma]\!]^r(\vec{R})$.

The interpretation of polymorphic types is the same in the two interpretations. This ends the induction proof.

For the last part of the lemma, suppose $\vec{i}\colon \vec{A} \multimap \vec{B}$ is a vector of isomorphisms. Since for each j the graph of i_j is $(i_j, id_{B_j})^* eq_{B_j}$ and so (i_j, id_{B_j}) is an isomorphism from $\langle i_j \rangle$ to eq_{B_j}. Thus $([\![\sigma]\!]^d(\vec{i}), id_{[\![\sigma]\!]^d(\vec{B})})$ is an isomorphism from $[\![\sigma]\!]^r(\langle \vec{i} \rangle)$ to $eq_{[\![\sigma]\!]^d(\vec{B})}$, i.e.,

$$
([\![\sigma]\!]^d(\vec{i}), id_{[\![\sigma]\!]^d(\vec{B})})^* eq_{[\![\sigma]\!]^d(\vec{B})} = [\![\sigma]\!]^r(\langle \vec{i} \rangle).
$$

We can use this to prove

$$
\langle [\![\sigma]\!]^d(\vec{i}) \rangle = ([\![\sigma]\!]^d(\vec{i}), id_{[\![\sigma]\!]^d(\vec{B})})^* eq_{[\![\sigma]\!]^d(\vec{B})} = [\![\sigma]\!]^r(\langle \vec{i} \rangle) = \llparenthesis \sigma \rrparenthesis^r(\langle \vec{i} \rangle) = \langle \llparenthesis \sigma \rrparenthesis^d(\vec{i}) \rangle,
$$

and so since their graphs agree, $[\![\sigma]\!]^d(\vec{i}) = \llparenthesis \sigma \rrparenthesis^d(\vec{i})$. □

Proof of Theorem 7.12. The interpretation ϕ is defined to be the identity on types. The interpretation of terms is defined inductively in Figure 5, where we have written the definition as rules. It is easily seen that the following properties of the interpretation hold (this is part of the reason the definition makes sense): For any term t of PILL$_Y \setminus \otimes$,

- the free type variables of $\phi(t)$ are the same as for t
- the free variables of $\phi(t)$ is the union of the free intuitionistic and linear variables of t and the types are preserved.
- any free linear variable x in t is labeled 1 in $\phi(t)$, free intuitionistic variables may be labeled either 0 or 1.

$$\begin{array}{c}\phi(\Xi \mid \Gamma; - \vdash Y\colon \prod \alpha.\,!(!\alpha \multimap \alpha) \multimap \alpha) = \\ \Gamma \mid \mathbf{0} \vdash_{\Xi} \Lambda\alpha.\,\lambda t\colon !(!\alpha \multimap \alpha).\,\mathrm{rec}\ x\colon \alpha.\,\mathrm{let}\ !u\ \mathrm{be}\ t\ \mathrm{in}\ u(!x)\end{array}$$

$$\phi(\Xi \mid \Gamma, x\colon \sigma; - \vdash x\colon \sigma) = \Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_{\Xi} x\colon \sigma$$

$$\phi(\Xi \mid \Gamma; x\colon \sigma \vdash x\colon \sigma) = \Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_{\Xi} x\colon \sigma$$

$$\dfrac{\begin{array}{c}\phi(\Xi \mid \Gamma; \Delta \vdash t\colon \sigma \multimap \tau) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(t)\colon \sigma \multimap \tau, \\ \phi(\Xi \mid \Gamma; \Delta' \vdash u\colon \sigma) = \Gamma, \Delta' \mid \delta' \vdash_{\Xi} \phi(u)\colon \sigma\end{array}}{\phi(\Xi \mid \Gamma; \Delta, \Delta' \vdash t\, u\colon \tau) = \Gamma, \Delta, \Delta' \mid \delta \vee \delta' \vdash_{\Xi} \phi(t)\phi(u)\colon \tau}$$

$$\dfrac{\phi(\Xi \mid \Gamma; \Delta, x\colon \sigma \vdash t\colon \tau) = \Gamma, \Delta \mid \delta, x :_1 \sigma \vdash_{\Xi} \phi(t)\colon \tau}{\phi(\Xi \mid \Gamma; \Delta \vdash \lambda^{\circ} x\colon \sigma.\, t\colon \tau) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \lambda x\colon \sigma.\, \phi(t)\colon \sigma \multimap \tau}$$

$$\dfrac{\phi(\Xi \mid \Gamma; - \vdash t\colon \sigma) = \Gamma \mid \delta \vdash_{\Xi} \phi(t)\colon \sigma}{\phi(\Xi \mid \Gamma; - \vdash !t\colon !\sigma) = \Gamma \mid \mathbf{0} \vdash_{\Xi} !\phi(t)\colon \sigma}$$

$$\dfrac{\phi(\Xi, \alpha \mid \Gamma; \Delta \vdash t\colon \sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi, \alpha} \phi(t)\colon \sigma \quad \Xi \mid \Gamma; \Delta}{\phi(\Xi \mid \Gamma; \Delta \vdash \Lambda\alpha.\, t\colon \prod \alpha.\, \sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \Lambda\alpha.\, \phi(t)\colon \prod \alpha\, .\, \sigma}$$

$$\dfrac{\phi(\Xi \mid \Gamma; \Delta \vdash t\colon \prod \alpha.\, \sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(t)\colon \prod \alpha.\, \sigma \quad \Xi \vdash \tau\colon \mathsf{Type}}{\phi(\Xi \mid \Gamma; \Delta \vdash t(\tau)\colon \sigma[\tau/\alpha]) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(t)(\tau)\colon \sigma[\tau/\alpha]}$$

$$\dfrac{\begin{array}{c}\phi(\Xi \mid \Gamma; \Delta \vdash s\colon !\sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(s)\colon !\sigma \\ \phi(\Xi \mid \Gamma, x\colon \sigma; \Delta' \vdash t\colon \tau) = \Gamma, \Delta', x\colon \sigma \mid \delta' \vdash_{\Xi} \phi(t)\colon \tau\end{array}}{\phi(\Xi \mid \Gamma; \Delta, \Delta' \vdash \mathrm{let}\ !x\ \mathrm{be}\ s\ \mathrm{in}\ t\colon \tau) = \Gamma, \Delta, \Delta' \mid \delta \vee \delta' \vdash_{\Xi} \mathrm{let}\ !x\ \mathrm{be}\ \phi(s)\ \mathrm{in}\ \phi(t)\colon \tau}$$

Figure 5: Inductive definition of ϕ.

- $\phi(t)$ has the same type as t.

In the interpretation defined in Figure 5 we use weakening (as in Lemma 7.2) implicitly in the rules for function application and let-expressions.

We need to show that $\llparenthesis - \rrparenthesis = \llbracket - \rrbracket \circ \phi$ on the closed terms. This is of course done by induction, and for the induction we need to consider open terms. But open terms are interpreted differently in the two interpretations. For an open term t of $\text{PILL}_Y \setminus \otimes$ with free variables

$$x_1 \colon \sigma_1, \dots, x_n \colon \sigma_n ; x'_1 \colon \sigma'_1, \dots, x'_m \colon \sigma'_m$$

$\llbracket t \rrbracket_{\vec{A}}$ is strict map from $\bigotimes_i L\llbracket \sigma_i \rrbracket^d(\vec{A}) \otimes \bigotimes_j \llbracket \sigma'_j \rrbracket^d(\vec{A})$ whereas $\llparenthesis \phi(t) \rrparenthesis_{\vec{A}}$, is a map from $\prod_i \llbracket \sigma_i \rrbracket^d(\vec{A}) \times \prod_j \llbracket \sigma'_j \rrbracket^d(\vec{A})$, which may be strict in some variables and not in others. The induction hypothesis is that for all domains $\vec{A}$, elements $x_i \colon \llbracket \sigma_i \rrbracket^d(\vec{A})$, $x'_j \colon \llbracket \sigma'_j \rrbracket^d(\vec{A})$

$$\llbracket t \rrbracket_{\vec{A}}(\{x_1\} \otimes \ldots \otimes \{x_n\} \otimes x'_1 \otimes \ldots \otimes x'_m) = \llparenthesis \phi(t) \rrparenthesis_{\vec{A}}(x_1, \ldots x_n, x'_1, \ldots, x'_m),$$

where $x \otimes y$ is shorthand for $\eta(x, y)$ where η is the natural transformation from $(-) \times (=)$ to $(-) \otimes (=)$.

We do the induction cases in the order of constructions of Figure 5, except for the case of Y which is the most difficult and is therefore postponed. The case of free variables is trivial.

For function application, suppose the term t has free linear variables $x'_1, \ldots, x'_i$ and u has free linear variables $x'_{i+1}, \ldots x'_m$. We have

$$\begin{gathered}\llbracket t\, u \rrbracket_{\vec{A}}(\{x_1\} \otimes \ldots \otimes \{x_n\} \otimes x'_1 \otimes \ldots \otimes x'_m) = \\ \llbracket t \rrbracket_{\vec{A}}(\{x_1\} \otimes \ldots \{x_n\} \otimes x'_1 \otimes \ldots x'_i)(\llbracket u \rrbracket_{\vec{A}}(\{x_1\} \otimes \ldots \{x_n\} \otimes x'_{i+1} \otimes \ldots x'_m))\end{gathered}$$

and

$$\llparenthesis t\, u \rrparenthesis_{\vec{A}}(\vec{x}, \vec{x}') = \llparenthesis t \rrparenthesis_{\vec{A}}(\vec{x}, x'_1 \ldots, x'_i)(\llparenthesis u \rrparenthesis_{\vec{A}}(\vec{x}, x'_{i+1}, \ldots, x'_m))$$

and so the induction step follows.

For lifted terms, we have by definition $\llparenthesis !t \rrparenthesis_{\vec{A}}(\vec{x}) = \{\llparenthesis t \rrparenthesis_{\vec{A}}(\vec{x})\}$. Let us for simplicity assume that t has exactly one free (intuitionistic) variable of type σ. In the PILL_Y-model, $\llbracket t \rrbracket_{\vec{A}}$ is a map $L(\llbracket \sigma \rrbracket^d(\vec{A})) \multimap \llbracket \tau \rrbracket^d(\vec{A})$ and $\llbracket !t \rrbracket_{\vec{A}}$ is the composite

$$L(\llbracket \sigma \rrbracket^d(\vec{A})) \xrightarrow{\delta} LL(\llbracket \sigma \rrbracket^d(\vec{A})) \xrightarrow{L\llbracket t \rrbracket_{\vec{A}}} L(\tau^d(\vec{A}))$$

So since $\delta(\{x\}) = \{\{x\}\}$, $\llbracket !t \rrbracket_{\vec{A}}(\{x\}) = \{\llbracket t \rrbracket_{\vec{A}}(\{x\})\}$. By induction hypothesis $\llbracket t \rrbracket_{\vec{A}}(\{x\}) = \llparenthesis t \rrparenthesis_{\vec{A}}(x)$, and so the induction step follows.

For the case $t = \Lambda\alpha.\, t'$ we have by induction for all $\vec{A}, \vec{x}, \vec{x}'$ and for all $D \in \mathbf{D}$

$$\llbracket t' \rrbracket_{\vec{A},D}(\bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j) = \llparenthesis t' \rrparenthesis_{\vec{A},D}(\vec{x}, \vec{x}')$$

and so

$$\llbracket t \rrbracket_{\vec{A}}(\bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j) = (\llbracket t' \rrbracket_{\vec{A},D} \bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j)_{D \in \mathbf{D}} = (\llparenthesis t' \rrparenthesis_{\vec{A},D}(\vec{x}, \vec{x}'))_{D \in \mathbf{D}} = \llparenthesis t \rrparenthesis_{\vec{A}}(\vec{x}, \vec{x}').$$

For the case $t = t'(\tau) \colon \sigma[\tau/\alpha]$, we have defined

$$\llbracket t \rrbracket_{\vec{A}}(\bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j) = \llbracket \sigma \rrbracket^d(id_{\vec{A}}, k)((\llbracket t' \rrbracket_{\vec{A}}(\bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j))_D)$$

for some iso $k\colon D \multimap [\![\tau]\!]^d(\vec{A})$, and the $(\![-]\!)$ interpretation is defined likewise, and so, using Lemma 7.13 we get the induction step.

For the case of terms of the form let $!x$ be s in t we assume that we have

$$\Xi \mid \Gamma; \Delta \vdash s\colon !\sigma \qquad \Xi \mid \Gamma, x\colon \sigma; \Delta' \vdash t\colon \tau$$

Assume for simplicity of notation that $\Gamma,\ \Delta, \Delta'$ consists of exactly one variable each. Then

$$(\![\text{let } !x \text{ be } s \text{ in } t]\!)_{\vec{A}}(x, y, z) = \widehat{(\![t]\!)}_{\vec{A}}(x, (\![s]\!)_{\vec{A}}(x, y), z)$$

where

$$\widehat{(\![t]\!)}_{\vec{A}}\colon (\![\Gamma]\!)^d(\vec{A}) \times L((\![\sigma]\!)^d(\vec{A})) \times (\![\Delta]\!)^d(\vec{A})) \to (\![\tau]\!)^d(\vec{A})$$

is the unique extension of

$$(\![t]\!)_{\vec{A}}\colon (\![\Gamma]\!)^d(\vec{A}) \times (\![\sigma]\!)^d(\vec{A}) \times (\![\Delta]\!)^d(\vec{A})) \to (\![\tau]\!)^d(\vec{A})$$

which is strict in the second variable. Since by induction, for all u, v, w

$$[\![t]\!]_{\vec{A}}(\{u\} \otimes \{v\} \otimes w) = (\![t]\!)_{\vec{A}}(u, v, w)$$

we get that also for all v'

$$[\![t]\!]_{\vec{A}}(\{u\} \otimes v' \otimes w) = \widehat{(\![t]\!)}_{\vec{A}}(u, v', w),$$

since $[\![t]\!]_{\vec{A}}$ is strict. Thus

$$\begin{aligned} (\![\text{let } !x \text{ be } s \text{ in } t]\!)_{\vec{A}}(x, y, z) &= [\![t]\!]_{\vec{A}}(\{x\} \otimes (\![s]\!)_{\vec{A}}(x, y) \otimes z) = \\ [\![t]\!]_{\vec{A}}(\{x\} \otimes [\![s]\!]_{\vec{A}}(\{x\} \otimes y) \otimes z) &= [\![\text{let } !x \text{ be } s \text{ in } t]\!]_{\vec{A}}(\{x\} \otimes y \otimes z) \end{aligned}$$

For fixed points, we have defined $[\![Y]\!] = (\mathit{fix}_D)_{D \in \mathbf{D}}$ in Lemma 5.7. In this definition, we have identified objects of $!D \multimap D$ with non-strict maps $D \to D$, and so strictly speaking, we should have defined $[\![Y]\!] = (a_D)_{D \in \mathbf{D}}$, where $a_D\colon !(!D \multimap D) \multimap D$ is the unique strict map such that $a_D(\{f\}) = \mathit{fix}_D(\hat{f})$ where $\hat{f}$ is the set theoretic map $D \to D$ corresponding to $f\colon !D \multimap D$.

To compute $(\![\phi(Y)]\!)$, consider first

$$(\![f :_1 !(!\alpha \multimap \alpha), x :_0 \alpha \vdash_\alpha \text{let } !u \text{ be } f \text{ in } u(!x)]\!)_D$$

which is the unique extension of the map

$$(f\colon LD \multimap D, x\colon D) \mapsto f(\{x\})$$

to a map $L(LD \multimap D) \times D \to D$ which is strict in the first variable. So

$$(\![f :_1 !(!\alpha \multimap \alpha) \vdash_\alpha \text{rec } x :_0 \alpha.\, \text{let } !u \text{ be } f \text{ in } u(!x)]\!)_D$$

is the unique extension of $f\colon LD \multimap D \mapsto \mathit{fix}_D(\hat{f})$ to a strict map $L(LD \multimap D) \multimap D$, and thus we conclude

$$\begin{gathered} (\![\phi(Y)]\!) = (\![\Lambda\alpha.\, \lambda f\colon !(!\alpha \multimap \alpha).\, \text{rec } x :_0 !\alpha.\, \text{let } !u \text{ be } f \text{ in } u(x)]\!) = \\ (a_D)_D = [\![Y]\!] \end{gathered}$$

For the last statement of the theorem, suppose $t\colon \sigma \multimap \tau$, $u\colon \tau \multimap \omega$ are terms of $\mathrm{PILL}_Y \setminus \otimes$. Then

$$\begin{gathered} \phi(u \circ t) = \phi(\lambda^\circ x\colon \sigma.\, u(t(x))) = \lambda x\colon \sigma.\, \phi(u(t(x))) = \\ \lambda x\colon \sigma.\, \phi(u)\phi(t)(x) = \phi(u) \circ \phi(t). \end{gathered}$$

□

The restriction of the translation to PILL$_Y \setminus \otimes$ in Theorem 7.12 is not essential.

Proposition 7.14. *There exists a translation ψ of PILL$_Y$ into PILL$_Y \setminus \otimes$ such that for any parametric PILL$_Y$-model X the diagram*

$$\begin{tikzcd}
PILL_Y \arrow[rr, "\psi"] \arrow[dr, "{[\![-]\!]}"'] & & PILL_Y \setminus \otimes \arrow[dl, "{[\![-]\!]}"] \\
& X &
\end{tikzcd}$$

commutes up to natural isomorphism. To be more precise, there exists a family of isomorphisms $f_\sigma \colon [\![\sigma]\!] \to [\![\psi(\sigma)]\!]$ indexed by closed types of PILL$_Y$, such that for each closed term $t \colon \sigma \multimap \tau$ of closed types, the diagram

$$\begin{tikzcd}
{[\![\sigma]\!]} \arrow[r, "f_\sigma"] \arrow[d, "{[\![t]\!]}"'] & {[\![\psi(\sigma)]\!]} \arrow[d, "{[\![\psi(t)]\!]}"] \\
{[\![\tau]\!]} \arrow[r, "f_\tau"] & {[\![\psi(\tau)]\!]}
\end{tikzcd}$$

commutes. Furthermore, the restriction of ψ to PILL$_Y \setminus \otimes$ is the identity, and for $\alpha \vdash \sigma(\alpha)$ a type in PILL$_Y \setminus \otimes$, $\psi(\sigma(\tau)) = \sigma(\psi(\tau))$. The translation is functorial in the sense that if $t \colon \sigma \multimap \tau$ and $s \colon \tau \multimap \omega$ are closed terms of closed types, then $\psi(s \circ t) = \psi(s) \circ \psi(t)$.

Of course, the core of the translation of Proposition 7.14 is the well known consequences of parametricity

$$\begin{array}{rcl}
\sigma \otimes \tau & \cong & \prod \alpha. (\sigma \multimap \tau \multimap \alpha) \multimap \alpha, \\
I & \cong & \prod \alpha. \alpha \multimap \alpha.
\end{array}$$

The interesting part of the proposition is that it is a consequence of parametricity that all *terms* of PILL$_Y$ expressible using $\otimes$, I and let-expressions, $\star$ etc. can also be expressed in the smaller language PILL$_Y \setminus \otimes$. For the proof of this proposition we refer to Appendix A.

Corollary 7.15. *There exists a translation of PILL$_Y$ into Lily$_{strict}$ which commutes with interpretation up to natural isomorphism. The translation is an extension of the translation of Theorem 7.12.*

Proof. This follows from Theorem 7.12 and Proposition 7.14. □

Recall that in [3] an equality theory on terms of PILL$_Y$ called external equality is defined. External equality is basically equality up to β, η- conversion.

Lemma 7.16. *The translation of PILL$_Y$ into Lily$_{strict}$ maps externally equivalent terms to ground contextually equivalent terms.*

Proof. Externally equal terms of PILL$_Y$ are interpreted as equal terms in the model. Since the translation commutes with interpretation into the model, by adequacy (Theorem 7.9), the translated terms are ground contextually equal. □

7.3 Consequences of parametricity for Lily$_{\text{strict}}$

We end this section by showing how to use Theorem 7.12, computational adequacy of the interpretation $(\![-]\!)$ and the results of [3] to obtain consequences of parametricity for the language Lily$_{\text{strict}}$.

Consider the category whose objects are the closed types of Lily$_{\text{strict}}$ and whose morphisms from σ to τ are closed terms of type $\sigma \multimap \tau$ of Lily$_{\text{strict}}$ identified up to ground contextual equivalence. We call this category $\mathbf{Lily}$.

Corollary 7.17. *For all closed types σ of Lily$_{strict}$, the objects σ and $\prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ are isomorphic as objects of* $\mathbf{Lily}$.

Proof. The maps of the isomorphism $\sigma \cong \prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ are defined as in [3]. Applying ϕ of Theorem 7.12 to these maps, we obtain morphisms of the right types in $\mathbf{Lily}$. In [3] it is shown that the interpretations of these maps into the parametric model are isomorphisms, and so by adequacy, we obtain the desired result. □

As always, type expressions $\alpha \vdash \sigma(\alpha)$ in Lily$_{\text{strict}}$ for which α only appears positively in σ induce functors on $\mathbf{Lily}$.

Theorem 7.18. *All functors* $\mathbf{Lily} \to \mathbf{Lily}$ *induced by types $\sigma(\alpha)$ in Lily$_{strict}$ have initial algebras and final coalgebras.*

Proof. First we notice that it is easy to see that the functorial interpretation of types commutes with ϕ.

We define the initial algebra by applying the translation of Theorem 7.12 to $\mathit{in}\colon \sigma(\mu\alpha.\, \sigma(\alpha)) \multimap \mu\alpha.\, \sigma(\alpha)$. To show that this defines a weak initial algebra, consider $\phi(\mathit{fold})$, that is, ϕ applied to the term that takes an algebra and produces a map from the initial algebra. Since

$$\Lambda\alpha.\, \lambda^\circ f\colon \sigma(\alpha) \multimap \alpha.\, f \circ \sigma(\mathit{fold}\,\alpha\,!f) = \Lambda\alpha.\, \lambda^\circ f\colon \sigma(\alpha) \multimap \alpha.\, (\mathit{fold}\,\alpha\,!f) \circ \mathit{in}$$

in PILL$_Y$, using Lemma 7.16 it is easy to see, that this defines a weak initial algebra.

Suppose we have two maps g, h out of this initial algebra definable in Lily$_{\text{strict}}$. Then $(\![g]\!), (\![h]\!)$ are maps out of $[\![\mathit{in}]\!]$ in the model. But since we know that $[\![\mathit{in}]\!]$ is an initial algebra in the model, $(\![h]\!) = (\![g]\!)$, and so by adequacy $h \equiv_{\text{gnd}} g$.

The proof for final coalgebras is exactly the same. □

Theorem 7.19. *For all types $\alpha \vdash \sigma(\alpha)\colon$ Type of Lily$_{strict}$, there exists a closed type τ of Lily$_{strict}$ such that τ and $\sigma(\tau)$ are isomorphic as objects of* $\mathbf{Lily}$.

Proof. We can define τ and the isomorphisms $\tau \cong \sigma(\tau)$ in pure PILL$_Y$. Now, apply the translation of Corollary 7.15 to this isomorphism. From this we get a type τ' and morphisms $\sigma(\tau') \multimap \tau'$, $\tau' \multimap \sigma(\tau')$ definable in Lily$_{\text{strict}}$. By functoriality of ϕ, the interpretations of both compositions of the two maps are identities in the model. Thus, by adequacy, the two compositions are ground contextual equivalent to the identity, and thus τ' and $\sigma(\tau')$ are isomorphic in $\mathbf{Lily}$. □

8 Conclusion

We have constructed an LAPL-structure based on the interpretation of $\text{Lily}_{\text{strict}}$ into models of synthetic domain theory presented in [14]. Comparing this with the concrete domain theoretic LAPL-structure of [8], the completion process for LAPL-structures of [7], and the LAPL-structure based on the operational semantics of Lily [1] under development at the moment of writing, this shows that the notion of LAPL-structure is general enough to handle very different kinds of parametric models.

The LAPL-structure also provides formal proof of the consequences of parametricity, such as the existence of recursive types, for the interpretation of [14].

Using adequacy of the interpretation of $\text{Lily}_{\text{strict}}$, we have shown consequences of parametricity for $\text{Lily}_{\text{strict}}$ up to ground contextual equivalence. These consequences include encodings of inductive, coinductive and recursive types.

A Tensor products in parametric LAPL-structures

In this appendix we prove the following Proposition.

Proposition A.1. *There exists a translation ψ of PILL$_Y$ into PILL$_Y \setminus \otimes$ such that for any parametric PILL$_Y$-model X the diagram*

$$\begin{tikzcd}
\mathit{PILL}_Y \arrow[rr, "\psi"] \arrow[dr, "\llbracket - \rrbracket"'] & & \mathit{PILL}_Y \setminus \otimes \arrow[dl, "\llbracket - \rrbracket"] \\
& X &
\end{tikzcd}$$

commutes up to natural isomorphism. To be more precise, there exists a family of isomorphisms $f_\sigma \colon \llbracket \sigma \rrbracket \to \llbracket \psi(\sigma) \rrbracket$ indexed by closed types, such that for each closed term $t \colon \sigma \multimap \tau$ of closed types, the diagram

$$\begin{tikzcd}
\llbracket \sigma \rrbracket \arrow[r, "f_\sigma"] \arrow[d, "\llbracket t \rrbracket"'] & \llbracket \psi(\sigma) \rrbracket \arrow[d, "\llbracket \psi(t) \rrbracket"] \\
\llbracket \tau \rrbracket \arrow[r, "f_\tau"] & \llbracket \psi(\tau) \rrbracket
\end{tikzcd}$$

commutes. Furthermore, ψ is the identity on PILL$_Y \setminus \otimes$ and for $\alpha \vdash \sigma(\alpha)$ a type in PILL$_Y \setminus \otimes$, $\psi(\sigma(\tau)) = \sigma(\psi(\tau))$. The translation is functorial in the sense, that if $t \colon \sigma \multimap \tau$ and $s \colon \tau \multimap \omega$ are closed terms of closed types, then $\psi(s \circ t) = \psi(s) \circ \psi(t)$.

We will prove this Proposition working in Abadi & Plotkin's logic, i.e., in stead of proving Proposition A.1 we prove the Proposition A.2 and Lemma A.7 below.

Proposition A.2. *There exists a translation ψ of PILL$_Y$ into PILL$_Y \setminus \otimes$ and maps $f_\sigma \colon \sigma \multimap \psi(\sigma)$ indexed by closed types, such that, assuming parametricity, for each closed term $t \colon \sigma \multimap \tau$ of closed types, the diagram*

$$\begin{tikzcd}
\sigma \arrow[r, "f_\sigma", multimap] \arrow[d, "t"', multimap] & \psi(\sigma) \arrow[d, "\psi(t)", multimap] \\
\tau \arrow[r, "f_\tau", multimap] & \psi(\tau)
\end{tikzcd}$$

commutes up to internal equality.

We define ψ as a translation defined on all types and all terms of PILL$_Y$ as described in Figure 6.

Lemma A.3. *Suppose*

$$\Xi \mid \vec{x} \colon \vec{\sigma}; \vec{y} \colon \vec{\sigma}' \vdash t \colon \tau$$

is a term of PILL$_Y$. Then

$$\Xi \mid \vec{x} \colon \vec{\psi(\sigma)}; \vec{y} \colon \vec{\psi(\sigma')} \vdash \psi(t) \colon \psi(\tau)$$

is a typing judgement of PILL$_Y \setminus \otimes$.

Proof. Easy induction on the structure of t. □

$$\psi(\alpha) = \alpha \qquad \psi(\sigma \multimap \tau) = \psi(\sigma) \multimap \psi(\tau) \qquad \psi(\textstyle\prod \alpha.\, \sigma) = \prod \alpha.\, \psi(\sigma)$$

$$\psi(!\sigma) = !\psi(\sigma) \qquad \psi(I) = \textstyle\prod \alpha.\, \alpha \multimap \alpha \text{ (for } \alpha \text{ fresh variable)}$$

$$\psi(\sigma \otimes \tau) = \textstyle\prod \alpha.\, (\psi(\sigma) \multimap \psi(\tau) \multimap \alpha) \multimap \alpha \text{ (for } \alpha \text{ fresh variable)}$$

$$\psi(x) = x\text{, for } x \text{ variable} \qquad \psi(\star) = \Lambda\alpha.\, \lambda^\circ x \colon \alpha.\, x \qquad \psi(Y) = Y$$

$$\psi(t\, u) = \psi(t)\, \psi(u) \qquad \psi(\lambda^\circ x \colon \sigma.\, t) = \lambda^\circ x \colon \psi(\sigma).\, \psi(t)$$

$$\psi(t \otimes u \colon \sigma \otimes \tau) = \Lambda\alpha.\, \lambda^\circ h \colon \psi(\sigma) \multimap \psi(\tau) \multimap \alpha.\, h\, \psi(t)\, \psi(u) \qquad \psi(!t) = !\psi(t)$$

$$\psi(\Lambda\alpha.\, t) = \Lambda\alpha.\, \psi(t) \qquad \psi(t(\sigma)) = \psi(t)(\psi(\sigma))$$

$$\psi(\text{let } x \otimes y \colon \sigma \otimes \sigma' \text{ be } t \text{ in } u \colon \tau) = \psi(t)\, \psi(\tau)\, (\lambda^\circ x \colon \psi(\sigma).\, \lambda^\circ y \colon \psi(\sigma').\, \psi(u))$$

$$\psi(\text{let } !x \text{ be } t \text{ in } u) = \text{let } !x \text{ be } \psi(t) \text{ in } \psi(u) \qquad \psi(\text{let } \star \text{ be } t \text{ in } u \colon \tau) = \psi(t)\, \psi(\tau)\, \psi(u)$$

Figure 6: Inductive definition of ψ.

Lemma A.4. *Suppose σ, τ are types of PILL$_Y$. The map*

$$i_{\sigma,\tau} \colon \sigma \otimes \tau \multimap \textstyle\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

defined as

$$\lambda^\circ y \colon \sigma \otimes \tau.\, \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, \textit{let } x \otimes x' \textit{ be } y \textit{ in } h\, x\, x'.$$

is natural in σ, τ, *i.e, if* $k \colon \sigma \multimap \sigma'$, $l \colon \tau \multimap \tau'$ *then*

$$i_{\sigma',\tau'} \circ k \otimes l = (\textstyle\prod \alpha.\, (k \multimap l \multimap \alpha) \multimap \alpha) \circ i_{\sigma,\tau}.$$

Using parametrictiy, one can show that $i_{\sigma,\tau}$ is an isomorphism up to internal equality. The terms

$$\lambda^\circ x \colon \textstyle\prod \alpha.\, \alpha \multimap \alpha.\, x\, I\, \star \colon (\prod \alpha.\, \alpha \multimap \alpha) \multimap I$$
$$\lambda^\circ x \colon I.\, \Lambda\alpha.\, \lambda^\circ y \colon \alpha.\, \textit{let } \star \textit{ be } x \textit{ in } y \colon I \multimap (\textstyle\prod \alpha.\, \alpha \multimap \alpha)$$

can be shown to be each others inverses up to internal equality using parametricity.

Proof. We show that $i_{\sigma,\tau}$ is natural in σ, τ. The rest of the proof can be found in [3].
Suppose $k \colon \sigma \multimap \sigma'$, $l \colon \tau \multimap \tau'$. Then for $y \colon \sigma \otimes \tau$,

$$(k \otimes l)(y) = \text{let } z \otimes w \text{ be } y \text{ in } k(z) \otimes l(w).$$

So $i_{\sigma',\tau'}((k \otimes l)(y))$ is

$$\begin{gathered}\Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, \text{let } x \otimes x' \text{ be } (k \otimes l)(y) \text{ in } h\, x\, x' = \\ \Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, \text{let } z \otimes w \text{ be } y \text{ in } (\text{let } x \otimes x' \text{ be } k(z) \otimes l(w) \text{ in } h\, x\, x') = \\ \Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, \text{let } z \otimes w \text{ be } y \text{ in } h\, (k(z))\, (l(w))\end{gathered}$$

On the other hand

$$\begin{gathered}(\textstyle\prod \alpha.\, (k \multimap h \multimap \alpha) \multimap \alpha)(i_{\sigma,\tau}(y)) = \\ \Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, (i_{\sigma,\tau}(y))\, \alpha\, ((k \multimap l \multimap \alpha)(h)) = \\ \Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, \text{let } x \otimes x' \text{ be } y \text{ in } (k \multimap l \multimap \alpha)(h)\, x\, x' = \\ \Lambda\alpha.\, \lambda^\circ h \colon \sigma' \multimap \tau' \multimap \alpha.\, \text{let } x \otimes x' \text{ be } y \text{ in } h\, (k(x))\, (l(x')).\end{gathered}$$

□

We define terms $f_\sigma \colon \sigma \multimap \psi(\sigma)$ and $g_\sigma \colon \psi(\sigma) \multimap \sigma$ of PILL$_Y$ for *all* types (not just the closed types) σ of PILL$_Y$ as described in Figure 7. Basically f_σ, g_σ are defined inductively, by using the functorial interpretations of type constructors $\multimap, !, \prod \alpha. (-)$. The induction step for $\sigma \otimes \tau$ is obtained using the isomorphism of Lemma A.4.

$$f_\alpha = id_\alpha \qquad f_{\sigma \multimap \tau} = \lambda^\circ t \colon \sigma \multimap \tau.\, f_\tau \circ t \circ g_\sigma \qquad f_{!\sigma} = !f_\sigma$$

$$f_{\prod \alpha.\sigma} = \lambda^\circ t \colon (\prod \alpha.\, \sigma).\, \Lambda\alpha.\, f_\sigma(t\,\alpha)$$

$$f_I = \lambda^\circ t \colon I.\, \Lambda\alpha.\, \lambda^\circ u \colon \alpha.\, \text{let } \star \text{ be } t \text{ in } u.$$

$$f_{\sigma\otimes\tau} = i_{\psi(\sigma),\psi(\tau)} \circ f_\sigma \otimes f_\tau$$

$$g_\alpha = id_\alpha \qquad g_{\sigma \multimap \tau} = \lambda^\circ t \colon \psi(\sigma) \multimap \psi(\tau).\, g_\tau \circ t \circ f_\sigma \qquad g_{!\sigma} = !g_\sigma$$

$$g_{\prod \alpha.\sigma} = \lambda^\circ t \colon (\prod \alpha.\, \sigma).\, \Lambda\alpha.\, g_\sigma(t\,\alpha)$$

$$g_I = \lambda^\circ t \colon \prod \alpha.\, \alpha \multimap \alpha.\, t\, I \star$$

$$g_{\sigma\otimes\tau} = g_\sigma \otimes g_\tau \circ i^{-1}_{\psi(\sigma),\psi(\tau)}$$

Figure 7: Inductive definitions of f, g.

Before we prove Lemma A.2 we need to prove a series of lemmas.

Lemma A.5. *For all types $\vec{\alpha} \vdash \sigma$ the maps f_σ, g_σ are each others inverses.*

Proof. Simple induction over the structure of σ. □

Lemma A.6. *Suppose* $\Xi, \alpha \vdash \sigma \colon \mathsf{Type}$ *and* $\Xi \vdash \tau \colon \mathsf{Type}$*. Then* $\psi(\sigma[\tau/\alpha]) = \psi(\sigma)[\psi(\tau)/\alpha]$.

Proof. Easy induction on the structure of σ. □

Lemma A.7. *ψ is the identity on PILL$_Y \setminus \otimes$. If $\alpha \vdash \sigma(\alpha)$ is a type in PILL$_Y \setminus \otimes$ and τ is any type of PILL$_Y$, then $\psi(\sigma(\tau)) = \sigma(\psi(\tau))$.*

Proof. Easy induction on σ. □

Lemma A.8. *If $\Xi \vdash \sigma$ is a type in PILL$_Y \setminus \otimes$ then f_σ is the identity.*

Proof. Easy induction on the structure of σ. □

The next few pages until Lemma A.12 we prove a series of lemmas describing the behavior of f_σ with respect to reindexing. Basically what makes this difficult is that since $f_\alpha = id_\alpha$ if α is a type variable, we cannot have $f_\sigma[\tau/\alpha] = f_{\sigma[\tau/\alpha]}$.

Suppose $\Xi, \alpha, \beta \vdash \sigma(\alpha, \beta)$ is a type in PILL$_Y$ in which α occurs only negatively and β only positively. For $\Xi \vdash f \colon \tau' \multimap \tau, g \colon \omega \multimap \omega'$ we write

$$\Xi \vdash \sigma(f, g) \colon \sigma(\tau, \omega) \multimap \sigma(\tau', \omega')$$

for the well known functorial interpretation of σ. Recall that this functorial interpretation of types is given by a term

$$M\colon \prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to (\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta')).$$

For details, see [3].

Lemma A.9. *[Groupoid-action Lemma] Suppose $\Xi, \alpha, \beta \vdash \sigma(\alpha, \beta)$ is a type in PILL$_Y$ in which α occurs only negatively and β only possitively. Suppose further that $f\colon \tau \multimap \tau'$ is an isomorphism, i.e., there exists a term f^{-1} which is an inverse to f up to internal equality. Then*

$$\Xi \mid - \mid - \vdash \sigma[\mathit{eq}_\Xi, \langle f \rangle, \langle f \rangle] \equiv \langle \sigma(f^{-1}, f) \rangle$$

Proof. Suppose $f\colon \tau \multimap \tau'$. Consider first the two commutative diagrams

$$\begin{tikzcd}
\tau \arrow[r, multimap, "id_\tau"] \arrow[d, multimap, "f"'] & \tau \arrow[d, multimap, "id_\tau"] \\
\tau' \arrow[r, multimap, "f^{-1}"] & \tau
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau \arrow[r, multimap, "id_\tau"] \arrow[d, multimap, "id_\tau"'] & \tau \arrow[d, multimap, "f"] \\
\tau \arrow[r, multimap, "f"] & \tau'.
\end{tikzcd}$$

These diagrams imply that

$$(id_\tau, f^{-1})\colon \langle f \rangle \multimap \mathit{eq}_\tau,$$
$$(id_\tau, f)\colon \mathit{eq}_\tau \multimap \langle f \rangle.$$

Instantiating the parametricity schema for the term

$$M\colon \prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to (\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta'))$$

giving the functorial interpretation of the type σ in the case

$$\alpha' = \langle f \rangle, \quad \alpha = \mathit{eq}_\tau, \quad \beta = \mathit{eq}_\tau, \quad \beta' = \langle f \rangle$$

we get

$$\sigma(id_\tau, id_\tau)(\sigma[\mathit{eq}_\tau, \mathit{eq}_\tau] \multimap \sigma[\langle f \rangle, \langle f \rangle])\sigma(f^{-1}, f).$$

which using identity extension and functoriality of σ gives

$$(id_{\sigma(\tau,\tau)}, \sigma(f^{-1}, f))\colon \mathit{eq}_{\sigma(\tau,\tau)} \multimap \sigma[\langle f \rangle, \langle f \rangle].$$

Since $\forall x\colon \sigma(\tau, \tau).\, x(\mathit{eq}_{\sigma(\tau,\tau)})x$ we have $\forall x\colon \sigma(\tau, \tau).\, x\sigma[\langle f \rangle, \langle f \rangle]\sigma(f^{-1}, f)x$. So $x\langle \sigma(f^{-1}, f) \rangle y$ implies $x\sigma[\langle f \rangle, \langle f \rangle]y$, proving the first direction of the lemma.

To show the other direction, we proceed as before. Consider first the two commutative diagrams

$$\begin{tikzcd}
\tau' \arrow[r, multimap, "f^{-1}"] \arrow[d, multimap, "id_{\tau'}"'] & \tau \arrow[d, multimap, "f"] \\
\tau' \arrow[r, multimap, "id_{\tau'}"] & \tau'
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau \arrow[r, multimap, "f"] \arrow[d, multimap, "f"'] & \tau \arrow[d, multimap, "id_{\tau'}"] \\
\tau' \arrow[r, multimap, "id_{\tau'}"] & \tau'.
\end{tikzcd}$$

Thus,

$$(f^{-1}, id_{\tau'})\colon \mathit{eq}_{\tau'} \multimap \langle f \rangle,$$
$$(f, id_{\tau'})\colon \langle f \rangle \multimap \mathit{eq}_{\tau'}.$$

So by parametricity

$$(\sigma(f^{-1}, f), id_{\tau'})\colon \sigma[\langle f \rangle, \langle f \rangle] \multimap \mathit{eq}_{\tau'}.$$

Suppose now $x\sigma[\langle f \rangle, \langle f \rangle]y$. Then $\sigma(f^{-1}, f)(x) = y$, i.e., $x\langle \sigma(f^{-1}, f) \rangle y$ as desired. □

Lemma A.10. *Suppose* $\Xi, \alpha, \beta \vdash \sigma(\Xi, \alpha, \beta)$ *is a type in PILL*$_Y \setminus \otimes$ *in which* α *occurs only negatively and* β *only positively. Suppose further that* $\Xi \vdash \tau \colon \mathsf{Type}$ *is any type. Then*

$$f_{\sigma[\tau/\alpha,\tau/\beta]} = \sigma(g_\tau, f_\tau)$$

Proof. The lemma is proved simultaneously with the statement

$$g_{\sigma[\tau/\alpha,\tau/\beta]} = \sigma(f_\tau, g_\tau)$$

by structural induction on σ. The base cases of σ being a variable are trivial. The case $\sigma = !\sigma'$ is clear from the fact that $f_{!\sigma} = !f_\sigma$ and likewise for g. The case of $\sigma = \prod \alpha'. \sigma'$ is proved likewise.

The case of $\sigma = \sigma' \multimap \sigma''$ is the most interesting. The calculation is

$$f_{(\sigma' \multimap \sigma'')[\tau/\alpha,\tau/\beta]} = \lambda^\circ h \colon (\sigma' \multimap \sigma'')[\tau/\alpha, \tau/\beta].\, f_{\sigma''[\tau/\alpha,\tau/\beta]} \circ h \circ g_{\sigma'[\tau/\alpha,\tau/\beta]}$$

which by induction is equal to

$$\lambda^\circ h \colon (\sigma' \multimap \sigma'')[\tau/\alpha, \tau/\beta].\, \sigma''(g_\tau, f_\tau) \circ h \circ \sigma'(f_\tau, g_\tau) = (\sigma' \multimap \sigma'')(g_\tau, f_\tau).$$

□

Lemma A.11. *Suppose* $\Xi, \alpha \vdash \sigma$ *is a type in PILL*$_Y \setminus \otimes$ *and* $\Xi \vdash \tau$ *is any type. Then the relation* $\sigma[eq_\Xi, \langle f_\tau \rangle]$ *is equivalent to the graph of* $f_{\sigma[\tau/\alpha]}$*. In particular, for any type* $\Xi, \alpha \vdash \sigma$*, the relation* $\psi(\sigma)[eq_\Xi, \langle f_\tau \rangle]$ *is equivalent to the graph of* $f_{\psi(\sigma)[\tau/\alpha]}$

Proof. We first write σ as $\Xi, \alpha, \beta \vdash \sigma(\Xi, \alpha, \beta)$ where we have split the appearences of α in σ into negative (α's) and positive (β's). By Lemma A.9 we know that

$$\sigma[eq_\Xi, \langle f_\tau \rangle, \langle f_\tau \rangle] \equiv \langle \sigma(id_\Xi, g_\tau, f_\tau) \rangle$$

which by Lemma A.10 is equivalent to the graph of $f_{\sigma[\tau/\alpha,\tau/\beta]}$ as desired. □

Lemma A.12. *Suppose* $\Xi, \alpha \vdash \sigma \colon \mathsf{Type}$ *and* $\Xi \vdash \tau \colon \mathsf{Type}$*. Then* $f_{\sigma[\tau/\alpha]} = f_{\psi(\sigma)[\tau/\alpha]} \circ f_\sigma[\tau/\alpha]$.

Proof. We prove this by induction on the structure of σ. The base cases of $\sigma = \alpha$ and $\sigma = \beta$ for $\beta \in \Xi$ are trivial. The case of $\sigma = I$ is also trivial since $f_{\psi(I)}$ is the identity by Lemma A.8.

The induction step for $\sigma = !\sigma'$ is simply using the fact that $f_{!\sigma'} = !f_{\sigma'}$ and $\psi(!\sigma') = !\psi(\sigma')$. We get

$$f_{!\sigma'[\tau/\alpha]} = !f_{\sigma'[\tau/\alpha]} = !f_{\psi(\sigma')[\tau/\alpha]} \circ !f_{\sigma'}[\tau/\alpha] = f_{\psi(!\sigma')[\tau/\alpha]} \circ f_{!\sigma'}[\tau/\alpha]$$

The induction step for $\sigma = \prod \beta. \sigma'$ is proved likewise.

Suppose $\sigma = \sigma' \multimap \sigma''$. Notice first, that since f and g are each others inverses, the induction hypothesis implies that

$$g_{\sigma'[\tau/\alpha]} = g_{\sigma'}[\tau/\alpha] \circ g_{\psi(\sigma')[\tau/\alpha]}$$

and so if $x \colon \sigma'[\tau/\alpha] \multimap \sigma''[\tau/\alpha]$,

$$f_{(\sigma' \multimap \sigma'')[\tau/\alpha]}(x) = f_{\sigma''[\tau/\alpha]} \circ x \circ g_{\sigma'[\tau/\alpha]}$$

which by the induction hypothesis equals

$$f_{\psi(\sigma'')[\tau/\alpha]} \circ f_{\sigma''}[\tau/\alpha] \circ x \circ g_{\sigma'}[\tau/\alpha] \circ g_{\psi(\sigma')[\tau/\alpha]} = f_{\psi(\sigma')[\tau/\alpha] \multimap \psi(\sigma'')[\tau/\alpha]} \circ f_{\sigma' \multimap \sigma''}[\tau/\alpha](x)$$

and we conclude that

$$f_{(\psi(\sigma') \multimap \psi(\sigma''))[\tau/\alpha]} \circ f_{\sigma' \multimap \sigma''}[\tau/\alpha] = f_{(\sigma' \multimap \sigma'')[\tau/\alpha]}.$$

Finally we consider the case of $\sigma = \sigma' \otimes \sigma''$. Denoting as usual, for any pair of types ω, ω' by $i_{\omega,\omega'}$ the isomorphism

$$\omega \otimes \omega' \multimap (\prod \beta.\, (\omega \multimap \omega' \multimap \beta) \multimap \beta)$$

we have, using Lemma A.4

$$f_{\sigma \otimes \sigma'[\tau/\alpha]} = (\prod \beta.\, (f_{\sigma[\tau/\alpha]} \multimap f_{\sigma'[\tau/\alpha]} \multimap \beta) \multimap \beta) \circ i_{\sigma[\tau/\alpha],\sigma'[\tau/\alpha]}$$

which by induction is equal to

$$\begin{gathered} (\prod \beta.\, (f_{\psi(\sigma)[\tau/\alpha]} \multimap f_{\psi(\sigma')[\tau/\alpha]} \multimap \beta) \multimap \beta) \circ \\ (\prod \beta.\, (f_\sigma[\tau/\alpha] \multimap f_{\sigma'}[\tau/\alpha] \multimap \beta) \multimap \beta) \circ i_{\sigma[\tau/\alpha],\sigma'[\tau/\alpha]} = \\ (\prod \beta.\, (f_{\psi(\sigma)[\tau/\alpha]} \multimap f_{\psi(\sigma')[\tau/\alpha]} \multimap \beta) \multimap \beta) \circ f_{\sigma \otimes \sigma'}[\tau/\alpha] = \\ f_{\psi(\sigma \otimes \sigma')[\tau/\alpha]} \circ f_{\sigma \otimes \sigma'}[\tau/\alpha] \end{gathered}$$

as desired. □

The next lemma is basically the induction step for type application for the proof of Proposition A.2. Notice that for this proof parametricity is crucial.

Lemma A.13. *Suppose* $\Xi, \alpha \vdash \sigma \colon \mathsf{Type}$ *and* $\Xi \vdash \tau \colon \mathsf{Type}$. *Then*

$$\begin{tikzcd} \prod \alpha.\, \sigma \arrow[r, "\mathit{app}_\tau", -o] \arrow[d, "f_{\prod \alpha.\sigma}"', -o] & \sigma[\tau/\alpha] \arrow[d, "f_{\sigma[\tau/\alpha]}", -o] \\ \prod \alpha.\, \psi(\sigma) \arrow[r, "\mathit{app}_{\psi(\tau)}", -o] & \psi(\sigma[\tau/\alpha]) \end{tikzcd}$$

commutes, where app_τ *is the map* $\lambda^\circ x \colon \prod \alpha.\, \sigma.\, x\, \tau$ *and* $\mathit{app}_{\psi[\tau/\alpha]}$ *defined likewise.*

Proof. Since

$$\mathit{app}_\tau \circ f_{\prod \alpha.\sigma}(x) = \mathit{app}_\tau(\Lambda\alpha.\, f_\sigma(x\, \alpha)) = f_\sigma[\tau/\alpha](x\, \tau)$$

we have $\mathit{app}_\tau \circ f_{\prod \alpha.\sigma} = f_\sigma[\tau/\alpha] \circ \mathit{app}_\tau$. Parametricity tells us that for all $x \colon \prod \alpha.\, \psi(\sigma)$,

$$\psi(\sigma)[\mathit{eq}_\Xi, \langle f_\tau \rangle](\mathit{app}_\tau(x), \mathit{app}_{\psi(\tau)}(x))$$

By Lemma A.11 we thus conclude

$$f_{\psi(\sigma)[\tau/\alpha]} \circ \mathit{app}_\tau = \mathit{app}_{\psi(\tau)}$$

Now, using Lemma A.12 we get

$$\mathit{app}_{\psi(\tau)} \circ f_{\prod \alpha.\sigma} = f_{\psi(\sigma)[\tau/\alpha]} \circ \mathit{app}_\tau \circ f_{\prod \alpha.\sigma} = f_{\psi(\sigma)[\tau/\alpha]} \circ f_\sigma[\tau/\alpha] \circ \mathit{app}_\tau = f_{\sigma[\tau/\alpha]} \circ \mathit{app}_\tau$$

as desired. □

Proof of Proposition A.2. The proof is by induction on the term t, but for that to go through, we need a stronger induction hypothesis, considering open terms as well. The induction hypothesis will be the following. Suppose we have an open term

$$\Xi \mid \vec{x}\colon \vec{\sigma}, \vec{y}\colon \vec{\sigma}' \vdash t(\vec{x},\vec{y})\colon \tau$$

then

$$\Xi \mid \vec{x}\colon \vec{\sigma}, \vec{y}\colon \vec{\sigma}' \vdash f_\tau(t(\vec{x},\vec{y})) =_{\psi(\tau)} \psi(t)(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})).$$

where $f_{\vec{\sigma}}(\vec{x})$ means the vector $f_{\sigma_1}(x_1), \ldots, f_{\sigma_n}(x_n)$ and so on.

We proceed to prove this by induction on t.

Case t a variable:

The base cases of t a variable are trivial since ψ acts on variables as the identity.

Case $t = \star$:

$f_I(\star) = \Lambda\alpha.\, \lambda^\circ x\colon \alpha.\, x = \psi(\star)$.

Case $t = Y$:

Since Lemma A.8 tells us that $f_{\prod \alpha.\alpha\multimap\alpha}$ is the identity and $\psi(Y) = Y$, both sides of the equation are equal to Y.

Case $t = \lambda^\circ y_{n+1}\colon \sigma'_{n+1}.\, t'$:

We assume for notational simplicity that the lambda-abstraction is over the last variable of t' such that we write $t'(\vec{x},\vec{y},y_{n+1})$. By definition

$$\psi(\lambda^\circ y_{n+1}\colon \sigma'_{n+1}.\, t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \lambda^\circ y_{n+1}\colon \psi(\sigma'_{n+1}).\, \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), y_{n+1})$$

and

$$f_{\sigma'_{n+1}\multimap\tau}(\lambda^\circ y_{n+1}\colon \sigma'_{n+1}.\, t')(\vec{x},\vec{y}) = \lambda^\circ y_{n+1}\colon \psi(\sigma'_{n+1}).\, f_\tau(t'(\vec{x},\vec{y},g_{\sigma'_{n+1}}(y_{n+1})))$$

By induction hypothesis, we know that for any $y_{n+1}\colon \sigma'_{n+1}$ we have

$$f_\tau(t'(\vec{x},\vec{y},y_{n+1})) = \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), f_{\sigma'_{n+1}}(y_{n+1}))$$

In particular this holds if we set y_{n+1} to be $g_{\sigma'_{n+1}}(y_{n+1})$ and since g and f are inverses, we get the desired equality.

Case $t = t'\,t''$:

We have

$$\psi(t'\,t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))\, \psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))$$

which by induction hypothesis is equal to

$$f_{\tau'\multimap\tau}(t'(\vec{x},\vec{y}))\, f_{\tau'}(t''(\vec{x},\vec{y})) = f_\tau \circ t'(\vec{x},\vec{y}) \circ g_{\tau'}(f_{\tau'}(t''(\vec{x},\vec{y}))) = f_\tau(t'(\vec{x},\vec{y})(t''(\vec{x},\vec{y})))$$

proving the induction case.

Case $t = t' \otimes t''$:

Suppose $t' \colon \tau', t'' \colon \tau''$. By definition, we have

$$\begin{gathered}\psi(t' \otimes t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \\ \Lambda\alpha.\, \lambda^\circ h \colon \psi(\tau') \multimap \psi(\tau'') \multimap \alpha.\, h\ (\psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))\ (\psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))\end{gathered}$$

by induction hypothesis this equals

$$\Lambda\alpha.\, \lambda^\circ h \colon \psi(\tau') \multimap \psi(\tau'') \multimap \alpha.\, h\ f_{\tau'}(t'(\vec{x}, \vec{y}))\ f_{\tau''}(t''(\vec{x}, \vec{y}))$$

which is equal to

$$f_{\tau' \otimes \tau''}(t' \otimes t''(\vec{x}, \vec{y}))$$

Case $t = \Lambda\alpha.\, t'$:

$$\psi(\Lambda\alpha.\, t'(\vec{x}, \vec{y})) = \Lambda\alpha.\, \psi(t'(\vec{x}, \vec{y})).$$

By induction this equals

$$\Lambda\alpha.\, f_\sigma(t'(\vec{f}_{\vec{\sigma}}(\vec{x}), \vec{f}_{\vec{\sigma}'}(\vec{y}))) = f_{\prod \alpha.\sigma}(\Lambda\alpha.\, t'(\vec{f}_{\vec{\sigma}}(\vec{x}), \vec{f}_{\vec{\sigma}'}(\vec{y}))).$$

Case $t = t'(\omega)$:

Suppose $t' \colon \prod \alpha.\, \tau$. Now,

$$\psi(t'\ \omega)(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))\ \psi(\omega)$$

By induction hypothesis this is equal to

$$f_{\prod \alpha.\tau}(t'(\vec{x}, \vec{y}))\ \psi(\omega)$$

which by Lemma A.13 is equal to

$$f_{\tau[\omega/\alpha]}(t'(\vec{x}, \vec{y})\ \omega)$$

which proves the induction step.

Case $t =!t'$:

$$f_{!\tau'}(!t'(\vec{x}, \vec{y})) =!(f_{\tau'}(t'(\vec{x}, \vec{y})))$$

and since

$$\psi(!t'(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))) =!\psi(t'(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))$$

the case follows from the induction hypothesis.

Case $t = \text{let } z \otimes z' \text{ be } t' \text{ in } t''$:

Suppose in this case that $t' \colon \tau' \otimes \tau''$ and $t'' \colon \tau$. Now,

$$\begin{gathered}\psi(\text{let } z \otimes z' \text{ be } t' \text{ in } t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \\ \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))\ \psi(\tau)\ (\lambda^\circ x \colon \psi(\tau').\, \lambda^\circ y \colon \psi(\tau'').\, \psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), x, y))\end{gathered} \tag{9}$$

Now, by induction

$$\psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), x, y) = \psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), f_{\tau'} g_{\tau'}(x), f_{\tau''} g_{\tau''}(y)) = \\ f_{\tau}(t''(\vec{x}, \vec{y}, g_{\tau'}(x), g_{\tau''}(y)))$$

And so by using the induction hypothesis on t' (9) reduces to

$$f_{\tau' \otimes \tau''}(t'(\vec{x}, \vec{y}))\ \psi(\tau)\ (\lambda^{\circ} x \colon \psi(\tau').\ \lambda^{\circ} y \colon \psi(\tau'').\ f_{\tau}(t''(\vec{x}, \vec{y}, g_{\tau'}(x), g_{\tau''}(y))) = \\ \text{let } z \otimes z' \text{ be } t'(\vec{x}, \vec{y}) \text{ in } f_{\tau}(t''(\vec{x}, \vec{y}, g_{\tau'} f_{\tau'}(z), g_{\tau''} f_{\tau''}(z'))) = f_{\tau}(t(\vec{x}, \vec{y})).$$

Case $t = \text{let } !y \text{ be } t' \text{ in } t''$:

In this case, suppose t' has type $!\tau'$ and $t'' \colon \tau$.

$$\psi(\text{let } !x \text{ be } t' \text{ in } t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \text{let } !x \text{ be } \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) \text{ in } \psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), x).$$

Using the induction hypothesis, this is equal to

$$\text{let } !x \text{ be } f_{!\tau'}(t'(\vec{x}, \vec{y})) \text{ in } f_{\tau}(t''(\vec{x}, \vec{y}, g_{\tau'}(x))).$$

Since $f_{!\tau'} = !f_{\tau'}$, this is equal to

$$\text{let } !x \text{ be } t'(\vec{x}, \vec{y}) \text{ in } f_{\tau}(t''(\vec{x}, \vec{y}, g_{\tau'}(f_{\tau'}(x)))) = f_{\tau}(t(\vec{x}, \vec{y})).$$

Case $t = \text{let } \star \text{ be } t' \text{ in } t''$:

In this case t'' has type τ. Now,

$$\psi(\text{let } \star \text{ be } t' \text{ in } t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = (\psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))\ \psi(\tau)\ (\psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))$$

Using the induction hypothesis, this is equal to

$$f_I(t'(\vec{x}, \vec{y}))\ \psi(\tau)\ f_{\tau}(t''(\vec{x}, \vec{y})) = \text{let } \star \text{ be } t'(\vec{x}, \vec{y}) \text{ in } f_{\tau}(t''(\vec{x}, \vec{y})) = f_{\tau}(t(\vec{x}, \vec{y}))$$

□

Finally, Proposition A.1 is the collected statement of Proposition A.2 and Lemma A.7.

References

[1] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal*, volume 41 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 2000. 1, 7, 7.1, 8

[2] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*. To appear. 5

[3] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of linear Abadi & Plotkin logic. Technical Report TR-2005-57, IT University of Copenhagen, February 2005. (document), 1, 6, 6, 7.2, 7.3, 7.3, A, A

[4] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. Submitted, 2005. (document), 1

[5] André Joyal and Ieke Moerdijk. *Algebraic Set Theory*. Number 220 in London Mathematical Society Lecture Notes in Mathematics. Cambridge University Press, 1995. 4

[6] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Math. Struct. in Comp. Science*, 11, 1996. 2, 7

[7] R. E. Møgelberg. Parametric completion for models of polymorphic intuitionistic / linear lambda calculus. Technical Report TR-2005-60, IT University of Copenhagen, February 2005. 1, 8

[8] R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005. 3, 3, 3, 3, 4, 5, 5, 8

[9] G. D. Plotkin. Type theory and recursion (extended abstract). In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, page 374, Montreal, Canada, 19–23 June 1993. IEEE Computer Society Press. 1

[10] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1

[11] Gordon Plotkin and Martín Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, volume 664 of *Lecture Notes in Comput. Sci.*, pages 361–375. Springer, Berlin, 1993. 1

[12] E.P. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In S. Abramsky, editor, *Proc. 9th Symposium in Logic in Computer Science*, pages 364–371, Paris, 1994. I.E.E.E. Computer Society. 1, 5

[13] G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, University of Oxford, 1986. 2.1

[14] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. (document), 1, 2, 2.2, 2.3, 2.1, 2.13, 3, 3.5, 4, 5, 7, 7.1, 7.1, 7.1, 7.1, 7.7, 7.8, 7.1, 7.9, 7.10, 7.11, 8

[15] Andrej Ščedrov. Intuitionistic set theory. In *Harvey Friedman's research on the foundations of mathematics*, volume 117 of *Stud. Logic Found. Math.*, pages 257–284. North-Holland, Amsterdam, 1985. 2

[16] A. Simpson. Computational adequacy in an elementary topos. In *CSL: 12th Workshop on Computer Science Logic*. LNCS, Springer-Verlag, 1998. 7, 7.11

[17] A. Simpson. Elementary axioms for categories of classes. In *14th Symposium on Logic in Computer Science (LICS'99)*, pages 77–87, Washington - Brussels - Tokyo, July 1999. IEEE. 4

[18] A. Simpson. Computational adequacy for recursive types in models of intuitionistic set theory. *Annals of Pure and Applied Logic*, 130, 2004. 4, 7, 7.11

Parametric Completion for Models of Polymorphic Linear / Intuitionistic Lambda Calculus

Rasmus Ejlers Møgelberg

Abstract

We show how the externalization of an internal PILL$_Y$-model in a quasi-topos gives rise to a canonical pre-LAPL-structure in which the logic is the internal logic of the quasi-topos. This corresponds to how one intuitively would think of parametricity for such internal models.

We describe a parametric completion process based on [10, 1] which takes an internal model of PILL$_Y$ in a quasi-topos and builds a new internal PILL$_Y$-model in a presheaf topos over the original quasi-topos. The externalization of this PILL$_Y$-model extends to a full parametric LAPL-structure. However, this LAPL-structure is different from the canonical one, since the logic comes from the original quasi-topos.

The concrete LAPL-structure of [2] is basically an example of this parametric completion process, although it is presented a bit different in *loc. cit.*. The PILL$_Y$-model constructed using synthetic domain theory in [11, 8, 9] is an example of an application of the parametric completion process, but the LAPL-structure provided for it in [8, 9] is different from the one presented here.

Contents

1 Introduction

In this paper we study the parametric parametric completion process of [10, 1] in the setting of PILL$_Y$-models. We assume that the reader is familiar with the concept of LAPL-structure [2, 3], and we show that the parametric completion process produces parametric LAPL-structures, thus providing a rich family of these. In earlier papers we have constructed a domain theoretic parametric LAPL-structure [2, 3] and shown how to construct parametric LAPL-structures using synthetic domain theory [8]. These LAPL-structures seem to be examples of a parametric completion process, and so the motivation for this work was to describe this process in general.

An internal PILL$_Y$-model in a finitely complete category is an internal linear category with products which is complete enough to model polymorphism, such that the co-Kleisli category is an internal *sub*-category of the ambient category. Of course the externalization of the adjunction between an internal PILL$_Y$-model and the co-Kleisli category is a PILL$_Y$-model in the sense of [7]. If the ambient category is a quasi-topos, the internal logic is sufficiently rich for reasoning about parametricity, and thus we can construct a canonical pre-LAPL-structure around the externalization of the internal PILL$_Y$-model.

A notion of admissible relations for an internal PILL$_Y$-model is an internal logic fibration giving a sublogic of the regular subobject fibration, such that relations in the logic give a notion of admissible relations in the sense of [2]. The parametric completion process takes an internal PILL$_Y$-model with a notion of admissible relations in a quasi-topos and produces a PILL$_Y$-model in the category of reflexive graphs over the original quasi-topos. Basically, the types in the externalization of this PILL$_Y$-model are types in the original PILL$_Y$-model with a relational interpretation based on the given notion of admissible relations satisfying identity extension, and so the externalization extends to a parametric LAPL-structure. The LAPL-structure, however, is not the canonical LAPL-structure of the completed PILL$_Y$-model inside the quasitopos of reflexive graphs as described above. Instead it is build from the logic of the original quasi-topos. This is due to the relational interpretations of types being in terms of the logic of the original topos.

The concrete LAPL-structure considered in [2] is a result of the parametric completion process applied to admissible pers over a reflexive domain seen as an internal subcategory of the category of assemblies, although the presentation in [2] is slightly different. This example motivates the generalization to quasi-toposes instead of toposes. We could have also considered admissible pers as an internal category in the effective topos, but that would have given us a different logic. The PILL$_Y$-model constructed using synthetic domain theory in [11, 8, 9] is an example of an application of the parametric completion process, but the LAPL-structure provided for it in [8, 9] is different from the one presented here.

The paper is organized as follows: In Section 2 we review some internal category theory needed for the rest of the paper. Section 3 defines internal PILL$_Y$-models in quasi-toposes and the canonical pre-LAPL-structure associated to one such. We describe the parametric completion process in Section 4 and Section 5 discusses the LAPL-structures of [2, 8] as examples of the parametric completion process.

2 Internal structures in quasi-toposes

We start by recalling a bit of internal category theory. In particular we will discuss internal fibrations and internal linear categories. For a general introduction to internal category theory (in particular the definition of internal categories and externalization of internal categories), however, the reader is referred to text books on the subject such as [4].

2.1 Internal Fibrations

We define an internal fibration in a quasi-topos to be an internal functor $\mathbf{E} \to \mathbf{B}$ satisfying the proposition stating that all maps in $\mathbf{B}$ have cartesian liftings in the internal language of the quasi-topos. A cleavage for an internal fibration $p\colon \mathbf{E} \to \mathbf{B}$ is a map from the pullback

$$\begin{tikzcd}
\mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{B}_1 \arrow[d, "\mathrm{dom}"] \\
\mathbf{E}_0 \arrow[r, "p"] & \mathbf{B}_0
\end{tikzcd}$$

into $\mathbf{B}_1$ such that any element $(X, f\colon Y \to pX) \in \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$ is mapped to a cartesian lift $\bar{f}$ of f, i.e., the proposition

$$\begin{gathered}\forall Y\colon \mathbf{E}_0.\,\forall f\colon \mathbf{B}_1.\,\mathrm{codom} f = pY \supset \forall g\colon \mathbf{E}_1.\,\forall u\colon \mathbf{B}_1.\, f \circ u = p(g) \wedge \\ \mathrm{codom}(g) = Y \supset \exists! v\colon \mathbf{E}_1.\, p(v) = u \wedge \bar{f} \circ v = g\end{gathered} \tag{1}$$

holds in the internal logic. We will continue to write the cleavage function as $(X, f) \mapsto \bar{f}$. We say that p is cloven if there exists (externally) a cleavage.

Lemma 2.1. *An internal functor $p\colon \mathbf{E} \to \mathbf{B}$ is a cloven internal fibration in a quasi-topos iff*

$$\mathrm{Fam}(p)\colon \mathrm{Fam}(\mathbf{E}) \to \mathrm{Fam}(\mathbf{B})$$

is a fibration.

For the proof we need the following lemma

Lemma 2.2. *Suppose $\mathbb{D} \to \mathbb{C}$, $\mathbb{B} \to \mathbb{C}$ are fibrations, and*

$$\begin{tikzcd}
\mathbb{D} \arrow[rr, "p"] \arrow[dr] & & \mathbb{B} \arrow[dl] \\
& \mathbb{C} &
\end{tikzcd}$$

is a fibred map. If each restriction to a fibre:

$$p_c\colon \mathbb{D}_c \to \mathbb{B}_c$$

for $c \in \mathbb{C}_0$ is a fibration and reindexing along maps in $\mathbb{C}$ preserve cartesian lifts, then p is a fibration.

Proof. This is an easy exercise. □

Proof of Lemma 2.1. Suppose first $p\colon \mathbf{E} \to \mathbf{B}$ is an internal fibration with cleavage denoted $f \mapsto \bar{f}$. Using Lemma 2.2 it suffices to show that each fibre of $\mathrm{Fam}(p)$ is a cloven fibration with cleavage preserved under reindexing.

Suppose $X\colon \Xi \to \mathbf{E}_0$ is an object of $\mathrm{Fam}(\mathbf{E})$ and $f\colon \Xi \to \mathbf{B}_1$ is a vertical map in $\mathrm{Fam}(\mathbf{B})$ with codomain $p \circ X$. By composing with the cleavage for p we get a lift of f:

$$\bar{f}\colon \Xi \to \mathbf{E}_1.$$

Suppose now that we are given $g\colon \Xi \to \mathbf{E}_1$ such that $\text{codom} \circ g = X$ and $u\colon \Xi \to \mathbf{B}_1$ such that expressed internally

$$f \circ u = p(g).$$

By assumption the statement (1) holds in the internal logic of $\mathbb{E}$. Thus by description in a quasi-topos there exists a map from

$$K = \{(X, f, g, u) \in \mathbf{E}_0 \times \mathbf{B}_1 \times \mathbf{E}_1 \times \mathbf{B}_1 \mid \text{codom} f = pX \wedge f \circ u = p(g) \wedge \text{codom}(g) = X\}$$

to $\mathbf{E}_1$ providing the unique v of (1). We may now compose the pairing of the given (X, f, g, u) above with this map, to obtain the unique v needed. This proves that each fibre of the externalization is a fibration, and clearly the cleavage is preserved by reindexing because it is given by composing with the cleavage map $f \mapsto \bar{f}$.

For the other direction, consider the projections $X\colon \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1 \to \mathbf{E}_0$ and

$$f\colon \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1 \to \mathbf{B}_1.$$

Since these present respectively an object of $\mathrm{Fam}(\mathbf{E})$ and a morphism of $\mathrm{Fam}(\mathbf{B})$ we can consider the cartesian lift of (X, f), which is a morphism $\bar{f}$ satisfying

$$\begin{tikzcd}
 & \mathbf{B}_1 \\
\mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1 \arrow[ur, "f"] \arrow[r, "\bar{f}"] \arrow[dr, "X"'] & \mathbf{E}_1 \arrow[u, "p"'] \arrow[d, "\text{codom}"] \\
 & \mathbf{E}_0
\end{tikzcd}$$

Consider now the map $g\colon K \to \mathbf{E}_1$ given by the third projection considered as a map in $\mathrm{Fam}(\mathbf{E})$ from $\text{dom} \circ g$ to X over the projection $K \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$ in $\mathbb{E}$. Consider also the map $u\colon K \to \mathbf{B}_1$ given by the fourth projection considered as a map in $\mathrm{Fam}(\mathbf{B})$ with codomain $\text{dom} \circ f$ over the projection $K \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$ in $\mathbb{E}$. Now, $f \circ u = \mathrm{Fam}(p)(g)$ in $\mathrm{Fam}(\mathbf{B})$ by definition of K and so there exists a unique map in $\mathrm{Fam}(\mathbf{B})$ over the projection $K \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$ in $\mathbb{E}$ given by $v\colon K \to \mathbf{E}_1$ such that $\mathrm{Fam}(p)(v) = u$ and $\bar{f} \circ v = g$. This map v proves the proposition

$$\begin{gathered}
\forall X\colon \mathbf{E}_0.\, \forall f\colon \mathbf{B}_1.\, \text{codom} f = pX \supset \forall g\colon \mathbf{E}_1.\, \forall u\colon \mathbf{B}_1.\, f \circ u = p(g) \wedge \\
\text{codom}(g) = X \supset \exists v\colon \mathbf{E}_1.\, p(v) = u \wedge \bar{f} \circ v = g
\end{gathered}$$

in the internal logic of $\mathbb{E}$.

Finally, for uniqueness of v, we set g to be the third projection from

$$\begin{aligned}
K' \;=\; & \{(X, f, g, u, v, v') \in \mathbf{E}_0 \times \mathbf{B}_1 \times \mathbf{E}_1 \times \mathbf{B}_1 \times \mathbf{E}_1 \times \mathbf{E}_1 \mid \\
& \text{codom} f = pX \wedge f \circ u = p(g) \wedge p(v) = p(v') = u \wedge \bar{f} \circ v = \bar{f} \circ v = g\}
\end{aligned}$$

considered as a map in $\mathrm{Fam}(\mathbf{E})$ into X over the projection $K' \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$ in $\mathbb{E}$. We define u to be the fourth projection out of K' considered as a map in $\mathrm{Fam}(\mathbf{B})$ into $\text{dom} \circ f$ over the projection $K' \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$. Define v, v' to be the obvious projections out of K' considered as maps of $\mathrm{Fam}(\mathbf{E})$ into $\text{dom} \circ \bar{f}$ over $K' \to \mathbf{E}_0 \times_{\mathbf{B}_0} \mathbf{B}_1$. Since we still have $f \circ u = \mathrm{Fam}(p)(g)$ and $\bar{f} \circ v = g = \bar{f} \circ v'$, by $\mathrm{Fam}(\mathbf{E}) \to \mathrm{Fam}(\mathbf{B})$ being a fibration, we conclude that the projections onto the v and v' coordinates are equal, which proves the uniqueness of v in (1) in the internal logic of $\mathbb{E}$. □

Lemma 2.3. *For $p\colon \mathbf{Q} \to \mathbf{E}$ an internal cloven fibration and $f\colon \mathbf{F} \to \mathbf{E}$ a functor in a quasi-topos $\mathbb{E}$, the pullback of p along f is a cloven internal fibration.*

For the next two examples we assume that $\mathbf{C}$ is an internal *sub*-category of $\mathbb{E}$, that is, there exists a faithful fibred map

$$\begin{tikzcd}
\mathrm{Fam}(\mathbf{C}) \arrow[r, "\phi"] \arrow[dr] & \mathbb{E}^{\to} \arrow[d, "\text{codom}"] \\
& \mathbb{E}
\end{tikzcd}$$

with codom denoting the codomain fibration. We also assume that this map preserves monos.

Below, we will need to do a few calculations in the codomain fibration, and so we establish some notation first. An object $E \to X$ of $\mathbb{E}^{\to}$ will be denoted $\coprod_{x\in X} E_x \to X$. Recall that a quasi-topos is locally cartesian closed, and so the fibrewise products and exponents of $\coprod_{x\in X} E_x \to X$ and $\coprod_{x\in X} E'_x \to X$ are denoted

$$\textstyle\coprod_{x\in X} E'_x \times E_x \to X, \qquad \coprod_{x\in X} E'^{E_x}_x \to X$$

respectively. If $f\colon Y \to X$ is a map in $\mathbb{E}$ we may reindex $\coprod_{x\in X} E_x \to X$ along f, and we write the resulting object as $\coprod_{y\in Y} E_{f(y)} \to Y$.

Since $id_{\mathbf{C}_0}$ is an object in $\mathrm{Fam}(\mathbf{C})$ over $\mathbf{C}_0$, $\phi(id_{\mathbf{C}_0})$ is a map in $\mathbb{E}$ with codomain $\mathbf{C}_0$. We will denote the codomain of $\phi(id_{\mathbf{C}_0})$ by $\coprod_{c\in\mathbf{C}_0} c$. For any object $f\colon X \to \mathbf{C}_0$ in $\mathrm{Fam}(\mathbf{C})$ we must have

$$\phi(f) = \phi(f^*(id_{\mathbf{C}_0})) = f^*\phi(id_{\mathbf{C}_0}) = \textstyle\coprod_{x\in X} f(x).$$

We can reindexing $\coprod_{c\in\mathbf{C}_0} c$ along either of the two projections $\pi, \pi'\colon \mathbf{C}_0^2 \to \mathbf{C}_0$ and take the fibrewise exponent yielding

$$\textstyle\coprod_{x\in\mathbf{C}_0^2} \pi(x)^{\pi'(x)} \to \mathbf{C}_0^2$$

which we usually simply denote

$$\textstyle\coprod_{c,c'\,\in\mathbf{C}_0} c^{c'} \to \mathbf{C}_0^2.$$

Now, for any object $X \in \mathbb{E}$, vertical maps in $\mathrm{Fam}(\mathbf{C})$ from $f\colon X \to \mathbf{C}_0$ to $g\colon X \to \mathbf{C}_0$ are maps h making the diagram

$$\begin{tikzcd}
X \arrow[rr, "h"] \arrow[dr, "\langle f{,}g\rangle"'] & & \mathbf{C}_1 \arrow[dl, "\langle\text{dom,codom}\rangle"] \\
& \mathbf{C}_0^2 &
\end{tikzcd}$$

commute. The functor ϕ takes these maps to vertical maps in $\mathbb{E}^{\to}$ from $\phi(f)$ to $\phi(g)$, which using that id_X is the terminal object of $\mathbb{E}^{\to}_X$ correspond to maps

$$\begin{tikzcd}
X \arrow[rr] \arrow[dr, "id_X"'] & & \coprod_{x\in X} g(x)^{f(x)} \arrow[dl] \\
& X &
\end{tikzcd}$$

This correspondence is natural in X and therefore there must be a map

$$\begin{tikzcd}
\mathbf{C}_1 \arrow[rr] \arrow[dr, "\langle\text{dom,codom}\rangle"'] & & \coprod_{c,c'\in\mathbf{C}_0} c'^{c} \arrow[dl] \\
& \mathbf{C}_0^2 &
\end{tikzcd} \tag{2}$$

inducing the functorial part of ϕ. We will often denote the object $\mathbf{C}_1 \to \mathbf{C}_0^2$ by $\coprod_{c',c\in\mathbf{C}_0} \mathbf{C}(c',c)$.

Recall also that the pullback of the regular subobject fibration $\mathrm{RegSub}_{\mathbb{E}} \to \mathbb{E}$ along $\mathrm{dom}\colon \mathbb{E}^{\to} \to \mathbb{E}$ gives an indexed higher order logic fibration [1, Lemma A.8]

$$\begin{tikzcd} \mathbb{Q} \arrow[r] & \mathbb{E}^{\to} \arrow[r, "\text{codom}"] & \mathbb{E}\,. \end{tikzcd}$$

The indexed generic object for this indexed higher order logic fibration is the family of projections $(\Sigma\times\Xi \to \Xi)_{\Xi\in\mathbb{E}}$, where Σ is the regular subobject classifier of $\mathbb{E}$. Using the notation introduced above, we will denote the subobject classifier

$$\textstyle\coprod_{x\in\Xi} \Sigma \to \Xi.$$

Example 2.4. *In this example we construct an internal fibration* $\mathbf{RegSub}_{\mathbb{E}} \to \mathbf{C}$ *such that we have a pullback*

$$\begin{tikzcd} \mathrm{Fam}(\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C})) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{Q} \arrow[d] \\ \mathrm{Fam}(\mathbf{C}) \arrow[r, "\phi"] & \mathbb{E}^{\to}. \end{tikzcd} \tag{3}$$

Thus we can think of $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C}) \to \mathbf{C}$ *as the internalization of the restriction of* $\mathrm{RegSub}_{\mathbb{E}} \to \mathbb{E}$ *to* $\mathbf{C}$.

We define the object of objects $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C})_0$ *to be* $\coprod_{c\in\mathbf{C}_0} \Sigma^c$. *Using the ordering on* Σ *and the inclusion (2), we can form the fibred subobject*

$$\textstyle\coprod_{c,c'\in\mathbf{C}_0}\{(f,g,h,x)\colon \Sigma^c \times \Sigma^{c'} \times \mathbf{C}(c,c') \times c \mid f(x) \le g(h(x))\}$$

of

$$\textstyle\coprod_{c,c'\in\mathbf{C}_0} \Sigma^c \times \Sigma^{c'} \times \mathbf{C}(c,c') \times c$$

in the fibre over $\mathbf{C}_0^2$. *Using the fibred first-order logic on* $\mathbb{E}^{\to} \to \mathbb{E}$*, we can form the subobject*

$$\textstyle\coprod_{c,c'\in\mathbf{C}_0}\{(f,g,h)\colon \Sigma^c \times \Sigma^{c'} \times \mathbf{C}(c,c') \mid \forall x\colon c.\, f(x) \le g(h(x))\} \tag{4}$$

of

$$\textstyle\coprod_{c,c'\in\mathbf{C}_0} \Sigma^c \times \Sigma^{c'} \times \mathbf{C}(c,c').$$

We define $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C})_1$ *to be (4) with domain and codomain projections mapping* (f,g,h) *to* f *and* g *respectively. Composition is given by composing the* h*-component, and the map* $f\colon \Sigma^c \mapsto (f,f,id_c)$ *maps an object of* $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C})_0$ *to the identity on* f.

Finally, the internal fibration $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C}) \to \mathbf{C}$ *maps* (f,g,h) *to* h. *The cleavage maps* (f,a) *in* $\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C})_0 \times_{\mathbf{C}_0} \mathbf{C}_1$ *to* $(f\circ a, a, f)$.

For the pullback diagram (3), notice that an object of $\mathrm{Fam}(\mathbf{RegSub}_{\mathbb{E}}(\mathbf{C}))$ *over* $f\colon X \to \mathbf{C}_0$ *in* $\mathrm{Fam}(\mathbf{C})$ *is a map* g *making the diagram*

$$\begin{tikzcd} & \coprod_{c\in\mathbf{C}_0} \Sigma^c \arrow[d] \\ X \arrow[ur, "g"] \arrow[r, "f"] & \mathbf{C}_0 \end{tikzcd}$$

commute. Such maps correspond to diagrams

$$\begin{tikzcd} X \arrow[rr] \arrow[dr, "id"'] & & \coprod_{x\in X} \Sigma^{f(x)} \arrow[dl] \\ & X & \end{tikzcd}$$

which correspond to diagrams

$$\begin{tikzcd}
\coprod_{x\in X} f(x) \arrow[rr] \arrow[dr] & & \coprod_{x\in X} \Sigma \arrow[dl] \\
& X, &
\end{tikzcd}$$

i.e., subobjects of $\coprod_{x\in X} f(x) \to X$ *in* $\mathbb{Q} \to \mathbb{E}^{\to}$.

Example 2.5. *There are a few canonical subfibrations of* $\mathbf{RegSub}_{\mathbb{E}} \to \mathbf{C}$*. For example, the subobjects in* $\mathbf{RegSub}_{\mathbb{E}}$ *are represented by regular monos in* $\mathbb{E}$*, but one could also consider regular monos in* $\mathbf{C}$*. In this example, we consider the monos from* $\mathbf{C}$*, that are regular in* $\mathbb{E}$*, but may not be so inside* $\mathbf{C}$ *(the equalizer diagram may live in* $\mathbb{E}$*, but not in the subcategory* $\mathbf{C}$*). We call this fibration* $\mathbf{Sub}(\mathbf{C}) \cap \mathbf{RegSub}_{\mathbb{E}}$.

First define the object of monos in $\mathbf{C}_0$ *as*

$$\mathbf{Monos}_{\mathbf{C}} = \coprod_{c,c'\in\mathbf{C}_0}\{f\colon \mathbf{C}(c,c') \mid \forall x,y\colon c.\, f(x)=f(y) \supset x=y\}$$

We assume that $\mathbf{C}$ *is closed under pullbacks of monos, i.e., for every mono* g *and map* h *both in* $\mathbf{C}$ *with the same codomain there exists a mono* g' *in* $\mathbf{C}$ *such that* g' *is the pullback of* g *along* h *as seen from* $\mathbb{E}$*. This can be expressed in the internal logic of* $\mathbb{E}$*, but notice that a diagram in* $\mathbf{C}$ *which is a pullback in* $\mathbb{E}$ *need not necessarily be a pullback in* $\mathbf{C}$*, since* $\mathbf{C}$ *is not required to be a* full *subcategory of* $\mathbb{E}$.

The object of objects $(\mathbf{Sub}(\mathbf{C}) \cap \mathbf{RegSub}_{\mathbb{E}})_0$ *is*

$$\coprod_{c\in\mathbf{C}_0}\{f\colon \Sigma^c \mid \exists c'\colon \mathbf{C}_0.\, \exists g\colon \mathbf{Monos}_{\mathbf{C}}(c',c).\, \forall x\colon c.\, f(x) \supset \exists y\colon c'.\, g(y)=x\}$$

and we consider this as a full subcategory of $\mathbf{RegSub}_{\mathbb{E}}$*. The assumption of closure under pullbacks of monos is what makes this a subfibration of* $\mathbf{RegSub}_{\mathbb{E}} \to \mathbb{E}$.

Remark 2.6. *Example 2.5 would have been simpler, if the internal category* $\mathbf{C}$ *had been a* full *internal subcategory. In the cases we consider, however, this will very often not be the case, since we will consider internal categories with comonads, such that the co-Kleisli category is an internal subcategory of the ambient category. In these cases* $\mathbf{C}$ *being a subcategory of the co-Kleisli category is a subcategory of the ambient category, but it is only full if the comonad is trivial.*

2.2 Internal linear categories

An internal linear category is an internal category with internal functors $\otimes, \multimap, !$ and the usual internal natural transformations such that the usual equations hold in the internal language (see [7, Definition 1.10]). Since the concept of internal categories and internal linear categories can be expressed in any finitely complete category, the standard assumption of this section will be that the ambient category $\mathbb{E}$ is simply a finitely complete category (and not necessarily a quasi-topos).

Lemma 2.7. *Suppose* $\mathbf{C}$ *is an internal category in a finitely complete category* $\mathbb{E}$*. There is a bijective correspondence between internal linear category structures on* $\mathbf{C}$ *and fibred linear structures on* $\mathrm{Fam}(\mathbf{C}) \to \mathbb{E}$.

Proof. This is a consequence of the externalization functor being a locally full and faithful 2-functor preserving products [4, Proposition 7.3.8]. □

For any finitely complete category $\mathbb{E}$ we define $\mathbf{Cat}(\mathbb{E})$ to be the category of internal categories and internal functors in $\mathbb{E}$. Likewise, we define $\mathbf{LinCat}(\mathbb{E})$ to be the category of internal linear categories and internal functors preserving the linear structure on the nose in $\mathbb{E}$. We write internal categories as

$$\begin{tikzcd} \mathbf{C}_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C}_0 \arrow[l] \end{tikzcd}$$

where $\mathbf{C}_1$ is the object of morphisms and $\mathbf{C}_0$ is the object of objects. Strictly speaking, the composition map should be mentioned in the description of the internal category, but we will often leave it implicit or denote it by comp.

For categories $\mathbb{E}, \mathbb{C}$ we denote by $\mathbb{E}^{\mathbf{C}}$ the category of functors and natural transformations. The rest of this section is devoted to proving the following (well-known) lemma:

Lemma 2.8. *Suppose $\mathbb{E}$ is a finitely complete category and $\mathbb{C}$ is any category. Then*

$$\mathbf{Cat}(\mathbb{E}^{\mathbb{C}}) \cong \mathbf{Cat}(\mathbb{E})^{\mathbb{C}}.$$

In one direction, the isomorphism associates to an internal category $F_1 \overset{\longrightarrow}{\underset{\longrightarrow}{\longleftarrow}} F_0$ in $\mathbb{E}^{\mathbb{C}}$ the functor that to each $c \in \mathbb{C}_0$ associates the internal category $F_1(c) \overset{\longrightarrow}{\underset{\longrightarrow}{\longleftarrow}} F_0(c)$ in $\mathbb{E}$. Likewise there is an isomorphism

$$\mathbf{LinCat}(\mathbb{E}^{\mathbb{C}}) \cong \mathbf{LinCat}(\mathbb{E})^{\mathbb{C}}.$$

For $\mathbf{Cat}(\mathbb{E}^{\mathbb{C}})$ to even make sense, we need the following lemma.

Lemma 2.9. *If $\mathbb{E}$ is a finitely complete category and $\mathbb{C}$ is any category, the category $\mathbb{E}^{\mathbf{C}}$ is finitely complete, and limits are computed pointwise.*

Proof. This is well-known, see for example [6, p. 22] or [5, p. 116]. □

Lemma 2.10. *Suppose $\mathbb{E}, \mathbb{F}$ are finitely complete categories and $F\colon \mathbb{E} \to \mathbb{F}$ is a functor preserving finite limits. Then F induces a functor $\mathbf{Cat}(F)\colon \mathbf{Cat}(\mathbb{E}) \to \mathbf{Cat}(\mathbb{F})$.*

If $G\colon \mathbb{E} \to \mathbb{F}$ is another finite limit preserving functor then any natural transformation $\mu\colon F \Rightarrow G$ induces a natural transformation $\mathbf{Cat}(\mu)\colon \mathbf{Cat}(F) \Rightarrow \mathbf{Cat}(G)$.

Moreover, F induces a functor

$$\mathbf{LinCat}(F)\colon \mathbf{LinCat}(\mathbb{E}) \to \mathbf{LinCat}(\mathbb{F})$$

and μ induces a natural transformation.

$$\mathbf{LinCat}(F) \Rightarrow \mathbf{LinCat}(G)$$

Proof. The functor $\mathbf{Cat}(F)$ maps an internal category

$$\begin{tikzcd} \mathbf{C}_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C}_0 \arrow[l] & \text{to} & F(\mathbf{C}_1) \arrow[r, shift left=2] \arrow[r, shift right=2] & F(\mathbf{C}_0) \arrow[l] \end{tikzcd}.$$

For this to be an internal category in $\mathbf{Cat}(\mathbb{F})$ we also need a composition map. Since F preserves finite limits, we have a pullback

$$\begin{tikzcd} F(\mathbf{C}_1 \times_{\mathbf{C}_0} \mathbf{C}_1) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & F(\mathbf{C}_1) \arrow[d, "F(\text{dom})"] \\ F(\mathbf{C}_1) \arrow[r, "F(\text{codom})"] & F(\mathbf{C}_0). \end{tikzcd}$$

Thus we can define the composition map by applying F to the composition map of $\mathbf{C}_1 \mathrel{\substack{\longrightarrow\\ \longleftarrow\\ \longrightarrow}} \mathbf{C}_0$. Clearly we can also apply F to internal functors of $\mathbb{E}$ (or internal natural transformations) and obtain internal functors (or internal natural transformations) in $\mathbb{F}$.

The natural transformation $\mathbf{Cat}(\mu)$ has as component at $\mathbf{C}_1 \mathrel{\substack{\longrightarrow\\ \longleftarrow\\ \longrightarrow}} \mathbf{C}_0$ the pair $(\mu_{\mathbf{C}_0}, \mu_{\mathbf{C}_1})$, which defines an internal functor by naturality of μ. Naturality of μ also implies naturality of $\mathbf{Cat}(\mu)$.

We define $\mathbf{LinCat}(F)$ as $\mathbf{Cat}(F)$ by applying F to all structure of the internal linear category. Again, it is crucial that F preserves finite limits, since for example F of the object part of the tensor functor is a map $F(\mathbf{C}_0 \times \mathbf{C}_0) \to F(\mathbf{C}_0)$ in $\mathbb{F}$, and we need a map with domain $F(\mathbf{C}_0) \times F(\mathbf{C}_0)$. The definition of an internal linear category requires a number of diagrams to commute (using [7, Lemma 1.11] to modify the last condition of [7, Definition 1.10]). Applying F to all these commutative diagrams of course yield commutative diagrams, and thus applying F to all the internal linear category structure does give an internal linear category structure.

If H is an internal functor between internal linear categories commuting with the linear structure of these, then $F(H)$ also commutes with the internal linear structure which proves that $\mathbf{LinCat}(F)$ does in fact define a functor.

For natural transformations μ, the naturality of μ implies that it commutes with all linear category structure, which proves that $\mathbf{LinCat}(\mu)$ is a natural transformation. □

Proof of Lemma 2.8. For each $c \in \mathbb{C}$, the functor $\mathrm{ev}_c \colon \mathbb{E}^{\mathbb{C}} \to \mathbb{E}$ given by evaluation at c preserves limits. By Lemma 2.10 we get an induced functor

$$\mathbf{Cat}(\mathbb{E}^{\mathbb{C}}) \to \mathbf{Cat}(\mathbb{E}).$$

For $f \colon c \to c'$ in $\mathbb{C}$, we have a natural transformation from ev_c to $\mathrm{ev}_{c'}$. This induces a functor

$$\mathbf{Cat}(\mathbb{E}^{\mathbb{C}}) \times \mathbb{C} \to \mathbf{Cat}(\mathbb{E}).$$

The functor ϕ of the lemma is the adjoint of this map. This proves that ϕ in fact is a well defined functor.

We call the inverse of ϕ for ψ. To define it, notice that we have two functors

$$(-)_0, (-)_1 \colon \mathbf{Cat}(\mathbb{E}) \to \mathbb{E}$$

mapping an internal category to its object of objects and morphisms respectively. We have natural transformations $\mathrm{dom}, \mathrm{codom}, \mathrm{id}$ between these corresponding to domain and codomain maps and identity, and we have a natural transformation

$$\mathrm{comp} \colon (-)_1 \times_{(-)_0} (-)_1 \Rightarrow (-)_1$$

given by the composition map in internal categories. These induce functors

$$(-)_0^{\mathbb{C}}, (-)_1^{\mathbb{C}} \colon \mathbf{Cat}(\mathbb{E})^{\mathbb{C}} \to \mathbb{E}^{\mathbb{C}}$$

and natural transformations $\mathrm{dom}^{\mathbb{C}}, \mathrm{codom}^{\mathbb{C}}, \mathrm{id}^{\mathbb{C}}$. Since

$$(-)_1^{\mathbb{C}} \times_{(-)_0^{\mathbb{C}}} (-)_1^{\mathbb{C}} \cong ((-)_1 \times_{(-)_0} (-)_1)^{\mathbb{C}}.$$

we also have a natural transformation $\mathrm{comp}^{\mathbb{C}} \colon (-)_1^{\mathbb{C}} \times_{(-)_0^{\mathbb{C}}} (-)_1^{\mathbb{C}} \to (-)_1^{\mathbb{C}}$.

The map ψ maps $F \colon \mathbf{Cat}(\mathbb{E})^{\mathbb{C}}$ to the diagram

$$\begin{tikzcd}
(-)_1^{\mathbb{C}}(F) \arrow[r, shift left=2] \arrow[r, shift right=2] & (-)_0^{\mathbb{C}}(F) \arrow[l]
\end{tikzcd}$$

which is clearly an internal category. For $H\colon F \Rightarrow G$ a morphism in $\mathbf{Cat}(\mathbb{E})^{\mathbb{C}}$ the natural transformations

$$(-)^{\mathbb{C}}_0(H)\colon (-)^{\mathbb{C}}_0(F) \to (-)^{\mathbb{C}}_0(G), \qquad (-)^{\mathbb{C}}_1(H)\colon (-)^{\mathbb{C}}_1(F) \to (-)^{\mathbb{C}}_1(G)$$

are morphisms in $\mathbb{E}^{\mathbb{C}}$. To check that this defines an internal functor in $\mathbb{E}^{\mathbb{C}}$ we must check that it commutes with $\mathrm{dom}, \mathrm{codom}, \mathrm{id}, \mathrm{comp}$, which it does, since these are natural transformations.

It is easy to see that ϕ, ψ are each others inverses.

By Lemma 2.10 we can define the map

$$\mathbf{LinCat}(\mathbb{E}^{\mathbb{C}}) \to \mathbf{LinCat}(\mathbb{E})^{\mathbb{C}}.$$

as we defined ϕ.

Finally, to define the map $\psi\colon \mathbf{LinCat}(\mathbb{E})^{\mathbb{C}} \to \mathbf{LinCat}(\mathbb{E}^{\mathbb{C}})$, notice that as above, we can define functors

$$(-)_0, (-)_1\colon \mathbf{LinCat}(\mathbb{E}) \to \mathbb{E}$$

and proceed as before. But this time we have many other natural transformations:

$$\begin{gathered} !_0\colon (-)_0 \Rightarrow (-)_0 \\ !_1\colon (-)_1 \Rightarrow (-)_1 \\ \otimes_0\colon (-)_0 \times (-)_0 \Rightarrow (-)_0 \\ \dots \\ \epsilon\colon (-)_0 \Rightarrow (-)_1 \\ \dots \end{gathered}$$

satisfying the usual equations. If we proceed as above we can thus define the functor

$$\psi\colon \mathbf{LinCat}(\mathbb{E})^{\mathbb{C}} \to \mathbf{LinCat}(\mathbb{E}^{\mathbb{C}})$$

as desired. As before, clearly ϕ, ψ are each others inverses. □

3 Internal PILL$_Y$-models

Definition 3.1. *Suppose we are given a quasi-topos* $\mathbb{E}$*. An* internal PILL$_Y$-model *in* $\mathbb{E}$ *is an internal linear category* $\mathbf{C}$ *with products such that*

1. $\mathbf{C}$ *is complete enough to model polymorphism, i.e., for all objects* Ξ *in* $\mathbb{E}$ *there exists right Kan extensions of all functors* $\Xi \times \mathbf{C}_0 \to \mathbf{C}$ *along the projection* $\Xi \times \mathbf{C}_0 \to \Xi$*. Here* Ξ *and* $\mathbf{C}_0$ *are considered as discrete categories.*

2. *The co-Kleisli category for* $!\colon \mathbf{C} \to \mathbf{C}$ *denoted* $\mathbf{C}_!$ *is an internal subcategory of* $\mathbb{E}$

3. *The products of* $\mathbf{C}_!$ *coincide with the products of* $\mathbb{E}$*.*

4. $\begin{tikzcd} \mathbf{C} \arrow[r, bend right] & \mathbf{C}_! \arrow[l, bend right] \end{tikzcd}$ *models the fixed point combinator* Y*, i.e., there exists a term* $Y\colon 1 \to \mathbf{C}_1$ *such that*

$$\begin{tikzcd} & \mathbf{C}_0 \\ 1 \arrow[ur, "I"] \arrow[r, "Y"] \arrow[dr, "{[\![\prod \alpha.(\alpha\to\alpha)\to\alpha]\!]}"'] & \mathbf{C}_1 \arrow[u, "dom"'] \arrow[d, "codom"] \\ & \mathbf{C}_0 \end{tikzcd}$$

commutes, where $[\![\prod\alpha.\,(\alpha\to\alpha)\to\alpha]\!]$ *is interpreted using Item 1, and such that it holds in the internal logic of* $\mathbb{E}$ *that*

$$\forall c\colon \mathbf{C}_0.\,\forall f\colon {!c}\multimap c.\, f(!(Y\,c\,!f)) = Y\,c\,!f.$$

Remark 3.2. *One can always construct* $\mathbf{C}_!$ *as an internal category in* $\mathbb{E}$*, but in Definition 3.1 we ask for it to be an internal* sub*category of* $\mathbb{E}$ *as defined in Section 2.1. Using the embedding of* $\mathbf{C}$ *into* $\mathbf{C}_!$ *we see that* $\mathbf{C}$ *is also an internal subcategory of* $\mathbb{E}$ *by the composite map*

$$\begin{tikzcd}
\mathrm{Fam}(\mathbf{C}) \arrow[r] \arrow[dr] & \mathrm{Fam}(\mathbf{C}_!) \arrow[r] \arrow[d] & \mathbb{E}^{\to} \arrow[dl] \\
& \mathbb{E} &
\end{tikzcd}$$

which also preserves fibred products.

We now describe how an internal PILL$_Y$-model gives rise to a pre-LAPL-structure in a canonical way in which the internal logic of $\mathbb{E}$ gives the logic of the pre-LAPL-structure.

The regular subobject fibration $\mathrm{RegSub}(\mathbb{E})\to\mathbb{E}$ induces a fibration $\mathbb{Q}\to\mathbb{E}^{\to}$ given as

$$\begin{tikzcd}
\mathbb{Q} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{RegSub}(\mathbb{E}) \arrow[d] \\
\mathbb{E}^{\to} \arrow[r, "\mathrm{dom}"] & \mathbb{E}
\end{tikzcd}$$

Proposition 3.3. *Given an internal model of PILL*$_Y$*, the schema*

$$\begin{tikzcd}
& & & \mathbb{Q} \arrow[d] \\
\mathrm{Fam}(\mathbf{C}) \arrow[rr, bend right, ""{name=B}] \arrow[dr] & & \mathrm{Fam}(\mathbf{C}_!) \arrow[ll, bend right, ""{name=A}] \arrow[r] \arrow[dl] & \mathbb{E}^{\to} \arrow[dll] \\
& \mathbb{E} & &
\arrow[from=A, to=B, phantom, "\perp"]
\end{tikzcd}$$

is a pre-LAPL-structure.

Proof. By [1, Lemma A.4] we have a fibred first order logic fibration.

The only non-trivial part of the proof is the construction of the map

$$\begin{tikzcd}
\mathrm{Fam}(\mathbf{C})\times_{\mathbb{E}}\mathrm{Fam}(\mathbf{C}) \arrow[rr, "U"] \arrow[dr] & & \mathbb{E}^{\to} \arrow[dl] \\
& \mathbb{E}. &
\end{tikzcd}$$

We define U to be

$$(f\colon \Xi\to\mathbf{C}_0, g\colon \Xi\to\mathbf{C}_0)\mapsto \textstyle\coprod_{x\in\Xi}(\mathbf{RegSub}_{\mathbb{E}})_{f(x)\times g(x)},$$

i.e., the pullback of $(\mathbf{RegSub}_{\mathbb{E}})_0\to\mathbf{C}_0$ along the composite

$$\Xi \xrightarrow{\langle f,g\rangle} \mathbf{C}_0^2 \xrightarrow{\times} \mathbf{C}_0.$$

Maps in the fibre from any object $X \to \Xi$ to $U(f, g)$ correspond to maps from the fibrewise product of $X \to \Xi$ and $\phi(f \times g \colon \Xi \to \mathbf{C}_0)$ to the subobject classifier $\coprod_{x \in \Xi} \Sigma$ of $\mathbb{Q} \to \mathbb{E}^{\to}$. Clearly the functor U satisfies the requirements for LAPL-structures [2, Definition 3.1]. □

Definition 3.4. *A subfibration*

$$\begin{tikzcd}
\mathbf{Q} \arrow[r, hook] \arrow[dr, "p"'] & \mathbf{RegSub}_{\mathbb{E}}(\mathbf{C}) \arrow[d] \\
& \mathbf{C}.
\end{tikzcd}$$

gives an internal notion of admissible relations if $\mathbf{Q}$ *is closed under the rules for admissible relations as expressed in LAPL (Figure 3 and Axiom 2.18 of [7]).*

An internal notion of admissible relations gives rise to a subfunctor of U by:

$$V(f \colon \Xi \to \mathbf{C}_0, g \colon \Xi \to \mathbf{C}_0) = \textstyle\coprod_{x \in \Xi} \mathbf{Q}_{f(x) \times g(x)}$$

which gives a notion of admissible relation for the LAPL-structure given by Proposition 3.3.

Remark 3.5. *In many situations the fibration* $p \colon \mathbf{Q} \to \mathbf{C}$ *will be the fibration of regular subobjects on objects of* $\mathbf{C}$ *represented by monos in* $\mathbf{C}$ *as in Example 2.5. In such cases the fibration will be closed under some constructions such as equality and reindexing along maps from* $\mathbf{C}$*, but one will need to check some of the other conditions in the concrete case.*

4 Parametric completion

In this section we assume

- $\mathbb{E}$ is a quasi-topos
- $\mathbf{C}$ is an internal PILL$_Y$-model in $\mathbb{E}$ which has pullbacks of monos and these are preserved by the inclusion into $\mathbb{E}$.
- $\mathbf{Q} \to \mathbf{C}$ is a cloven internal subfibration of $\mathbf{Sub}(\mathbf{C}) \cap \mathbf{RegSub}_{\mathbb{E}} \to \mathbf{C}$ giving a notion of admissible relations
- the proposition
$$Y((\forall \alpha, \beta, R \colon \mathbf{AdmRel}(\alpha, \beta)).\, (R \to R) \to R)Y$$
holds in the pre-LAPL-structure associated to the internal PILL$_Y$-model $\mathbf{C}$ as in Proposition 3.3 with admissible relations given by $\mathbf{Q} \to \mathbf{C}$.

We show how to construct a parametric internal PILL$_Y$-model from this. However, we stress that the internal PILL$_Y$-model is not exactly parametric with respect to the LAPL-structure constructed in the previous section, but with respect to an LAPL-structure with a different logic. Since the PILL$_Y$-model is just the externalization of the internal PILL$_Y$-model, we still get proofs of the consequences of parametricity for this.

Consider the internal category $\mathbf{LR_Q(C)}$ whose objects are pairs of objects of $\mathbf{C}$ plus a relation on their product (relations in sense of the logic from $\mathbf{Q} \to \mathbf{C}$ and morphisms are pairs of morphisms preserving relations, i.e., $\mathbf{LR_Q(C)}$ is given by the pull-back

$$\begin{tikzcd}
\mathbf{LR_Q(C)} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Q} \arrow[d, "p"] \\
\mathbf{C} \times \mathbf{C} \arrow[r, "\times"] & \mathbf{C}
\end{tikzcd}$$

of internal categories in $\mathbb{E}$.

Notice that we clearly have a reflexive graph of functors

$$\begin{tikzcd}
\mathbf{LR_Q(C)} \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l]
\end{tikzcd} \tag{5}$$

where the two maps from $\mathbf{LR_Q(C)}$ to $\mathbf{C}$ are the domain and codomain map respectively, and the map going the other way is the map that maps $d \in \mathbf{C}$ to the equality relation on d.

Let G denote the small category

$$\begin{tikzcd}
\cdot \arrow[r, shift left=2] \arrow[r, shift right=2] & \cdot \arrow[l]
\end{tikzcd} .$$

Lemma 2.8 states that internal categories in $\mathbb{E}^G$ are reflexive graphs of internal categories in $\mathbb{E}$, and so the reflexive graph (5) is an internal category in $\mathbb{E}^G$. We aim to show that this internal category has the Kan-extensions needed to model polymorphism. We proceed exactly as in [1] but include the proofs for completeness. Consider first the internal category

$$\begin{tikzcd}[column sep=small]
& \mathbf{LR_Q(C)} \arrow[dl] \arrow[dr] & \\
\mathbf{C} & & \mathbf{C}
\end{tikzcd}$$

in the quasi-topos $\mathbb{E}^\Lambda$, where Λ is the obvious diagram. Consider further the fibration

$$\mathbf{LinAdmRelations_C} \to \mathbf{AdmRelCtx_C}$$

constructed as usual from the pre-LAPL-structure associated to $\mathbf{C}$ with admissible relations from $\mathbf{Q}$.

Lemma 4.1. *The fibrations*

$$\mathbf{LinAdmRelations_C} \to \mathbf{AdmRelCtx_C}$$

and

$$\mathrm{Fam}\left(\begin{tikzcd}[column sep=small]
& \mathbf{LR_Q(C)} \arrow[dl] \arrow[dr] & \\
\mathbf{C} & & \mathbf{C}
\end{tikzcd}\right) \to \mathbb{E}^\Lambda$$

are isomorphic.

Proof. Unwinding the definition of $\mathbf{AdmRelCtx_C}$, we find that the objects are triples (Ξ_0, Ξ_1, Ξ) together with maps $\Xi \to \Xi_0 \times \Xi_1$ in $\mathbb{E}$. A map from $\Xi \to \Xi_0 \times \Xi_1$ to $\Xi' \to \Xi_0' \times \Xi_1'$ is a triple

$$\rho \colon \Xi \to \Xi', f \colon \Xi_0 \to \Xi_0', g \colon \Xi_1 \to \Xi_1'$$

making the obvious diagram commute. Thus $\mathbf{AdmRelCtx_C} \cong \mathbb{E}^\Lambda$.

Objects in $\mathbf{LinAdmRelations_C}$ are given as morphism in $\mathbf{AdmRelCtx_C}$ into the interpretation of $\alpha, \beta \mid R \subset \alpha \times \beta$ which is $\coprod_{\alpha,\beta \in \mathbf{C}_0} (\mathrm{RegSub}_\mathbf{C}(\mathbf{C}))_{\alpha\times\beta} \to \mathbf{C}_0 \times \mathbf{C}_0$, and since

$$\mathbf{LR_Q(C)}_0 = \textstyle\coprod_{\alpha,\beta \in \mathbf{C}_0} (\mathrm{RegSub}_\mathbf{C}(\mathbf{C}))_{\alpha\times\beta}$$

we get a bijective correspondence between the objects of $\mathrm{Fam}\left(\begin{tikzcd}[row sep=small, column sep=tiny] & \mathbf{LR}_{\mathbf{Q}}(\mathbf{C}) \arrow[dl] \arrow[dr] & \\ \mathbf{C} & & \mathbf{C}\end{tikzcd}\right)$ and $\mathbf{LinAdmRelations}_{\mathbf{D}}$. For morphisms, a vertical morphism in $\mathrm{Fam}\left(\begin{tikzcd}[row sep=small, column sep=tiny] & \mathbf{LR}_{\mathbf{Q}}(\mathbf{C}) \arrow[dl] \arrow[dr] & \\ \mathbf{C} & & \mathbf{C}\end{tikzcd}\right)$ from (f, g, ρ) to (f', g', ρ') is by the above discussion a pair of morphisms $t\colon f \to f', s\colon g \to g'$ satisfying $\rho \supset (t \times s)^* \rho'$, which is exactly the same as a vertical morphism in $\mathbf{LinAdmRelations}_{\mathbf{C}}$. □

Lemma 4.2. *The fibration*

$$\mathrm{Fam}\left(\begin{tikzcd}[row sep=small, column sep=tiny] & \mathbf{LR}_{\mathbf{Q}}(\mathbf{C}) \arrow[dl] \arrow[dr] & \\ \mathbf{C} & & \mathbf{C}\end{tikzcd}\right) \to \mathbb{E}^{\Lambda}$$

has simple products, i.e., models polymorphism.

Proof. This is a consequence of Lemma 4.1. □

Let us now consider the case that we are really interested in. We shall assume that we are given a functor (f_0, f_1) in $\mathbb{E}^G$:

$$\begin{tikzcd}
\Xi' \times \mathbf{LR}_{\mathbf{Q}}(\mathbf{C})_0 \arrow[rr, "\pi"] \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] \arrow[ddrr, "f_1"] & & \Xi' \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] \\
\Xi \times \mathbf{C}_0 \arrow[u, "I" description] \arrow[rr, "\pi"] \arrow[dddrr, "f_0"] & & \Xi \arrow[u, "I" description] \\
& & \mathbf{LR}_{\mathbf{Q}}(\mathbf{C}) \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] \\
& & \mathbf{C}, \arrow[u, "I" description]
\end{tikzcd} \tag{6}$$

(considering the sets mentioned above as discrete categories) and we would like to find a right Kan extension of (f_0, f_1) along (π, π) (notice that we have used the notation $\partial_0, \partial_1, I$ for the structure maps of all objects of $\mathbb{E}^G$ - this should not cause any confusion, since it will be clear from the context which map is referred to). Let us call this extension $((\prod_{par} f)_0, (\prod_{par} f)_1)$. An obvious idea is to try the pair $((\prod f)_0, (\prod f)_1)$ provided by Lemma 4.2. However, $(\prod_{par} f)_1$ should commute with I, and we cannot know that $(\prod f)_1$ will do that. Consider $(\prod f)_1(I(A))$ for some $A \in \Xi$:

$$\begin{tikzcd}
(\prod f)_1(I(A)) \arrow[d, tail] \\
(\prod f)_0(A) \times (\prod f)_0(A).
\end{tikzcd}$$

If we pull this relation back along the diagonal on $(\prod f)_0(A)$ we get a subobject

$$|(\textstyle\prod f)_1(I(A))| \rightarrowtail (\textstyle\prod f)_0(A)$$

(called the *field* of $(\prod f)_1(I(A))$). Logically, $|(\prod f)_1(I(A))|$ is the set $\{x \in (\prod f)_0(A) \mid (x, x) \in \prod f_1(I(A))\}$, so if we restrict $(\prod f)_1(I(A))$ to this subobject, we get a relation relation containing the identity relation. The other inclusion will be easy to prove. Thus the idea is to let $(\prod_{par} f)_0$ be the map that maps A to $|\prod f_1(I(A))|$, and let $\prod_{par} f_1(R)$ be the relation obtained by restricting $(\prod f)_1(R)$ to $\prod_{par} f_0(\partial_0(R)) \times \prod_{par} f_0(\partial_1(R))$.

Notice that in the above sketch and the proof below, since $\mathbf{Q}$ consist of subobjects in $\mathbf{C}$, all objects and morphisms are in the category $\mathbf{C}$. However, by pullbacks we mean pullbacks in the greater category $\mathbb{E}$, since these give the reindexing in $\mathbf{Q} \to \mathbf{C}$. A pullback in $\mathbb{E}$ need not be a pullback in $\mathbf{C}$ even if all maps in the diagram are in $\mathbf{C}$, since $\mathbf{C}$ is not required to be a *full* subcategory of $\mathbb{E}$.

Lemma 4.3. *The fibration*

$$\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q(C)}\\ \downarrow\uparrow\downarrow\\ \mathbf{C}\end{pmatrix} \to \mathbb{E}^G$$

models polymorphism.

Proof. We define $(\prod_{par} f)_0(A)$ as the pullback

$$\begin{tikzcd}
(\prod_{par} f)_0(A) \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "\lrcorner", very near start] & (\prod f)_1(I(A)) \arrow[d, tail] \\
(\prod f)_0(A) \arrow[r, "\Delta"] & (\prod f)_0(A) \times (\prod f)_0(A)
\end{tikzcd}$$

where Δ is the diagonal map. We define $(\prod_{par} f)_1(R)$ for $R \in \Xi'$, to be the pullback

$$\begin{tikzcd}
(\prod_{par} f)_1(R) \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "\lrcorner", very near start] & (\prod f)_1(R) \arrow[d, tail] \\
(\prod_{par} f)_0(\partial_0 R) \times (\prod_{par} f)_0(\partial_1 R) \arrow[r, tail] & (\prod f)_0(\partial_0 R) \times (\prod f)_0(\partial_1 R).
\end{tikzcd}$$

We first show that $(\prod_{par} f)_1(I(A)) = I((\prod_{par} f)_0(A))$ for all A. Logically

$$(\textstyle\prod_{par} f)_1(I(A)) = \{(x,y) \in (\prod f)_1(I(A)) \mid (y,y),(x,x) \in (\prod f)_1(I(A))\} \supseteq \\ \{(x,x) \mid x \in (\prod_{par} f)_0(A)\} = I((\prod_{par} f)_0(A))$$

To prove the other inclusion suppose $(x,y) \in (\prod_{par} f)_1(I(A)) \subseteq (\prod f)_1(I(A))$. Then for any $\sigma_{n+1} \in \mathbf{C}_0$,

$$(x,y) \in \pi^*((\textstyle\prod f)_1)(I(A), I(\sigma_{n+1})) = (\prod f)_1(I(A)).$$

Let $\epsilon_{A,\sigma_{n+1}}$ denote the appropriate component of the counit for $\pi^* \dashv \prod$. Then

$$(\epsilon_{A,\sigma_{n+1}} x, \epsilon_{A,\sigma_{n+1}} y) \in f_1(I(A), I(\sigma_{n+1})) = I(f_0(A, \sigma_{n+1})),$$

so $\epsilon_{A,\sigma_{n+1}} x = \epsilon_{A,\sigma_{n+1}} y$. Since $(\prod f)_0(A)$ is the product of $f_0(A,\sigma_{n+1})$ over σ_{n+1} in $\mathbf{C}_0$, and $\epsilon_{A,\sigma_{n+1}}$ is simply the projection onto the σ_{n+1}-component, $\epsilon_{A,\sigma_{n+1}} x = \epsilon_{A,\sigma_{n+1}} y$ for all σ_{n+1} implies $x = y$ as desired.

Finally we will show that $\prod_{par}$ provides the desired right adjoint. A morphism from (g_0, g_1) to (h_0, h_1), where

$$\begin{tikzcd}
\Xi' \arrow[r, "g_1"] \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] & \mathbf{LR_Q(C)}_0 \arrow[d, shift right=2, "\partial_0"'] \arrow[d, shift left=2, "\partial_1"] \\
\Xi \arrow[r, "g_0"] \arrow[u, "I" description] & \mathbf{C}_0 \arrow[u, "I" description]
\end{tikzcd}$$

and likewise (h_0, h_1) is a morphism $s\colon g_0 \to h_0$ preserving relations (see Remark 4.4 below). In the internal language this means that for each $A \in \Xi$ we have a map $s_A\colon g_0(A) \to h_0(A)$ such that for R with $\partial_0(R) = A, \partial_1(R) = B$, $(x,y) \in g_1(R)$ implies $(s_A(x), s_B(y)) \in h_1(R)$.

Now, from Lemma 4.2 we easily derive a bijection between maps $(g_0, g_1) \to ((\prod f)_0, (\prod f)_1)$ and maps $(g_0 \circ \pi, g_1 \circ \pi) \to (f_0, f_1)$. Since $\prod_{par} f_0(A) \subseteq (\prod f)_0(A)$, if $s\colon (g_0, g_1) \to ((\prod_{par} f)_0, (\prod_{par} f)_1)$ is a map then clearly the correspondence gives a map $\tilde{s}\colon (g_0 \circ \pi, g_1 \circ \pi) \to (f_0, f_1)$. On the other hand, if we have a map $s\colon (g_0 \circ \pi, g_1 \circ \pi) \to (f_0, f_1)$ then a priori $\tilde{s}\colon (g_0, g_1) \to ((\prod f)_0, (\prod f)_1)$ and we need to show that for each A, the image of $\tilde{s}_A$ is contained in $(\prod_{par} f)_0(A)$. So suppose $x \in g_0(A)$. Since $(x, x) \in g_1(I(A)) = I(g_0(A))$, we must have $(\tilde{s}(x), \tilde{s}(x)) \in (\prod f)_1(I(A))$, so $\tilde{s}(x) \in \prod_{par} f_0(A)$ as desired. □

Remark 4.4. *Consider a morphism ξ between types $f = (f_0, f_1)$ and $g = (g_0, g_1)$ in the model*

$$\mathrm{Fam}\left(\begin{tikzcd}[row sep=small] \mathbf{LR_Q(C)} \arrow[d, shift right=2] \arrow[d, shift left=2] \\ \mathbf{C} \arrow[u] \end{tikzcd}\right) \to \mathbb{E}^G.$$

At first sight, such a morphism is a pair of morphism (ξ_0, ξ_1) with $\xi_i : f_i \to g_i$. But morphisms in $\mathbf{LR_Q}(\mathbf{C})$ are given by pairs of maps in $\mathbf{C}$, and commutativity of

$$\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C})_0^n \arrow[r, "\xi_1"] \arrow[d, "\partial_i"'] & \mathbf{LR_Q}(\mathbf{C})_1 \arrow[d, "\partial_i"] \\ \mathbf{C}_0^n \arrow[r, "\xi_0"] & \mathbf{C}_1 \end{tikzcd}$$

tells us that ξ_1 must be given by (ξ_0, ξ_0). Thus morphisms between types are morphisms between the usual interpretations of types preserving the relational interpretations.

Lemma 4.5. *The category $\mathbf{LR_Q}(\mathbf{C})$ is an internal linear category with products and this structure commutes with the maps of (5).*

Proof. The fibred linear structure on

$$\mathbf{LinAdmRelations_C} \to \mathbf{AdmRelCtx_C}$$

gives a fibred linear structure on

$$\mathrm{Fam}\left(\begin{tikzcd}[column sep=small, row sep=small] & \mathbf{LR_Q(C)} \arrow[dl] \arrow[dr] & \\ \mathbf{C} & & \mathbf{C} \end{tikzcd}\right) \to \mathbb{E}^\Lambda$$

using Lemma 4.1. By Lemma 2.7 and Lemma 2.8 we get linear structures on $\mathbf{LR_Q}(\mathbf{C})$ and $\mathbf{C}$ commuting with the domain and codomain functors.

To see that $\otimes, \multimap, !$ all preserve identities we first notice that these constructions can be written out in the internal logic of $\mathbb{E}$. Suppose $\rho\colon \mathbf{AdmRel}(\sigma, \tau)$, $\rho'\colon \mathbf{AdmRel}(\sigma', \tau')$ then

$$\begin{array}{rcl} !\rho & = & (x\colon\ !\sigma, y\colon !\tau).\, x \downarrow \supset\!\subset y \downarrow \wedge x \downarrow \supset \rho(\epsilon x, \epsilon y) \\ \rho \multimap \rho' & = & (f\colon \sigma \multimap \sigma', g\colon \tau \multimap \tau').\, \forall x\colon \sigma.\, \forall y\colon \tau.\, \rho(x, y) \supset \rho'(f(x), g(y)) \\ \rho \otimes \rho' & = & (f_{\sigma,\sigma'}, f_{\tau,\tau'})^*(\forall(\alpha, \beta, R\colon \mathbf{AdmRel}(\alpha, \beta)).\, (\rho \multimap \rho' \multimap R) \multimap R) \end{array}$$

for the natural transformation

$$f_{\sigma,\tau} : \sigma \otimes \tau \multimap \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

defined as

$$f_{\sigma,\tau}\, x = \text{let } x' \otimes x''\colon \sigma \otimes \tau \text{ be } x \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\, x'\, x''.$$

Now, one can easily prove that ! and $\multimap$ preserve equalities, using Axiom 2.18 of [2] for the case of !.

We proceed to show that $eq_\sigma \otimes eq_\tau$ is the equality on $\sigma \otimes \tau$ using the Yoneda lemma. Suppose we are given an admissible relation $R\colon \mathbf{AdmRel}(\omega, \omega')$. Maps

$$(f, g)\colon eq_\sigma \otimes eq_\tau \multimap R$$

in $\mathbf{LR_Q}(\mathbf{C})$ correspond to maps

$$(\hat{f}, \hat{g})\colon eq_\sigma \multimap eq_\tau \multimap R.$$

Since R is simply a subobject of $\omega \times \omega'$ in the category $\mathbf{C}$, such maps correspond to

$$\langle \hat{f}, \hat{g} \rangle\colon \sigma \multimap \tau \multimap R$$

in $\mathbf{C}$. Such maps correspond to maps

$$\widehat{\langle \hat{f}, \hat{g} \rangle}\colon \sigma \otimes \tau \multimap R$$

still in $\mathbf{C}$, which correspond to maps

$$(f, g)\colon eq_{\sigma \otimes \tau} \multimap R$$

in $\mathbf{LR_Q}(\mathbf{C})$. By the Yoneda Lemma, $eq_\sigma \otimes eq_\tau$ is isomorphic to $eq_{\sigma\otimes\tau}$ in $\mathbf{LR_Q}(\mathbf{C})$, and by inspection of the correspondence provided above, this isomorphism is given by $(id_{\sigma\otimes\tau}, id_{\sigma\otimes\tau})$, which means that the two relations are equivalent.

The products are defined as

$$\rho \times \rho' = (x\colon \sigma \times \sigma', y\colon \tau \times \tau').\, \rho(\pi(x), \pi(y)) \wedge \rho'(\pi'(x), \pi'(y)),$$

where π, π' denote first and second projection respectively. This product clearly also commutes with domain and codomain maps and preserves equalities. □

It is interesting to notice that in the above proof, the argument for $\otimes$ preserving identities was not purely logical, but used the fact that admissible relations corresponded to subobjects in $\mathbf{C}$.

Combining Lemmas 2.8,4.5 we get the following lemma.

Lemma 4.6. *The reflexive graph of internal categories in* $\mathbb{E}$

$$\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l] \end{tikzcd}$$

constitutes an internal linear category in $\mathbb{E}^G$.

Remark 4.7. *Lemmas 4.3,4.6 together prove that the fibration*

$$\mathrm{Fam}\left(\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l] \end{tikzcd} \right) \to \mathbb{E}^G$$

models all of PILL$_Y$ *except* Y *(we show that* Y *is modeled in Lemma 4.9 below). Types with* n *free variables are modeled as pairs of maps* $([\![\vec{\alpha} \vdash \sigma]\!]_1, [\![\vec{\alpha} \vdash \sigma]\!]_0)$*:*

$$\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C})_0^n \arrow[r, "{[\![\vec{\alpha}\vdash\sigma]\!]_1}"] \arrow[d, shift left=2] \arrow[d, shift right=2] & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[d, shift left=2] \arrow[d, shift right=2] \\ \mathbf{C}_0 \arrow[r, "{[\![\vec{\alpha}\vdash\sigma]\!]_0}"] \arrow[u] & \mathbf{C}_0 \arrow[u] \end{tikzcd}$$

making the obvious 3 squares commute. Let us denote by $[\![\vec{\alpha} \vdash \sigma]\!]$ *the interpretation of* $\vec{\alpha} \vdash \sigma$ *in the fibration* $\mathrm{Fam}(\mathbf{C}) \to \mathbb{E}$ *and compare this to* $[\![-]\!]_0$. *From the definitions above, it is clear that the constructions* $\multimap, !, \otimes$ *are modeled the same way in* $[\![-]\!]_0$ *and* $[\![-]\!]$*, but the interpretation of* $\prod \alpha.\, (-)$ *is different in the two. For example*

$$[\![\alpha \vdash (\alpha \to \alpha) \to \alpha]\!]_0 = [\![\alpha \vdash (\alpha \to \alpha) \to \alpha]\!]$$

but

$$[\![\textstyle\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]_0 = \{x \colon [\![\textstyle\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!] \mid x(\forall \alpha, \beta, R \colon \mathbf{AdmRel}(\alpha, \beta).\, (R \to R) \to R)x\}$$

corresponding to our intuition that the parametric completion process should restrict polymorphic types to parametric elements. From the proof of Lemma 4.3 we see that type application is modeled the same way in $[\![-]\!]$ *and* $[\![-]\!]_0$.

Notice also that closed types in the model

$$\mathrm{Fam}\left(\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l] \end{tikzcd}\right) \to \mathbb{E}^G$$

are given by their σ_0 *component, since we have required* $\sigma_1 = I(\sigma_0)$.

Lemma 4.6 shows in particular that we have a comonad ! on $\mathbf{LR_Q}(\mathbf{C})$, and so we can form the co-Kleisli category $\mathbf{LR_Q}(\mathbf{C})_!$ as the internal category with $\mathbf{LR_Q}(\mathbf{C})_0$ as object of objects and with object of morphisms defined by the pull-back:

$$\begin{tikzcd} (\mathbf{LR_Q}(\mathbf{C})_!)_1 \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbf{LR_Q}(\mathbf{C})_1 \arrow[d] \\ \mathbf{LR_Q}(\mathbf{C})_0 \times \mathbf{LR_Q}(\mathbf{C})_0 \arrow[r, "! \times id"] & \mathbf{LR_Q}(\mathbf{C})_0 \times \mathbf{LR_Q}(\mathbf{C})_0 \end{tikzcd}$$

Lemma 4.8. *The co-Kleisli category for the comonad* ! *on* $\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C} \arrow[l] \end{tikzcd}$ *inside* $\mathbb{E}^G$ *is isomorphic to*

$$\begin{tikzcd} \mathbf{LR_Q}(\mathbf{C})_! \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{C}_! \arrow[l] \end{tikzcd}$$

Proof. The co-Kleisli category is constructed pointwise. □

Lemma 4.9. *The schema*

$$\begin{tikzcd} \mathrm{Fam}\left(\begin{smallmatrix} \mathbf{LR_Q}(\mathbf{C}) \\ \downarrow\uparrow\downarrow \\ \mathbf{C} \end{smallmatrix}\right) \arrow[rr, bend right=10] \arrow[dr] & \perp & \mathrm{Fam}\left(\begin{smallmatrix} \mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_! \end{smallmatrix}\right) \arrow[ll, bend right=10] \arrow[dl] \\ & \mathbb{E}^G & \end{tikzcd}$$

is a PILL$_Y$*-model.*

Proof. The only thing still to prove is that it models Y. Recall the computation of the interpretation of $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$ from Remark 4.7. Since $[\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]_0$ is a subtype of $[\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]$ we may ask if $Y \in [\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]_0$. This is true, since we have required that

$$Y(\forall \alpha, \beta, R \colon \mathbf{AdmRel}(\alpha, \beta).\, (R \to R) \to R)Y.$$

From the proof of Lemma 4.3 we see that type instantiation is interpreted the same way in $[\![-]\!]$ and $[\![-]\!]_0$, and so the term

$$\alpha \mid f \colon \alpha \to \alpha \vdash Y\,\alpha\,!f$$

is interpreted equally in the two interpretations. Validity of

$$\alpha \mid f \colon \alpha \to \alpha \vdash f\,!(Y\,\alpha\,!f) = (Y\,\alpha\,!f)$$

in the model

$$\begin{tikzcd}
\mathrm{Fam}(\,\mathbf{LR_Q}(\mathbf{C}) \arrow[r, shift left=2] \arrow[r, leftarrow] \arrow[r, shift right=2] & \mathbf{C}\,) \arrow[r] & \mathbb{E}^G
\end{tikzcd}$$

thus follows from validity of the same in

$$\mathrm{Fam}(\mathbf{C}) \to \mathbb{E}.$$

□

Consider the functor $(-)_0 \colon \mathbb{E}^G \to \mathbb{E}$ defined by mapping

$$\begin{tikzcd}
\Xi_1 \arrow[r, shift left=2] \arrow[r, leftarrow] \arrow[r, shift right=2] & \Xi_0
\end{tikzcd}$$

to Ξ_0. We define the categories $\mathbb{C}$ and $\mathbb{P}$ by the pullbacks

$$\begin{tikzcd}
\mathbb{P} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{Q} \arrow[d] \\
\mathbb{C} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{E}^{\to} \arrow[d] \\
\mathbb{E}^G \arrow[r, "(-)_0"] & \mathbb{E}
\end{tikzcd}$$

Lemma 4.10. *The composable fibration $\mathbb{P} \to \mathbb{C} \to \mathbb{E}^G$ is an indexed first-order logic fibration with an indexed family of generic objects. Moreover, the composable fibration has simple products, simple coproducts and very strong equality.*

Proof. The composable fibration $\mathbb{P} \to \mathbb{C} \to \mathbb{E}^G$ is a pullback of $\mathbb{Q} \to \mathbb{E}^{\to} \to \mathbb{E}$ which has the desired properties. All of this structure is always preserved under pullback, except simple products and coproducts. These are preserved since the map $(-)_0$ preserves products. □

Consider the map into the pullback

$$\begin{tikzcd}
\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix} \arrow[r] \arrow[dr] & (-)_0^*(\mathrm{Fam}\,(\mathbf{C}_!)) \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathrm{Fam}(\mathbf{C}_!) \arrow[d] \\
& \mathbb{E}^G \arrow[r, "(-)_0"] & \mathbb{E}
\end{tikzcd}$$

given by the map, that maps $\begin{pmatrix}\Xi_1 \\ \downarrow\uparrow\downarrow \\ \Xi_0\end{pmatrix} \to \begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix}$ in $\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix}$ to $\Xi_0 \to \mathbf{C}_!$ in $\mathrm{Fam}(\mathbf{C}_!)$. We define the map I:

$$\begin{tikzcd}
\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix} \arrow[r, "I"] \arrow[dr] & \mathbb{C} \arrow[d] \\
& \mathbb{E}^G
\end{tikzcd}$$

to be the composition of this map with the pullback of the inclusion of $\mathrm{Fam}(\mathbf{C}_!)$ into $\mathbb{E}^{\to}$. One could also express this definition as the map that maps

$$\begin{tikzcd}
\Xi_1 \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C})_! \arrow[d, shift right=2] \arrow[d, shift left=2] \\
X_0 \arrow[r, "f_0"] \arrow[u] & \mathbf{C}_! \arrow[u]
\end{tikzcd}$$

to $\phi(f_0)$, where ϕ is the inclusion of $\mathrm{Fam}(\mathbf{C}_!)$ into $\mathbb{E}^{\to}$.

Lemma 4.11. *The diagram*

$$\begin{tikzcd}
 & & \mathbb{P} \arrow[d] \\
\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C}) \\ \downarrow\uparrow\downarrow \\ \mathbf{C}\end{pmatrix} \arrow[r, shift right] \arrow[rrd] & \mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix} \arrow[l, shift right] \arrow[r, "I"] \arrow[rd] & \mathbb{C} \arrow[d] \\
 & & \mathbb{E}^G
\end{tikzcd}$$

is a pre-LAPL-structure.

Proof. Using Lemma 4.10 we see that all we need to prove is that $\mathbb{C} \to \mathbb{E}^G$ has fibred products, that I is faithful and product preserving and that the functor U exists. The first follows from $\mathbb{E}^{\to} \to \mathbb{E}$ having fibred products.

Recall from Remark 4.4 that a map in $\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix}$ is a natural transformation preserving relations and the functor from $\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_! \\ \downarrow\uparrow\downarrow \\ \mathbf{C}_!\end{pmatrix}$ into $(-)_0^*(\mathrm{Fam}\,(\mathbf{C}_!))$ is simply the identity on maps. Since also the inclusion of $\mathrm{Fam}\,(\mathbf{C}_!)$ into $\mathbb{E}^{\to}$ is assumed faithful, I is faithful. Since the inclusion of $\mathbf{C}_!$ into $\mathbb{E}$ is required to preserve products for all internal PILL$_Y$-models, I preserves products.

The functor U is defined using the functor U of Proposition 3.3 as the composition

$$\begin{tikzcd}
\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C}) \\ \downarrow\uparrow\downarrow \\ \mathbf{C}\end{pmatrix}^2 \arrow[r] \arrow[rd] & (-)_0^*(\mathrm{Fam}(\mathbf{C}))^2 \arrow[r, "(-)_0^*U"] \arrow[d] & \mathbb{C} \arrow[ld] \\
 & \mathbb{E}^G &
\end{tikzcd}.$$

In words, U maps an object of $\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q}(\mathbf{C}) \\ \downarrow\uparrow\downarrow \\ \mathbf{C}\end{pmatrix}^2$ (square taken fibrewise) given by the maps

$$\begin{tikzcd}
\Xi_1 \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C}) \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0 \arrow[r, "f_0"] \arrow[u] & \mathbf{C} \arrow[u]
\end{tikzcd}, \quad
\begin{tikzcd}
\Xi_1 \arrow[r, "g_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C}) \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0 \arrow[r, "g_0"] \arrow[u] & \mathbf{C} \arrow[u]
\end{tikzcd}$$

to $\coprod_{x \in \Xi_0} (\mathbf{RegSub}_{\mathbb{E}})_{f_0(x) \times g_0(x)} \to \Xi_0$. □

Consider the subfunctor V of U defined by mapping an object $((f_0, f_1), (g_0, g_1))$ in $\mathrm{Fam}\begin{pmatrix}\mathbf{LR_Q(C)} \\ \downarrow\uparrow\downarrow \\ \mathbf{C}\end{pmatrix}^2$ to $\coprod_{x\in\Xi_0} \mathbf{Q}'_{f_0(x)\times g_0(x)} \to \Xi_0$.

Lemma 4.12. *The functor V defines a notion of admissible relations for the pre-LAPL-structure of Lemma 4.11.*

Proof. All terms occurring in the rules for admissible relations and in Axiom 2.18 are constructed without use of type abstraction. Thus the terms are interpreted exactly as in the pre-LAPL-structure of Proposition 3.3. Since the logic in the models of Proposition 3.3 and Lemma 4.11 are the same, all relations occurring in the rules and Axiom 2.18 are interpreted equally. Since also the notion of admissible relations is the same in the two models, the Lemma follows since we have assumed that the model of Proposition 3.3 models admissible relations. □

Consider the graph W:

$$\begin{tikzcd}
\cdot \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \cdot \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
& \cdot \arrow[dl] \arrow[dr] & \\
\cdot \arrow[uu] & & \cdot \arrow[uu]
\end{tikzcd}$$

where we assume that the two graphs included are reflexive graphs. The graph $\mathbf{W}$:

$$\begin{tikzcd}
\mathbf{LR_Q(C)} \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \mathbf{LR_Q(C)} \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
& \mathbf{LR_Q(C)} \arrow[dl] \arrow[dr] & \\
\mathbf{C} \arrow[uu] & & \mathbf{C} \arrow[uu]
\end{tikzcd}$$

defines an internal category in $\mathbb{E}^W$. By Lemma 2.8 $\mathbf{W}$ is an internal linear category with structure computed pointwise.

We denote by

$$\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$$

the fibration of admissible relations based on the pre-LAPL-structure constructed in Lemma 4.11.

Proposition 4.13. *There is an isomorphism of fibrations:*

$$\begin{pmatrix} \mathrm{Fam}(\mathbf{W}) \\ \Big\downarrow \\ \mathbb{E}^W \end{pmatrix} \xrightarrow{\cong} \begin{pmatrix} \mathbf{LinAdmRelations} \\ \Big\downarrow \\ \mathbf{AdmRelCtx} \end{pmatrix}$$

preserving the fibred linear category structure.

Proof. An object of $\mathbf{AdmRelCtx}$ is a pair of objects of $\mathbb{E}^G$:

$$\begin{tikzcd}
\Xi_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi_0 \arrow[l] \ , & \Xi'_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi'_0 \arrow[l]
\end{tikzcd}$$

plus an object of $\mathbb{E}^{\to}$ with domain $\Xi_0 \times \Xi'_0$, i.e. a map $\Xi_2 \to \Xi_0 \times \Xi'_0$ in $\mathbb{E}$. A map in $\mathbf{AdmRelCtx}$ from

$$\begin{tikzcd}
(\ \Xi_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi_0 \arrow[l] \ , \ \Xi'_1 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi'_0 \arrow[l] \ , a \colon \Xi_2 \to \Xi_0 \times \Xi'_0)
\end{tikzcd}$$

to

$$\begin{tikzcd}
(\ \Xi_4 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi_3 \arrow[l] \ , \ \Xi'_4 \arrow[r, shift left=2] \arrow[r, shift right=2] & \Xi'_3 \arrow[l] \ , b \colon \Xi_5 \to \Xi_3 \times \Xi'_3)
\end{tikzcd}$$

is a pair of maps in $\mathbb{E}^G$, i.e., a quadruple of maps (f_0, f_1, f_0', f_1') such that

$$\begin{tikzcd}
\Xi_1 \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \Xi_4 \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0 \arrow[r, "f_0"] \arrow[u] & \Xi_3 \arrow[u]
\end{tikzcd} , \qquad
\begin{tikzcd}
\Xi_1' \arrow[r, "f_1'"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \Xi_4' \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0' \arrow[r, "f_0'"] \arrow[u] & \Xi_3' \arrow[u]
\end{tikzcd}$$

both commute, plus a vertical map in $\mathbb{E}^{\to} \to \mathbb{E}$ over $\Xi_0 \times \Xi_1$:

$$\begin{tikzcd}
\Xi_2 \arrow[rr, "h"] \arrow[dr, "a"'] & & (f_0 \times f_0')^*\Xi_5 \arrow[dl, "(f_0 \times f_0')^* b"] \\
& \Xi_0 \times \Xi_0'. &
\end{tikzcd}$$

Since the map h corresponds to a map h' making

$$\begin{tikzcd}
& \Xi_2 \arrow[rrr, "h'"] \arrow[dl] \arrow[dr] & & & \Xi_5 \arrow[dl] \arrow[dr] & \\
\Xi_0 \arrow[rrr, "f_0"] & & \Xi_0' \arrow[rrr, "f_0'"'] & \Xi_3 & & \Xi_3'
\end{tikzcd}$$

commute, we get the isomorphism $\mathbf{AdmRelCtx} \cong \mathbb{E}^W$.

An object of $\mathbf{LinAdmRelations}$ over

$$(\; \Xi_1 \mathrel{\substack{\longrightarrow \\ \longleftarrow \\ \longrightarrow}} \Xi_0 \; , \; \Xi_1' \mathrel{\substack{\longrightarrow \\ \longleftarrow \\ \longrightarrow}} \Xi_0' \; , a \colon \Xi_2 \to \Xi_0 \times \Xi_0')$$

is a pair of types, i.e., maps (f_0, f_1, f_0', f_1') such that

$$\begin{tikzcd}
\Xi_1 \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0 \arrow[r, "f_0"] \arrow[u] & \mathbf{C}_0 \arrow[u]
\end{tikzcd} \qquad
\begin{tikzcd}
\Xi_1' \arrow[r, "f_1'"] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[d, shift right=2] \arrow[d, shift left=2] \\
\Xi_0' \arrow[r, "f_0'"] \arrow[u] & \mathbf{C} \arrow[u]
\end{tikzcd}$$

commute, plus a map ρ:

$$\begin{tikzcd}
\Xi_2 \arrow[rr, "\rho"] \arrow[dr, "a"'] & & \coprod_{x \in \Xi_0, y \in \Xi_0'} (\mathbf{RegSub}_{\mathbb{E}})_{f_0(x) \times f_0'(x)} \arrow[dl] \\
& \Xi_0 \times \Xi_0' &
\end{tikzcd}$$

Since ρ corresponds to a map ρ':

$$\begin{tikzcd}
& \Xi_2 \arrow[rrr, "\rho'"] \arrow[dl] \arrow[dr] & & & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[dl] \arrow[dr] & \\
\Xi_0 \arrow[rrr, "f_0"] & & \Xi_0' \arrow[rrr, "f_0'"'] & \mathbf{C}_0 & & \mathbf{C}_0
\end{tikzcd}$$

we get the bijective correspondence between objects of $\mathbf{LinAdmRelations}$ and objects of $\mathrm{Fam}(\mathbf{W})$. This correspondence extends to morphisms, since vertical morphism in both fibrations correspond to pairs of morphisms preserving relations.

The isomorphism preserves the fibred linear structure on the nose, since in both fibrations, the fibred linear structure is defined using the internal linear structure on $\mathbf{C}$ and $\mathbf{LR_Q}(\mathbf{C})$ respectively. □

Lemma 4.14. *The graph* $\mathbf{W}$ *models polymorphism.*

Proof. This is a consequence of Proposition 4.13. □

Proposition 4.15. *There is a reflexive graph of fibred linear categories*

$$\begin{tikzcd}
\left(\begin{array}{c}\mathrm{Fam}\left(\begin{array}{c}\mathbf{LR_Q}(\mathbf{C})\\ \downarrow\uparrow\downarrow\\ \mathbf{C}\end{array}\right)\\ \Bigg\downarrow \\ \mathbb{E}^G\end{array}\right) \arrow[r, leftarrow, shift left=3] \arrow[r] \arrow[r, leftarrow, shift right=3] & \left(\begin{array}{c}\mathrm{Fam}(\mathbf{W})\\ \Bigg\downarrow \\ \mathbb{E}^W\end{array}\right)
\end{tikzcd}.$$

The 3 *maps preserve products in the base, generic object and simple products.*

Comparing with Proposition 2.9 of [7] the maps of Proposition 4.15 give rise to a reflexive graph of maps between the corresponding PILL_Y-models.

Remark 4.16. *The reflexive graph in [10] arises this way, although the setup of [10] is slightly different.*

Proof. An object of $\mathrm{Fam}(\mathbf{W})$ is a map in $\mathbb{E}^W$

$$\left(\begin{tikzcd}
\Xi_1 \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \Xi_4 \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
& \Xi_3 \arrow[dl] \arrow[dr] & \\
\Xi_2 \arrow[uu] & & \Xi_5 \arrow[uu]
\end{tikzcd}\right) \to \left(\begin{tikzcd}
\mathbf{LR_Q}(\mathbf{C})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] & & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[dd, shift right=2] \arrow[dd, shift left=2] \\
& \mathbf{LR_Q}(\mathbf{C})_0 \arrow[dl] \arrow[dr] & \\
\mathbf{C}_0 \arrow[uu] & & \mathbf{C}_0 \arrow[uu]
\end{tikzcd}\right).$$

Let us denote such objects as triples (f, g, ρ) where

$$f = (f_0, f_1)\colon \begin{pmatrix}\Xi_1\\ \downarrow\uparrow\downarrow\\ \Xi_2\end{pmatrix} \to \begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_0\\ \downarrow\uparrow\downarrow\\ \mathbf{C}_0\end{pmatrix}, \quad g = (g_0, g_1) : \begin{pmatrix}\Xi_4\\ \downarrow\uparrow\downarrow\\ \Xi_5\end{pmatrix} \to \begin{pmatrix}\mathbf{LR_Q}(\mathbf{C})_0\\ \downarrow\uparrow\downarrow\\ \mathbf{C}_0\end{pmatrix}$$

and $\rho : \Xi_3 \to \mathbf{LR_Q}(\mathbf{C})_0$. The domain and codomain maps of the postulated reflexive graph map (f, g, ρ) to f and g respectively, and the last map maps f to (f, f, f_1). Clearly generic objects and products in the basecategories are preserved, and since the linear category structure is computed pointwise in both fibrations, it is clearly preserved by all maps.

We now show that all maps preserve simple products. The domain and codomain maps preserve simple products since from the viewpoint of Proposition 4.13 these are just the domain and codomain maps out of

$$\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}.$$

Consider the map going the other way. Mapping f along this map and then taking products gives us the map that — described in the internal language of the topos $\mathbb{E}$ — maps $\vec{R}\colon \mathrm{AdmRel}(\vec{A}, \vec{B})$ to

$$\{(x, y) \in (\textstyle\prod f)_0(\vec{A}) \times (\prod f)_0(\vec{B}) \mid \forall A, B\colon \mathbf{C}_0. \forall R\colon \mathrm{AdmRel}(A, B).\, f_1(\vec{R}, R)(x_A, y_B)\}$$

where $(\prod f)_0$ denotes the type-component of the simple product $\prod f$ (called $\prod_{par} f$ in the proof of Lemma 4.3) taken in $\mathrm{Fam}(\ \mathbf{LR_Q}(\mathbf{C}) \overset{\longrightarrow}{\underset{\longrightarrow}{\longleftarrow}} \mathbf{LR_Q}(\mathbf{C})\ ) \to \mathbb{E}^G$. This map coincides with $(\prod f)_1$, the relational interpretation of $\prod f$ as desired. □

Proposition 4.17. *The pre-LAPL-structure of Lemma 4.11 has relational interpretation of all types.*

Proof. This follows from Proposition 4.15 and Proposition 4.13. □

Lemma 4.18. *The LAPL-structure of Lemma 4.11 satisfies extensionality.*

Proof. The model has very strong equality, which implies extensionality. □

Lemma 4.19. *The LAPL-structure of Lemma 4.11 satisfies the identity extension axiom.*

Proof. Consider a type $f = (f_1, f_0)$:

$$\begin{tikzcd}
\mathbf{LR_Q}(\mathbf{C})_0^n \arrow[r, "f_1"] \arrow[d, shift right=2] \arrow[d, leftarrow] \arrow[d, shift left=2] & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[d, shift right=2] \arrow[d, leftarrow] \arrow[d, shift left=2] \\
\mathbf{C}_0^n \arrow[r, "f_0"] & \mathbf{C}_0
\end{tikzcd} \tag{7}$$

with n free variables. We need to show that

$$\langle id_{\Omega^n}, id_{\Omega^n}\rangle^* J(f) \circ [\![\vec{\alpha} \mid - \mid - \vdash eq_{\vec{\alpha}}]\!] = [\![\vec{\alpha} \vdash \boldsymbol{eq}_{f(\vec{\alpha})}]\!].$$

The map J is defined as the composition of two maps. The first map maps f to (f, f, f_1) :

$$\left(\begin{tikzcd}[column sep=small]
\mathbf{LR_Q}(\mathbf{C})_0^n \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] & & \mathbf{LR_Q}(\mathbf{C})_0^n \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] \\
& \mathbf{LR_Q}(\mathbf{C})_0^n \arrow[dl] \arrow[dr] & \\
\mathbf{C}_0^n & & \mathbf{C}_0^n
\end{tikzcd}\right) \to \left(\begin{tikzcd}[column sep=small]
\mathbf{LR_Q}(\mathbf{C})_0 \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] & & \mathbf{LR_Q}(\mathbf{C})_0 \arrow[dd, shift right=2] \arrow[dd, leftarrow] \arrow[dd, shift left=2] \\
& \mathbf{LR_Q}(\mathbf{C})_0 \arrow[dl] \arrow[dr] & \\
\mathbf{C}_0 & & \mathbf{C}_0
\end{tikzcd}\right)$$

and the second identifies this with an element of $\mathbf{LinAdmRelations}$, which in the internal language of the LAPL-structure may be written as

$$[\![\vec{\alpha}, \vec{\beta} \mid - \mid \vec{R} \colon \mathrm{AdmRel}(\vec{\alpha}, \vec{\beta}) \vdash f_1(\vec{R}) \colon \mathrm{AdmRel}(f(\vec{\alpha}), f(\vec{\beta}))]\!]$$

Since f makes the diagram (7) commute we conclude that

$$\langle id_{\Omega^n}, id_{\Omega^n}\rangle^* J(f) \circ [\![\vec{\alpha} \mid - \mid - \vdash eq_{\vec{\alpha}}]\!] =$$
$$[\![\vec{\alpha} \mid - \mid \vec{R} \colon \mathrm{AdmRel}(\vec{\alpha}, \vec{\alpha}) \vdash f_1(\vec{R}) \colon \mathrm{AdmRel}(f(\vec{\alpha}), f(\vec{\alpha}))]\!] \circ [\![\vec{\alpha} \mid - \mid - \vdash eq_{\vec{\alpha}}]\!] = [\![\vec{\alpha} \vdash \boldsymbol{eq}_{f(\vec{\alpha})}]\!].$$

□

Summing up we have:

Theorem 4.20. *The pre-LAPL-structure of Lemma 4.11 is a parametric LAPL-structure.*

5 Examples

For any reflexive domain D, one can form the category of admissible pers $\mathbf{AP}(D)$ and the category of admissible pers with maps tracked by strict trackers $\mathbf{AP}(D)_\perp$ as in [2]. As is well-known, the category of pers is an internal subcategory of the category of assemblies $\mathbf{Asm}(D)$ over D, and using the same construction one may easily show that $\mathbf{AP}(D)$ and $\mathbf{AP}(D)_\perp$ are internal subcategories of $\mathbf{Asm}(D)$. In fact $\mathbf{AP}(D)_\perp$ is an internal PILL$_Y$- model in the quasi-topos $\mathbf{Asm}(D)$ with co-Kleisli category $\mathbf{AP}(D)$.

The category of regular subobjects of admissible pers internalizes to an internal fibration

$$\mathbf{RegSub}_{\mathbf{AP}(D)_\perp} \to \mathbf{AP}(D)_\perp$$

which we may use for a notion of admissible relations. Applying the completion process to this structure, we obtain the LAPL-structure:

$$\begin{tikzcd}
& & \mathbf{UFam}(\mathrm{RegSub}_{\mathbf{Asm}(D)}) \arrow[d] \\
\mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[r, shift right] \arrow[rrd] & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[l, shift right] \arrow[r] \arrow[rd] & \mathbf{UFam}(\mathbf{Asm}(D)) \arrow[d] \\
& & \mathbf{PAP}(D).
\end{tikzcd} \tag{8}$$

The PILL$_Y$-model on the left is the PILL$_Y$ model of [2]. The fibre of

$$\mathbf{UFam}(\mathbf{Asm}(D)) \to \mathbf{PAP}(D)$$

over an object n has as objects maps $\mathbf{AP}(D)^n \to \mathbf{Asm}(D)$ and as morphisms uniformly tracked morphisms between assemblies. The logic

$$\mathbf{UFam}(\mathrm{RegSub}_{\mathbf{Asm}(D)}) \to \mathbf{UFam}(\mathbf{Asm}(D))$$

is the fibration of families of regular subobjects of assemblies, i.e., a subobject of $f\colon \mathbf{AP}(D)^n \to \mathbf{Asm}(D)$ is a family of subsets $A_{\vec{R}} \subseteq | f(\vec{R}) |$, where $| - |$ is the forgetful functor from $\mathbf{Asm}(D)$ to $\mathbf{Set}$.

The LAPL-structure (8) is the LAPL-structure of [2] with the category of sets replaced by assemblies. The logic of the two are the same since we have a pullback

$$\begin{tikzcd}
\mathbf{UFam}(\mathrm{RegSub}_{\mathbf{Asm}(D)}) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{Sub}(\mathbf{Set}) \arrow[d] \\
\mathbf{UFam}(\mathbf{Asm}(D)) \arrow[r] & \mathrm{Fam}(\mathbf{Set}).
\end{tikzcd}$$

Therefore, even though the presentation is different, the LAPL-structure of [2] is basically the LAPL-structure obtained from parametric completion as presented in this paper.

5.1 The LAPL-structure from synthetic domain theory

The LAPL-structure from [11, 8, 9] is not directly an application of the parametric completion process presented in this paper. The logic is given by sets, and the PILL$_Y$-model is constructed using the category of domains, which is not small.

A natural way to view the LAPL-structure from SDT is to view it as coming from $\mathbf{Dom}_\perp$ as seen as an internal category in the category of (not necessarily small) groupoids via the following construction:

Consider the functor $(\cdot)_{\text{iso}}$ from the category of categories to the category of groupoids mapping a category to its restriction to isomorphisms. Suppose $\mathbb{C}$ is a category, then the diagram

$$\begin{tikzcd} \mathbb{C}^{\rightarrow}_{\text{iso}} \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbb{C}_{\text{iso}} \arrow[l] \end{tikzcd} \tag{9}$$

is an internal category in the category of groupoids. The category $\mathbb{C}^{\rightarrow}_{\text{iso}}$ has as objects arrows of $\mathbb{C}$ and as morphisms pairs of isomorphisms making the obvious square commute. The two left to right going maps map an arrow to its domain and codomain respectively and the last map maps an object to the identity on that object. This construction extends to a functor from the category of categories to the category of internal categories in the category of groupoids.

The externalization of (9) has as object over $\mathbb{C}^n_{\text{iso}}$ functors $\mathbb{C}^n_{\text{iso}} \to \mathbb{C}$ (since these are the same as functors $\mathbb{C}^n_{\text{iso}} \to \mathbb{C}_{\text{iso}}$) and as morphisms natural transformations.

Using the above construction on the category $\mathbf{Dom}_\perp$ of domains with strict morphisms, we obtain an internal PILL$_Y$-model in the category of groupoids. We may further apply the construction to the fibration of regular subobjects on $\mathbf{Dom}_\perp$. Using this as our notion of admissible relations, the PILL$_Y$-model constructed as in the parametric completion process is the model presented in [11, 8, 9].

The LAPL-structure of [9], however, is not derived from the internal logic in the category of groupoids. Instead, the category of contexts and the logic fibration in the LAPL-structure of *loc. cit.* is the externalization of

$$\text{Sub}(\mathbf{Set}) \to \mathbf{Set}$$

seen as an internal fibration in the category of groupoids using the construction above.

6 Conclusion

We have defined a notion of internal PILL$_Y$-model in a quasi-topos and shown how the externalization of an internal PILL$_Y$-model can be extended to a pre-LAPL-structure in which the logic is given by the regular subobject logic of the quasi-topos. This corresponds to the way one would usually think of parametricity for such internal models.

We have described a parametric completion process based on the parametric completion process of [10] which takes an internal PILL$_Y$-model in a quasi-topos and returns an internal PILL$_Y$- model in a presheaf-category over the original quasi-topos. The externalization of the resulting PILL$_Y$-model extends to a parametric LAPL-structure. This LAPL-structure is different from the canonical LAPL-structure associated to internal PILL$_Y$-models as mentioned above, and in fact the logic of the LAPL-structure is the logic of the original quasi-topos.

The concrete LAPL-structure of [2] is an example of this parametric completion process, although it is presented a bit different in *loc. cit.*. The PILL$_Y$-model constructed using synthetic domain theory in [11, 8, 9] is an example of an application of the parametric completion process, but the LAPL-structure provided for it in [8, 9] is different from the one presented here.

References

[1] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*. To appear. (document), 1, 2.1, 3, 4

[2] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of linear Abadi & Plotkin logic. Technical Report TR-2005-57, IT University of Copenhagen, February 2005. (document), 1, 3, 4, 5, 5, 6

[3] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. Submitted, 2005. 1

[4] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 2, 2.2

[5] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971. 2.2

[6] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic. A First Introduction to Topos Theory*. Springer, New York, 1992. 2.2

[7] R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005. 1, 2.2, 2.2, 3.4, 4

[8] R. E. Møgelberg, L. Birkedal, and G. Rosolini. Synthetic domain theory and models of linear Abadi & Plotkin logic. Technical Report TR-2005-59, IT University of Copenhagen, February 2005. (document), 1, 5.1, 5.1, 6

[9] R. E. Møgelberg, L. Birkedal, and G. Rosolini. Synthetic domain theory and models of linear Abadi & Plotkin logic. Submitted, 2005. (document), 1, 5.1, 5.1, 6

[10] E.P. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In S. Abramsky, editor, *Proc. 9th Symposium in Logic in Computer Science*, pages 364–371, Paris, 1994. I.E.E.E. Computer Society. (document), 1, 4.16, 6

[11] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. (document), 1, 5.1, 5.1, 6