# Synthetic Domain Theory and Models of Linear Abadi & Plotkin Logic

Rasmus Ejlers Møgelberg
Lars Birkedal
Giuseppe Rosolini

June 2007

**Abstract**

Plotkin suggested using a polymorphic dual intuitionistic / linear type theory ($\mathtt{PILL}_Y$) as a metalanguage for parametric polymorphism and recursion. In recent work the first two authors and R.L. Petersen have defined a notion of parametric LAPL-structure, which are models of $\mathtt{PILL}_Y$, in which one can reason using parametricity and, for example, solve a large class of domain equations, as suggested by Plotkin.

In this paper we show how an interpretation of a strict version of Bierman, Pitts and Russo's language `Lily` into synthetic domain theory presented by Simpson and Rosolini gives rise to a parametric LAPL-structure. This adds to the evidence that the notion of LAPL-structure is a general notion suitable for treating many different parametric models, and it provides formal proofs of consequences of parametricity expected to hold for the interpretation. Finally, we show how these results in combination with Rosolini and Simpson's computational adequacy result can be used to prove consequences of parametricity for `Lily`. In particular we show that one can solve domain equations in `Lily` up to ground contextual equivalence.

## 1 Introduction

It was first realized by Plotkin [21, 20] that $\mathtt{PILL}_Y$, a polymorphic type theory with linear as well as intuitionistic variables and fixed points, is a suitable metalanguage for the combination of parametric polymorphism and recursion. Plotkin showed how to encode a number of type constructors including initial algebras and final coalgebras in $\mathtt{PILL}_Y$, which by the existence of fixed points gave solutions for general recursive domain equations, as in Freyd's theory of algebraically compact categories [9, 8, 10]. 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, and where recursive domain equations are solved using polymorphism rather than the traditional limit-colimit construction.

Recently the first two authors together with R.L. Petersen have presented a variant of Abadi & Plotkin's logic for parametricity [22] suitable for reasoning about parametricity in $\mathtt{PILL}_Y$ and defined the categorical notion of parametric LAPL-structure (Linear Abadi-Plotkin Logic), which are models of the logic [5, 6]. Using Plotkin's constructions one can solve recursive domain equations in LAPL-structures. In [7] a concrete domain-theoretic LAPL-structure based on admissible pers over a reflexive domain is constructed, and in [18] a parametric completion process along the lines of [23] is presented constructing parametric LAPL-structures out of a large class of models of $\mathtt{PILL}_Y$.

In recent work Simpson and Rosolini [24] have constructed an interpretation of a strict version of `Lily` [3] — a language that we shall call $\mathtt{Lily}_{\mathtt{strict}}$ — based on Synthetic Domain Theory (SDT), and show the interpretation adequate. 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 [24] gives an interpretation of $\mathtt{Lily}_{\mathtt{strict}}$, but the advantage of the set-theoretic approach is that one does not have to know the details of these models to use the interpretation.

In this paper we present a parametric LAPL-structure based on the interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ of [24]. 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 $\mathtt{PILL}_Y$-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 [3] can be presented as reasoning in an LAPL-structure.

Our second motivation is that the interpretation presented in [24] 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 $\mathtt{Lily}_{\mathtt{strict}}$ to show formally consequences of parametricity for `Lily`. This builds upon the idea from [24] of giving denotational proofs of the theorems in [3], and extends it to prove properties not included in [3]. Among these results is the existence of solutions to recursive type equations in `Lily` up to ground contextual equivalence.

The paper is organized as follows. The first section discusses the model-theoretic setup and the linear structure of the category of domains with strict maps. In Sections 3-5 we present the LAPL-structure. We first present a model of $\mathtt{PILL}_Y$ 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 6 we show how to use the parametric LAPL-structure to reason about `Lily`.

**Acknowledgments.** We thank Alex Simpson for helpful discussions.

## 2 A linear category structure for synthetic domains

We start by recalling some of the setup of [24] on which we will base the constructions of this paper.

In [24] a set theoretic model of the language $\mathtt{Lily}_{\mathtt{strict}}$ is constructed. The basis of the construction is a collection of sets called predomains, which is assumed given and satisfying a number of axioms. In the setting of classical set theory, the axioms imply that no nontrivial set can be a predomain, but there do exist interesting models of the axioms in intuitionistic set theory. In the following, when we talk of a model of synthetic domain theory (SDT), we shall mean a model of intuitionistic set theory with a given collection of predomains satisfying the axioms of *loc. cit.* (although we later need to be a bit more advanced, see Remark 2.2).

Further, in [24] a notion of pointed set is defined, and a domain is defined to be a pointed predomain. Strict maps between pointed sets are maps that preserve the pointed structure and the notation $f\colon A \multimap B$ is used to indicate that a map is strict. We fix the notation $\mathbf{Dom}$ for the category of domains with all maps and $\mathbf{Dom}_\perp$ for the category of domains with strict maps. Referring to [24] for the details of the above mentioned definitions we recall the following consequences.

**Theorem 2.1** ([24])**.** *The following are consequences of the axioms for predomains.*

- *The category $\mathbf{Dom}_\perp$ is complete and cartesian closed with limits and exponentials computed as in $\mathbf{Set}$.*
- *The forgetful functor $\mathbf{Dom}_\perp \to \mathbf{Dom}$ has a left adjoint L.*
- *There exists a set $\mathbf{D}$ of domains such that any domain is isomorphic to a domain in $\mathbf{D}$ (via strict maps).*

In this paper, when constructing the model, we will work in intuitionistic set theory following the informal style of [24]. Rather than seeing the construction as a model in intuitionistic set theory, perhaps it is better to think of the construction as giving a family of models: for any given model of SDT, the construction gives a parametric LAPL-structure.

**Remark 2.2.** *For the constructions of this paper we will need a somewhat more advanced notion of model of SDT, than what is needed for the constructions of [24], since some of the constructions we need involve constructions on classes. Consider,* *for example, the category of domains with all set theoretic maps* $\mathbf{Dom}$*. As the collection of domains is a real class and not a set, the category of endofunctors on* $\mathbf{Dom}$ *need not be a class. As such constructions on classes will appear in the following, we sketch for the concerned reader how these issues may be resolved.*

*As 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 [12] as refined in [26] with the notion of classic structure on a regular category with a universe and a small natural numbers object. We further assume that the collections of morphisms between any two objects of the category of classes form a class in the external sense. Given such a setting, the category of domains is an internal category in the regular category of classes while the collection of all internal functors* $\mathbf{Dom} \to \mathbf{Dom}$ *is a class in the external sense, since it is a subclass of the class of morphisms of the category of classes. This way, the fibrations in Lemma 3.6 below are defined externally. The example mentioned in [24] of modelling synthetic domain theory in a realizability topos by taking the well-complete objects [14] as the collection of predomains still provide models as these embed into categories of classes as described in [27].*

The rest of this section is devoted to showing that the category $\mathbf{Dom}_\perp$ has a linear category structure, in the sense of [1], i.e. a symmetric monoidal closed category (SMCC) structure with a symmetric monoidal comonad ! and natural transformations $e\colon !(-) \to I$ and $d\colon !(-) \to !(-) \otimes !(-)$ satisfying a number of axioms. Linear categories model a dual linear / intuitionistic type theory, and showing the existence of this structure is a first step on the way to the construction of a model of $\mathtt{PILL}_Y$.

We know from [24] that for any pair of domains $A, B$ the set $A \multimap B$ is a domain. An application of the General Adjoint Functor Theorem proves that for each domain $A$, the functor

$$A \multimap (-)\colon \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$$

has a left adjoint $A \otimes (-)$. This gives rise to a tensor product on $\mathbf{Dom}_\perp$. By construction $A \otimes B \multimap C \cong A \multimap (B \multimap C)$, and since $A \multimap (B \multimap C)$ is a subset of $A \to (B \to C)$, there is an injective map from $A \otimes B \multimap C$ to $A \times B \to C$. The image of this map consists of the maps in two arguments that are bistrict, i.e. preserve the pointed structure in each argument separately. The embedding above determines the universal bistrict map $\eta\colon A \times B \to A \otimes B$, as usual in such cases.

The domain $\Sigma = L1$ is a unit for the tensor product, since

$$A \otimes L1 \multimap B \cong A \multimap (L1 \multimap B) \cong A \multimap (1 \to B) \cong A \multimap B$$

This proves the following proposition.

**Proposition 2.3.** *The category* $\mathbf{Dom}_\perp$ *has a symmetric monoidal closed structure*

**Lemma 2.4.** *The forgetful functor $U\colon \mathbf{Dom}_\perp \to \mathbf{Dom}$ is a symmetric monoidal functor with respect to the cartesian structure on $\mathbf{Dom}$. The lifting functor $L\colon \mathbf{Dom} \to \mathbf{Dom}_\perp$ is a strong symmetric monoidal functor.*

*Proof.* For the first statement, the natural transformations needed are $\eta$ and the map $1 \to \Sigma$ is the unit of the adjunction.

For the second statement, notice that for all domains $A, B, C$

$$\begin{gathered} L(A \times B) \multimap C \cong A \times B \to C \cong A \to B \to C \cong \\ LA \multimap LB \multimap C \cong LA \otimes LB \multimap C \end{gathered} \tag{1}$$

so that $LA \otimes LB \cong L(A \times B)$ and by definition $L1 \cong \Sigma$. This defines the natural transformation $m$ and map $m_\Sigma$ needed for $L$ to be a strong symmetric monoidal functor. □

**Corollary 2.5.** *The adjunction*

$$\begin{tikzcd} \mathbf{Dom}_\perp \arrow[r, bend right, "U"'] \arrow[r, phantom, "\perp"] & \mathbf{Dom} \arrow[l, bend right, "L"'] \end{tikzcd}$$

*is symmetric monoidal.*

*Proof.* This is a consequence of a general theorem due to Kelly [13] stating that an adjunction of symmetric monoidal functors is a symmetric monoidal adjunction if and only if the left adjoint is strong. See also [19, Theorem 1.4]. □

**Corollary 2.6.** *The functor $LU\colon \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$ extends the SMCC structure on the category of domains to a linear category structure.*

*Proof.* This follows from Corollary 2.5 by a theorem from [2], see also [19, Proposition 1.14]. □

## 3 The domains fibration

In this section we construct a $\mathtt{PILL}_Y$ model based on the linear category structure on $\mathbf{Dom}_\perp$. A $\mathtt{PILL}_Y$ model is a fibred linear category $\mathbb{E} \to \mathbb{B}$ (a fibred symmetric monoidal closed category with a fibred symmetric monoidal comonad with extra structure making each fibre a linear category) with base category $\mathbb{B}$ cartesian, with a generic object, simple products (for modelling polymorphism) and a polymorphic fixed point combinator. However, we shall very often talk about a fibred symmetric monoidal adjunction

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

being a $\mathtt{PILL}_Y$ model, meaning that $\mathbb{E} \to \mathbb{B}$ with the fibred comonad $FG$ is a $\mathtt{PILL}_Y$ model in the first sense and $\mathbb{D}$ is the closure of the coKleisli category for $FG$ under fibred products. The reason for this is that the category $\mathbb{D}$ plays an important role in LAPL-structures (see [16] for a discussion of adjunctions versus monads as models of dual intuitionistic / linear lambda calculi). The reader is referred to [6] or [19] for details on $\mathtt{PILL}_Y$ models, and to [11] for background on the theory of fibrations. The model described in this section will be modified to a parametric $\mathtt{PILL}_Y$-model in Section 4.

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 3.1.** *The fibration*

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$$

*has a fibred linear category 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 category structure hold, since they hold pointwise. □

The next lemma shows how the construction used in [24] to model polymorphism gives right Kan extensions along projections. This structure is what is needed to model polymorphism in the fibration

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}.$$

The proof essentially also appears in [24].

**Lemma 3.2.** *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 computed as sets.

The universal natural transformation $\tau\colon \mathrm{RK}_\pi(g) \circ \pi \Rightarrow g$ is defined as follows. Given any domain $B$, there exists $i\colon D \multimap B$ isomorphism, and we define

$$\tau_{\vec{A},B}\colon \mathrm{RK}_\pi(g)(\vec{A}) \multimap g(\vec{A}, B)$$

as the composition

$$\begin{tikzcd}
\mathrm{RK}_\pi(g)(\vec{A}) \arrow[r, "\pi_D", -o] & g(\vec{A}, D) \arrow[r, "g(\vec{A},i)", -o] & g(\vec{A}, B)
\end{tikzcd}$$

where $\pi_D$ is the projection onto the $D$'th coordinate. To show that this definition is independent of the choice of $D, i$, suppose $D', i'$ is another such choice. Then we have a commutative diagram

$$\begin{tikzcd}
\mathrm{RK}_\pi(g)(\vec{A}) \arrow[r, "\pi_D", -o] \arrow[dr, "\pi_{D'}"', -o] & g(\vec{A}, D) \arrow[r, "g(\vec{A},i)", -o] \arrow[d, "g(\vec{A},(i')^{-1}\circ i)"', -o] & g(\vec{A}, B) \\
& g(\vec{A}, D') \arrow[ur, "g(\vec{A},i')"', -o] &
\end{tikzcd}$$

where the first triangle commutes by definition of $\mathrm{RK}_\pi(g)$ and the second triangle commutes because $g$ is a functor.

One may easily check that the correspondence taking $t\colon f \Rightarrow \mathrm{RK}_\pi(g)$ to $\tau \circ (f\pi)\colon f\pi \Rightarrow g$ is natural and bijective. □

**Lemma 3.3.** *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

$$\frac{\pi^* f \to g}{\overline{f \to \prod g}}$$

□

**Remark 3.4.** *From the proof of 3.2 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 $x$.*

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 3.5.** *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 [19]. □

**Proposition 3.6.**

$$\begin{tikzcd}
\mathbf{DFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right, "\perp"] \arrow[dr] & & \mathbf{DFam}(\mathbf{Dom}) \arrow[ll, bend right] \arrow[dl] \\
& \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\} &
\end{tikzcd}$$

*is a* $\mathtt{PILL}_Y$*-model.*

*Proof.* All that remains to be shown is that the fixed point combinator $Y$ can be modelled. For this 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

$$\begin{tikzcd}
D \arrow[r, "i^{-1}\circ f\circ i"] \arrow[d, "i"] & D \arrow[d, "i"] \\
D' \arrow[r, "f"] & D'
\end{tikzcd}$$

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 3.4, 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 $\mathit{fix}_A$. 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 $\mathit{fix}_A$ 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 $\mathit{fix}_A(f)$. □

## 4 The parametric fibration

In this section, we apply a parametric completion process as in [23, 4] 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 $B$. Following [24] an admissible relation between domains $A, B$ is a subdomain of $A \times B$ and we write $\mathbf{AdmRel}(A, B)$ for the set of admissible relations from $A$ to $B$. We shall often write $R(x, y)$ for $(x, y) \in R$.

**Lemma 4.1** ([24]). *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:

$$\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\text{iso}} \arrow[r, shift left=2] \arrow[r, shift right=2] & (\mathbf{Dom}_\perp)_{\mathtt{iso}} \arrow[l]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[r, shift left=2] \arrow[r, shift right=2] & \mathbf{Dom}_\perp \arrow[l]
\end{tikzcd}$$

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 4.2.** *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

$$R \times S\colon \mathbf{AdmRel}(A \times C, B \times D)$$

as

$$\{((x,y),(w,z))\colon (A \times C) \times (B \times D) \mid R(x,w) \wedge S(y,z)\}$$

and $R \multimap S\colon \mathbf{AdmRel}(A \multimap C, B \multimap D)$ is defined as in [24] as

$$\{(f,g)\colon (A \multimap C) \times (B \multimap D) \mid \forall x\colon A, y\colon B.\, R(x,y) \supset S(f(x),g(y))\}.$$

The relation $R \times S$ is easily seen to be admissible from Lemma 4.1. For each $x, y$

$$\{(f,g)\colon (A \multimap C) \times (B \multimap D) \mid R(x,y) \supset S(f(x),g(y))\} = \\ \bigcap_{(x',y') \in R \cap \{(x,y)\}} \{(f,g)\colon (A \multimap C) \times (B \multimap D) \mid S(f(x'),g(y'))\}$$

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 4.1.

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. instead 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

$$\begin{tikzcd}
& \bar{R} \otimes \bar{S} \arrow[dl] \arrow[dr] \arrow[rrr, "r"] & & & T \arrow[dl] \arrow[dr] & \\
A \otimes C \arrow[rrr, bend right, "t"'] & & B \otimes D \arrow[rrr, bend right, "s"'] & E & & F
\end{tikzcd}$$

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 $\eta$, the map $r$ exists iff there exists a map $u$ making

$$\begin{tikzcd}
& \bar{R} \times \bar{S} \arrow[dl] \arrow[dr] \arrow[rrr, "u"] & & & T \arrow[dl] \arrow[dr] & \\
A \times C \arrow[rrr, bend right, "\hat{s}"'] & & B \times D \arrow[rrr, bend right, "\hat{t}"'] & E & & F
\end{tikzcd}$$

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 $\Sigma \in \mathbf{Dom}_\perp$. Maps $R \otimes eq_\Sigma \multimap S$ correspond to bistrict maps $R \times eq_\Sigma \multimap S$, which correspond to strict maps $R \multimap S$ so that $R \cong R \otimes \Sigma$.

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 4.3.** *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 $\epsilon$:

$$\begin{tikzcd} & !R \arrow[dl] \arrow[dr] \arrow[rrr, "\epsilon"] & & & R \arrow[dl] \arrow[dr] & \\ !A \arrow[rrr, bend right, "\epsilon"'] & & !B \arrow[rrr, bend right, "\epsilon"'] & A & & B. \end{tikzcd}$$

The same reasoning applies for the rest of the linear category 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

$$\begin{tikzcd}
!R \arrow[rr, "!\Delta"] \arrow[d] \arrow[dr] & & !(R \times R) \arrow[rr, "\cong"] \arrow[d] \arrow[dr] & & !R\otimes!R \arrow[d] \arrow[dr] & \\
!A \arrow[rr, bend right, "!\Delta"'] & !B \arrow[rr, bend right, "!\Delta"'] & !(A \times A) \arrow[rr, bend right, "\cong"'] & !(B \times B) \arrow[rr, bend right, "\cong"'] & !A\otimes!A & !B\otimes!B.
\end{tikzcd}$$

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 $\delta, e, m, m_\Sigma$ preserve relations is done likewise. The commutative diagrams of [19, Definition 1.10, Lemma 1.11] commute since they commute in $\mathbf{Dom}_\perp$. $\square$

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 *dom*, *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 4.4.** *The fibration* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ *has a fibred linear category structure and fibred products.*

*Proof.* The structure is defined pointwise, using Lemma 4.3, 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 3.1 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 $\epsilon$ 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. $\square$

The interpretation of polymorphic types as defined in [24] gives simple products in our parametric $\mathtt{PILL}_Y$ model:

**Lemma 4.5.** *The fibration* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ *has a generic object and simple products.*

*Proof.* The generic object is the inclusion

$$\begin{tikzcd} \mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\text{iso}} \arrow[r] \arrow[d, shift right=2] \arrow[d, shift left=2] & \mathbf{AdmRel}(\mathbf{Dom}_\perp) \arrow[d, shift right=2] \arrow[d, shift left=2] \\ (\mathbf{Dom}_\perp)_{\mathtt{iso}} \arrow[r] \arrow[u] & \mathbf{Dom}_\perp \arrow[u] \end{tikzcd}$$

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 4.1.

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 $\mathbf{PFam}(\mathbf{Dom}_\bot)$. 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 3.3. 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 3.2 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

$$\hat{u}_{\vec{A},A}(x) = f^d(id_{\vec{A}}, i) \circ \pi_D \circ u_{\vec{A}}(x) \qquad \hat{u}_{\vec{B},B}(y) = f^d(id_{\vec{B}}, i') \circ \pi_{D'} \circ u_{\vec{B}}(y). \quad (2)$$

Since $(i, i')^*R' \in \mathbf{AdmRel}(D, D')$, and since $u$ preserves relations, we must have

$$(\pi_D \circ u_{\vec{A}}(x), \pi_{D'} \circ u_{\vec{B}}(y)) \in f^r(\vec{R}, (i, i')^*R') \quad (3)$$

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 4.6.** *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 [19, Proposition 1.21]. □

**Proposition 4.7.**

$$\begin{tikzcd} \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{PFam}(\mathbf{Dom}) \arrow[ll, bend right] \arrow[dl] \\ & \mathbf{PDom} & \end{tikzcd} \tag{4}$$

*is a* $\mathtt{PILL}_Y$*-model.*

*Proof.* We just need to show how to model the $Y$-combinator in the fibration

$$\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{((\mathbf{Dom}_\perp)_{\mathtt{iso}})^n \mid n\}.$$

This is 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

$$\forall D, D' \in \mathbf{D}. \forall R\colon \mathbf{AdmRel}(D, D'). \forall f\colon D \to D, g\colon D' \to D'.$$
$$(R \to R)(f, g) \supset R(\mathit{fix}_D f, \mathit{fix}_{D'} g).$$

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

$$\begin{tikzcd} R \arrow[r, "(f\times g)|_R"] \arrow[d, "\circ" marking] & R \arrow[d, "\circ" marking] \\ D \times D' \arrow[r, "f\times g"] & D \times D'. \end{tikzcd}$$

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

$$\begin{tikzcd} D \times D' \arrow[r, "f\times g"] \arrow[d, "\circ" marking] & D \times D' \arrow[d, "\circ" marking] \\ D \arrow[r, "f"] & D \end{tikzcd}$$

(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 Proposition 3.6. □

## 5 The LAPL-structure

In this section we show that the $\mathtt{PILL}_Y$-model (4) is parametric by constructing a parametric LAPL-structure around it. An LAPL-structure [6] is a model of Linear Abadi & Plotkin Logic [5], which is a logic for reasoning about parametricity and recursion. The construction proceeds in two steps: first a pre-LAPL structure, given by a diagram of categories and functors:

$$\begin{tikzcd} & & \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\ \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[r, shift right] \arrow[rrd] & \mathbf{PFam}(\mathbf{Dom}) \arrow[l, shift right] \arrow[r] \arrow[rd] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\ & & \mathbf{PDom}. \end{tikzcd} \qquad (5)$$

is constructed. The left hand side is the model of polymorphism and recursion that the logic reasons about. This is just (4). The category $\mathbf{DFam}(\mathbf{Set})$ is used to interpret contexts of the logic. Contexts can contain relational variables of the form $R\colon \mathsf{Rel}(\sigma, \tau)$ and collections of relations do not naturally live in the model (4). Finally $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{DFam}(\mathbf{Set})$ is a logic fibration, which models the propositions of the logic.

The second half of the construction is a reflexive graph

$$\begin{tikzcd} \left(\begin{array}{c}\mathbf{PFam}(\mathbf{Dom}_\perp)\\ \downarrow \\ \mathbf{PDom}\end{array}\right) \arrow[r, shift left=2, leftarrow] \arrow[r, "\Phi"] \arrow[r, shift right=2, leftarrow] & \left(\begin{array}{c}\mathbf{LinAdmRel}\\ \downarrow \\ \mathbf{AdmRelCtx}\end{array}\right) \end{tikzcd}$$

of maps of $\mathtt{PILL}_Y$ models where the right hand side is a fibration, whose objects of the total category essentially are indexed relations, constructed from (5). This reflexive graph gives a relational interpretation of types of the model, essentially as was the idea of [15].

The intuition of the reasoning about the model is the following. Even though types in the model (4) are pairs $(f^r, f^d)$, when reasoning about parametricity, we will just consider the $f^d$ part of a type. Propositions on types of the model are modelled as subsets, so $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{DFam}(\mathbf{Set})$ is essentially an indexed version of the subobject fibration on $\mathbf{Set}$. 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 $f^r(\vec{R})\colon \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B}))$. This is reflected in the model by $\Phi$ essentially mapping a type $(f^p, f^r)$ to $f^r$.

We now give the formal definition of the categories of (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 5.1.** *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 5.2.** *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 5.3.** *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

$$(\overline{\pi})^*\colon \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))_f \to \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))_{\pi^* f}.$$

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

$$(t^*(\prod h))_{\vec{A}} = (t^*_{\vec{A}}(\bigcap_{D\in\mathbf{D}} h(\vec{A}, D)))_{\vec{A}} = (\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in \bigcap_{D\in\mathbf{D}} h(\vec{A}, D)\})_{\vec{A}}$$

and on the other hand

$$(\prod((\pi^* t)^*(h)))_{\vec{A}} = (\prod(\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in h(\vec{A}, B)\})_{\vec{A},B})_{\vec{A}} =$$
$$(\bigcap_{D\in\mathbf{D}}\{x \in f(\vec{A}) \mid t_{\vec{A}}(x) \in h(\vec{A}, D)\})_{\vec{A}}$$

Since these two are clearly equal, the Beck-Chevalley condition is satisfied. □

**Lemma 5.4.** *The diagram (5) is a pre-LAPL-structure.*

*Proof.* All that is missing in this proof is the definition of the fibred functor $U$

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \times_{\mathbf{PDom}} \mathbf{PFam}(\mathbf{Dom}_\perp)^{\mathrm{fop}} \arrow[r] \arrow[dr] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
& \mathbf{PDom}
\end{tikzcd}$$

(where $(-)^{\mathrm{fop}}$ is the operation that takes the opposite category in each fibre, see [6]). 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 5.5.** *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).*

*Proof.* 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 4.1. That admissible relations are closed under conjunction and universal quantification is a consequence of the same lemma.

If $\phi$ is a proposition and $\rho$ is an admissible relation, then

$$\{(x, y) \mid \phi \supset \rho(x, y)\} = \bigcap_{z \in \{0 | \phi\}} \{(x, y) \mid \rho(x, y)\}$$

which is an admissible relation by Lemma 4.1. So $(x, y).\, \phi \supset \rho(x, y)$ is an admissible relation. $\square$

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 category structure on types defined in the model here agrees with the linear category structure on $\mathbf{LinAdmRel} \to \mathbf{AdmRelCtx}$ defined abstractly in the LAPL-logic.

**Theorem 5.6.** *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

$$(\mathbf{Dom}_\perp)^{n+m}_{\mathtt{iso}} \to \mathbf{Set}.$$

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

$$(k_{\vec{A},\vec{B}} \colon f^d(\vec{A}, \vec{B}) \to \mathbf{AdmRel}(g^d(\vec{A}), h^d(\vec{B})))_{(\vec{A},\vec{B}) \in (\mathbf{Dom}_\perp)^{n+m}_{\mathtt{iso}}}.$$

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

$$\begin{array}{r} t\colon (g^r, g^d) \to ((g')^r, (g')^d) \text{ in } \mathbf{PFam}(\mathbf{Dom}_\perp)_n \\ s\colon (h^r, h^d) \to ((h')^r, (h')^d) \text{ in } \mathbf{PFam}(\mathbf{Dom}_\perp)_m \end{array}$$

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 $\mathtt{PILL}_Y$-models:

$$\left(\begin{tikzcd} \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[d] \\ \mathbf{PDom} \end{tikzcd}\right) \begin{array}{c} \longleftarrow \\ \longleftarrow \end{array} \left(\begin{tikzcd} \mathbf{LinAdmRel} \arrow[d] \\ \mathbf{AdmRelCtx} \end{tikzcd}\right)$$

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 $\Phi$ going the other way by first defining the map

$$\mathbf{PDom} \to \mathbf{AdmRelCtx}$$

to map an object $n$ to $(n, n, \prod_{i\leq n} V(\pi_i \circ \pi, \pi_i \circ \pi'))$ where $\pi, \pi'$ 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

$$(\textstyle\prod_{i\leq n} \mathbf{AdmRel}(A_i, B_i))_{\vec{A}\in\mathbf{Dom}^n, \vec{B}\in\mathbf{Dom}^n}$$

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 $\mathtt{PILL}_Y$-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

$$(\vec{R} \in \textstyle\prod_{i\leq n} \mathbf{AdmRel}(A_i, B_i) \mapsto f^r(\vec{R}) \in \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B})))_{\vec{A},\vec{B}}.$$

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 $\Phi$ commutes with reindexing and therefore defines a map of fibrations. It is also evident that $\Phi$ 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 $\Phi$ preserves generic object. It is also clear that it preserves products in the base.

Let us show that $\Phi$ preserves !. Recall that applying ! in $\mathbf{PFam}(\mathbf{Dom}_\perp)$ maps a relation to the relation obtained by lifting both maps in the span. On the other hand, we know from [5] that given $\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})$ the relation $!\Phi(f^r, f^d)(\vec{R})$ is the smallest admissible relation containing

$$\{(\eta_{f^p(\vec{A})}(x), \eta_{f^p(\vec{B})}(y)) \mid (x, y) \in f^r(\vec{R})\}$$

where $\eta\colon id \to !(-)$ is the unit of the monad $L$ on $\mathbf{Dom}$. It is an easy diagram chase to see that $\Phi(!(f^r, f^d))(\vec{R})$ satisfies this characterising property, and so $\Phi(!(f^r, f^d))(\vec{R}) = !\Phi(f^r, f^d)(\vec{R})$.

To see that the simple products are preserved, an easy calculation shows that both combinations of simple products and $\Phi$ 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 $\Phi$ preserves $\multimap$.

Finally, we show that $\Phi$ 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 4.4, 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 $\Phi$ preserves tensor. □

**Theorem 5.7.** *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

$$(f^r\colon \textstyle\prod_{i \le n} \mathbf{AdmRel}(A_i, B_i) \to \mathbf{AdmRel}(f^d(\vec{A}), f^d(\vec{B})))_{\vec{A}, \vec{B}}.$$

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. $\square$

## 6 Proving consequences of parametricity for `Lily`

The parametric LAPL-structure constructed in this paper is based on the interpretation of $\texttt{Lily}_{\texttt{strict}}$ of [24] with the type constructors defined the same way. Thus the results about LAPL-structures should be available for us for reasoning about the interpretation of $\texttt{Lily}_{\texttt{strict}}$, and since the computational adequacy result of [24] gives a strong correspondence between syntax and semantics, we should be able to lift these results to the syntax. Precisely, in this section we would like to prove the correctness of encodings of recursive types in $\texttt{Lily}_{\texttt{strict}}$ up to ground contextual equivalence.

Rather than working with $\texttt{Lily}_{\texttt{strict}}$ we shall work with a subset of $\texttt{PILL}_Y$ equipped with an operational semantics, which we shall call `Lily` since it essentially is the language of [3] (in [3] recursion is introduced via recursive thunks rather than the recursion operator used here, but these constructions are interdefinable). We do this because it greatly simplifies the presentation.

The types of `Lily` are given by the grammar:

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

The terms are given by the grammar

$$t ::= x \mid \lambda x \colon \sigma.\, t \mid s(t) \mid !t \mid \texttt{let}\, !x\, \texttt{be}\, s\, \texttt{in}\, t \mid \Lambda \alpha.\, t \mid t(\sigma) \mid \texttt{rec}\, x \colon \sigma.\, t$$

and the typing rules are presented in Figure 1. In the figure, the metanotation $\Xi$ is used for the context of type variables, $\Gamma$ is an intuitionistic variable context and $\Delta$ is a linear variable context. In all the rules of the figure, it is assumed that all

$$\frac{\Xi \mid \Gamma, x\colon \sigma; - \vdash t\colon \sigma}{\Xi \mid \Gamma; - \vdash \mathtt{rec}\, x\colon \sigma.\, t\colon \sigma}$$

$$\frac{}{\Xi \mid \Gamma, x\colon \sigma; - \vdash x\colon \sigma} \qquad \frac{}{\Xi \mid \Gamma; x\colon \sigma \vdash x\colon \sigma}$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t\colon \sigma \multimap \tau \quad \Xi \mid \Gamma; \Delta' \vdash u\colon \sigma}{\Xi \mid \Gamma; \Delta, \Delta' \vdash t\, u\colon \tau}\ \Delta, \Delta' \text{ disjoint}$$

$$\frac{\Xi \mid \Gamma; \Delta, x\colon \sigma \vdash u\colon \tau}{\Xi \mid \Gamma; \Delta \vdash \lambda^{\circ} x\colon \sigma.\, u\colon \sigma \multimap \tau}$$

$$\frac{\Xi \mid \Gamma; - \vdash t\colon \sigma}{\Xi \mid \Gamma; - \vdash !t\colon \sigma}$$

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

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

$$\frac{\Xi \mid \Gamma; \Delta \vdash s\colon !\sigma \qquad \Xi \mid \Gamma, x\colon \sigma; \Delta' \vdash t\colon \tau}{\Xi \mid \Gamma; \Delta, \Delta' \vdash \mathtt{let}\, !x\colon !\sigma\, \mathsf{be}\, s\, \mathsf{in}\, t\colon \tau}\ \Delta, \Delta' \text{ disjoint}$$

Figure 1: Typing rules for `Lily`

typing judgements $\Xi \mid \Gamma; \Delta \vdash t \colon \sigma$ are well formed, i.e., all the free type variables occurring in $\Gamma, \Delta, t$ and $\sigma$ are in $\Xi$.

The language `Lily` presented here is essentially the subset of $\mathtt{PILL}_Y$ excluding the $\otimes$ and $I$ type constructors. The only difference is the formulation of the recursion operator, $\mathtt{PILL}_Y$ having a polymorphic recursion operator

$$Y \colon \prod \alpha.\, (\alpha \to \alpha) \to \alpha,$$

where $\sigma \to \tau$ is shorthand for $!\sigma \multimap \tau$. The two formulations are interdefinable, as `rec` can be defined from $Y$ by

$$\mathtt{rec}\, x \colon \sigma.\, t = Y\, \sigma\, !(\lambda x \colon \sigma.\, t)$$

where $\lambda x \colon \sigma.\, t$ is shorthand for $\lambda^\circ y \colon !\sigma.\, \mathtt{let}\, !x\, \mathtt{be}\, y\, \mathtt{in}\, t$, and $Y$ can be defined from `rec` as

$$Y = \Lambda\alpha.\, \lambda f \colon \alpha \to \alpha.\, \mathtt{rec}\, x \colon \alpha.\, f(x).$$

Formally the former encoding can be seen as defining an interpretation of `Lily` into $\mathtt{PILL}_Y$.

The operational semantics is given by an evaluation relation relating programs of `Lily`, i.e., closed terms of closed type, to values. The operational semantics is the call-by-value operational semantics for $\mathtt{Lily}_{\mathtt{strict}}$ as defined in [24], which we recall in Figure 2.

$$\frac{}{\lambda^\circ x \colon \sigma.\, t \Downarrow \lambda^\circ x \colon \sigma.\, t} \qquad \frac{s \Downarrow \lambda^\circ x \colon \sigma.\, t' \quad t \Downarrow v' \quad t'[v'/x] \Downarrow v}{s(t) \Downarrow v}$$

$$\frac{}{!t \Downarrow !t} \qquad \frac{s \Downarrow !s' \quad t[s'/x] \Downarrow v}{\mathtt{let}\, !x\, \mathtt{be}\, s\, \mathtt{in}\, t \Downarrow v} \qquad \frac{}{\Lambda\alpha.\, t \Downarrow \Lambda\alpha.\, t}$$

$$\frac{t \Downarrow \Lambda\alpha.\, t' \quad t'[\sigma/\alpha] \Downarrow v}{t(\sigma) \Downarrow v} \qquad \frac{t[\mathtt{rec}\, x \colon \sigma.\, t/x] \Downarrow v}{\mathtt{rec}\, x \colon \sigma.\, t \Downarrow v}$$

Figure 2: The call-by-value operational semantics of `Lily`

The next proposition relates `Lily` to the language $\mathtt{Lily}_{\mathtt{strict}}$ of [24]. In particular, this will allow us to transfer the computational adequacy result from [24] to our setting (Theorem 6.2 below).

**Proposition 6.1.** *For every well typed term $\Xi \mid \Gamma; \Delta \vdash t \colon \sigma$ of* `Lily` *there exists a labelling $\delta$ such that $\Gamma, \Delta \mid \delta \vdash_\Xi t \colon \sigma$ is a well typed term of* $\mathtt{Lily}_{\mathtt{strict}}$. *In* *particular, every* `Lily` *program is a program in* $\texttt{Lily}_{\texttt{strict}}$ *of the same type. The interpretation of* $\texttt{Lily}_{\texttt{strict}}$ *given in [24] coincides on types and programs of* `Lily` *with that of* $\texttt{PILL}_Y$ *into the* $\texttt{PILL}_Y$ *model (4) up to the interpretation of* `Lily` *into* $\texttt{PILL}_Y$ *as defined above.*

*Proof Sketch.* We just sketch the proof of the second half of the theorem as the first half is straightforward.

Concerning the interpretation of types, the verification is by structural induction on types, and most cases are straightforward, as the $\texttt{PILL}_Y$ model in this paper was constructed using the constructions of [24]. We just show that the two relational interpretations of the ! type constructor agrees. We write $\llparenthesis - \rrparenthesis^d$, $\llparenthesis - \rrparenthesis^r$ for the interpretation defined in [24] and $[\![-]\!]^d$, $[\![-]\!]^r$ for the interpretation of $\texttt{PILL}_Y$ types into the LAPL-structure. For $\vec{R}\colon \mathbf{AdmRel}(\vec{A},\vec{B})$.

$$\begin{array}{rcl}
\llparenthesis \vdash_\Xi !\sigma \rrparenthesis^r(\vec{R}) & = & \{(e,f)\colon L\llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{A}) \times L\llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{A}) \mid \\
& & \forall x\colon \llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{A}).\, x \in e \supset \exists y \in f \supset \llparenthesis \vdash_\Xi \sigma\rrparenthesis^r(\vec{R})(x,y) \wedge \\
& & \forall y\colon \llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{B}).\, y \in f \supset \exists x \in e \supset \llparenthesis \vdash_\Xi \sigma\rrparenthesis^r(\vec{R})(x,y)\} \\
& = & \{(e,f)\colon L\llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{A}) \times L\llparenthesis \vdash_\Xi \sigma\rrparenthesis^d(\vec{A}) \mid \\
& & \exists x \in e \supset\subset \exists y \in f \wedge (\forall x \in e, y \in f.\, (x,y) \in \llparenthesis \vdash_\Xi \sigma\rrparenthesis^r(\vec{R}))\}.
\end{array}$$

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 $\exists x \in e \supset\subset \exists y \in f$ and $x \in e, y \in f \supset (x,y) \in [\![\Xi \vdash \sigma]\!]^r(\vec{R})$.

For proving that the two interpretations agree on terms, again this is done by structural induction, and we must construct an induction hypothesis that can be used on general open terms. An open term

$$\Xi \mid x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n; x'_1\colon \sigma'_1, \ldots, x'_m\colon \sigma'_m \vdash t\colon \tau$$

is interpreted in the $\texttt{PILL}_Y$ model as an indexed family of strict functions

$$([\![t]\!]_{\vec{A}}\colon \bigotimes_i L[\![\sigma_i]\!]^d(\vec{A}) \otimes \bigotimes_j [\![\sigma'_j]\!]^d(\vec{A}) \multimap [\![\tau]\!]^d(\vec{A}))_{\vec{A}}.$$

On the other hand the interpretation of $t$ as defined in [24] is as a family of maps

$$(\llparenthesis t \rrparenthesis_{\vec{A}}\colon \textstyle\prod_i [\![\sigma_i]\!]^d(\vec{A}) \times \prod_j [\![\sigma'_j]\!]^d(\vec{A}) \to [\![\tau]\!]^d(\vec{A}))_{\vec{A}}$$

where $A$ ranges over vectors of domains. Since maps with the type of $[\![t]\!]_{\vec{A}}$ correspond bijectively to maps with the type of $(\![t]\!)_{\vec{A}}$ which are strict in each of the last $m$ variables, the induction hypothesis states that for all $\vec{A}$ the map $[\![t]\!]_{\vec{A}}$ corresponds to $(\![t]\!)_{\vec{A}}$. Further details of the induction proof can be found in [17]. $\square$

Contextual equivalence is defined with respect to observing termination at !-types. More precisely, consider two (possibly open) terms $t, t'$ of the same type in the same context. We say that $t$ and $t'$ are contextually equivalent if for all contexts $C[-]$ such that both $C[t]$ and $C[t']$ are programs of some type of the form $!\sigma$, $\exists v.\, C[t] \Downarrow v$ if and only if $\exists v.\, C[t'] \Downarrow v$. Since `Lily` is a sublanguage of `Lily`$_{\texttt{strict}}$ every `Lily` context is a `Lily`$_{\texttt{strict}}$ context, and so if two `Lily` terms are contextually equivalent considered as `Lily`$_{\texttt{strict}}$ terms, they are also equivalent as `Lily` terms. This allows us to import the following results from [24].

**Theorem 6.2** ([24])**.** *The interpretation is computationally adequate in the sense that if $[\![t]\!] = [\![t']\!]$ for terms $t, t'$ of the same type then $t$ and $t'$ are contextually equivalent.*

**Remark 6.3.** *Theorem 6.2 is proved in [24] by reasoning in the intuitionistic set theory of the model construction. As argued in* loc. cit.*, this means that given a model of SDT, Theorem 6.2 holds in the "real world" (not just as seen from the given model of SDT) iff the given model of SDT is 1-consistent in the sense of [25, 27]: any sentence of the form $\exists n\colon \mathbb{N}.\, \phi(n)$, for $\phi$ a primitive recursive predicate,—a $\Sigma^0_1$-sentence—is true in the model iff there exists (in the external sense) a natural number $n$ such that $\phi(n)$ is true.*

*Since such models of SDT do exist (the example of a realizability topos satisfying the strong completeness axiom [14] where one takes predomains to be the well-complete objects mentioned earlier is an example), in this section we will assume that we are given such a model. We emphasise that since the results that we aim to prove (Theorems 6.5, 6.6 and 6.7) are purely syntactic, they are completely independent of the choice of SDT model, and so just the existence of a model satisfying the requirement above implies that they hold in the "real world".*

**Remark 6.4.** *We have given a call-by-value operational semantics for* `Lily`*, but it is well known ([3, 24]) that the call-by-name evaluation relation $\Downarrow^n$ obtained from $\Downarrow$ by replacing the rule for function evaluation in Figure 2 by*

$$\frac{s \Downarrow^n \lambda^\circ x\colon \sigma.\, t' \qquad t'[t/x] \Downarrow^n v}{s(t) \Downarrow^n v}$$

*gives the same notion of contextual equivalence as $\Downarrow$, and so the results we prove here for the call-by-value semantics also hold for call-by-name semantics.*

## 6.1 Consequences of parametricity for `Lily`

Finally, the results from the theory of parametric LAPL-stuctures can be used to prove parametricity results for `Lily` up to contextual equivalence, using the computational adequacy result.

For the formulation of the parametricity results for `Lily`, consider the category $\mathbf{Lily}$ whose objects are the closed types of `Lily` and whose morphisms from $\sigma$ to $\tau$ are closed terms of type $\sigma \multimap \tau$ of `Lily` identified up to ground contextual equivalence.

**Theorem 6.5.** *For all closed types $\sigma$ of* `Lily`*, the types $\sigma$ 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 [5]. We know from the theory of parametric LAPL structures that both compositions of these maps are interpreted in the LAPL-structure as identities and thus by computational adequacy the two maps are each others inverses in `Lily` up to contextual equivalence. □

As always, type expressions $\alpha \vdash \sigma(\alpha)$ in `Lily` for which $\alpha$ only appears positively in $\sigma$ induce functors on $\mathbf{Lily}$.

**Theorem 6.6.** *All functors $\mathbf{Lily} \to \mathbf{Lily}$ induced by types $\sigma(\alpha)$ in* `Lily` *have initial algebras and final coalgebras.*

*Proof.* The initial algebra

$$\texttt{in} : \sigma(\mu\alpha.\, \sigma(\alpha)) \multimap \mu\alpha.\, \sigma(\alpha)$$

and the term $\texttt{fold} : \prod \alpha.\, (\sigma(\alpha) \multimap \alpha) \to \mu\alpha.\, \sigma(\alpha) \multimap \alpha$ are defined as in $\texttt{PILL}_Y$ [5]. To see that this is a week initial algebra, suppose $f \colon \sigma(\tau) \multimap \tau$ is a program of `Lily`. From the theory of parametric LAPL-structures we know that $f \circ \sigma(\texttt{fold}\ \tau\, !f)$ and $(\texttt{fold}\ \tau\, !f) \circ \texttt{in}$ are interpreted equally in the model, and are thus contextually equivalent. Suppose now `Lily` programs $g, h$ are maps of algebras from `in` to some other algebra. Since we know that `in` is interpreted as an initial algebra in the model, $g, h$ are interpreted equally in the model and are thus contextually equivalent.

The existence of final coalgebras is proved the same way, except that one has to be more careful about the encoding of the structure, as the encoding used in [5] uses the $\otimes$ type constructor which is not in `Lily`. However, the final coalgebras can be defined in `Lily` using Plotkin's encoding $\sigma \otimes \tau \cong \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$, which is correct as a consequence of parametricity. □

We can even solve recursive type equations with parameters in `Lily`. For a precise statement of this, define for each context of free type variables $\Xi$, the category $(\mathbf{Lily})_\Xi$ whose objects are types $\vdash_\Xi \sigma$ and whose morphisms are closed terms (here meaning: no free varaibles, only free type variables contained in $\Xi$) considered equal up to ground contextual equivalence, as in the definition of $\mathbf{Lily}$.

**Theorem 6.7.** *Suppose* $\vdash_{\vec{\alpha},\vec{\beta},\alpha,\beta} \sigma(\vec{\alpha},\vec{\beta},\alpha,\beta)$ *is a type in* `Lily`*, in which the type variables* $\vec{\alpha},\alpha$ *occur only negatively and the type variables* $\vec{\beta},\beta$ *only positively. Then there exists a type* $\mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta})$ *such that the types*

$$\mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta}) \text{ and } \sigma(\vec{\alpha},\vec{\beta},\mathit{Fix}(\sigma)(\vec{\beta},\vec{\alpha}),\mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta}))$$

*are isomorphic as objects of* $(\mathbf{Lily})_{\vec{\alpha},\vec{\beta}}$.

*Moreover, the type* $\mathit{Fix}(\sigma)$ *together with the isomorphism above satisfies the following indexed version of Freyd's initial dialgebra property. For any pair of types* $\vdash_{\vec{\alpha},\vec{\beta}} \omega,\omega'$ *and closed terms*

$$g\colon \sigma(\vec{\alpha},\vec{\beta},\omega',\omega) \multimap \omega$$
$$g'\colon \omega' \multimap \sigma(\vec{\beta},\vec{\alpha},\omega,\omega')$$

*there exists unique* $h$, $h'$ *making*

$$\begin{tikzcd}
\sigma(\vec{\alpha},\vec{\beta},\mathit{Fix}(\sigma)(\vec{\beta},\vec{\alpha}),\mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta})) \arrow[r, multimap, "\cong"] \arrow[d, multimap, "\sigma(\vec{\alpha},\vec{\beta},h',h)"'] & \mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta}) \arrow[d, multimap, "h"] \\
\sigma(\vec{\alpha},\vec{\beta},\omega',\omega) \arrow[r, multimap, "g"] & \omega
\end{tikzcd}$$

$$\begin{tikzcd}
\omega' \arrow[r, multimap, "g'"] \arrow[d, multimap, "h'"'] & \sigma(\vec{\beta},\vec{\alpha},\omega,\omega') \arrow[d, multimap, "\sigma(\vec{\beta},\vec{\alpha},h,h')"] \\
\mathit{Fix}(\sigma)(\vec{\beta},\vec{\alpha}) \arrow[r, multimap, "\cong"] & \sigma(\vec{\beta},\vec{\alpha},\mathit{Fix}(\sigma)(\vec{\alpha},\vec{\beta}),\mathit{Fix}(\sigma)(\vec{\beta},\vec{\alpha}))
\end{tikzcd}$$

*commute.*

*Proof.* As in the proof of 6.6, the solutions to the recursive domain equations are encoded as in $\mathtt{PILL}_Y$ using encodings of $\otimes$ and the fixed point combinator $Y$ in `Lily` as defined above. The general theory of LAPL-structures then proves that the relevant equalities hold in the model, which implies that they hold in `Lily` up to contextual equivalence. $\square$

**Remark 6.8.** *The above reasoning can also be used to prove similar theorems for* $\mathtt{Lily}_{\mathtt{strict}}$.

## 7 Conclusions

We have constructed an LAPL-structure based on the interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ into models of synthetic domain theory presented in [24]. Comparing this with the concrete domain theoretic LAPL-structure of [7], the completion process for

LAPL-structures of [18], and the LAPL-structure based on the operational semantics of `Lily` [3] 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 [24]. Combining these results with adequacy of the interpretation of $\texttt{Lily}_{\texttt{strict}}$, we have shown consequences of parametricity for `Lily` up to ground contextual equivalence. These consequences include encodings of inductive, coinductive and recursive types.

A more direct route to proving the consequences of parametricity for $\texttt{Lily}_{\texttt{strict}}$ or `Lily` would have been to work out the proofs used in [5] in the intuitionistic set theory of the model. For some readers this may be more appealing. However, apart from showing the generality of the notion of LAPL-structure, the route taken here has the advantage of giving a model of $\texttt{PILL}_Y$ and LAPL rather than just an interpretation, presenting semantic notions of such concepts as open types an propositions on open types.

The connection between parametricity and the question of strictness vs. linearity is not fully understood. The concrete model of [7] as well as the LAPL-structure considered here both model strictness rather than linearity. The LAPL-structure based on the operational semantics of `Lily` [3] under construction will be the first concrete example of a parametric LAPL-structure modelling linearity rather than strictness, but it would be interesting to study parametric LAPL-structures not constructed syntactically, modelling linearity rather than strictness.

## References

[1] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge, August 1992. 2

[2] Nick Benton, G. M. Bierman, J. Martin E. Hyland, and Valeria de Paiva. Linear $\lambda$-calculus and categorical models revisited. In E. Börger, editor, *Proceedings of the Sixth Workshop on Computer Science Logic*, volume 702 of *Lecture Notes in Computer Science*, pages 61–84. Springer-Verlag, 1992. 2

[3] 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, 6, 6.4, 7

[4] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*, 15(4):709–772, 2005. 4

[5] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Linear Abadi & Plotkin logic. *Logical Methods in Computer Science*, 2, 2006. 1, 5, 5, 6.1, 6.1, 7

[6] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Category-theoretic models of linear Abadi & Plotkin logic. *Theory and Applications of Categories*, 20:116–151, 2008. 1, 3, 5, 5

[7] Lars Birkedal, Rasmus Ejlers Møgelberg, and Rasmus Lerchedahl Petersen. Domain-theoretical models of parametric polymorphism. *Theor. Comput. Sci*, 388(1-3):152–172, 2007. 1, 7

[8] 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. 1

[9] 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. 1

[10] 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. 1

[11] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 3

[12] André Joyal and Ieke Moerdijk. *Algebraic Set Theory*. Number 220 in London Mathematical Society Lecture Notes in Mathematics. Cambridge University Press, 1995. 2.2

[13] 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. 2

[14] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Math. Struct. in Comp. Science*, 11, 1996. 2.2, 6.3

[15] 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

[16] Maria Emilia Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. *Applied Categorical Structures*, 13(1):1–36, 2005. 3

[17] R. E. Møgelberg. *Categorical and domain theoretic models of parametric polymorphism*. PhD thesis, IT University of Copenhagen, 2005. 6

[18] 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, 7

[19] R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005. 2, 2, 3, 3, 4, 4

[20] 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

[21] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1

[22] 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

[23] 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, 4

[24] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, available from http://homepages.inf.ed.ac.uk/als/Research/, 2004. 1, 2, 2.1, 2, 2.2, 2, 3, 4, 4.1, 4, 4, 6, 6, 6, 6.1, 6, 6.2, 6.3, 6.4, 7

[25] A. Simpson. Computational adequacy in an elementary topos. In *CSL: 12th Workshop on Computer Science Logic*. LNCS, Springer-Verlag, 1998. 6.3

[26] 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. 2.2

[27] A. Simpson. Computational adequacy for recursive types in models of intuitionistic set theory. *Annals of Pure and Applied Logic*, 130, 2004. 2.2, 6.3