The IT University
of Copenhagen

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

**Rasmus Ejlers Møgelberg**
**Lars Birkedal**
**Giuseppe Rosolini**

**IT University Technical Report Series**

**TR-2005-59**

**ISSN 1600–6100**

**February 2005**

ISSN 1600–6100

ISBN 87-7949-088-3

Copies may be obtained by contacting:

IT University of Copenhagen
Glentevej 67
DK-2400 Copenhagen NV
Denmark

Telephone: +45 38 16 88 88
Telefax: +45 38 16 88 99
Web `www.itu.dk`

# 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 [13] 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.

## 1 Introduction

It was first realized by Plotkin [9, 8] 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 [9] Plotkin also sketched a logic for reasoning about parametricity for PILL$_Y$ (the logic is a variant of Abadi & Plotkin's logic for parametricity [10]) 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 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 [6] a parametric completion process along the lines of [11] is presented constructing parametric LAPL-structures out of a large class of models of PILL$_Y$.

In recent work Simpson and Rosolini [13] have constructed an interpretation (or rather a family of interpretations) of 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 [13] gives an interpretation of 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 [13]. 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 $\text{PILL}_Y$-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 [13] 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 [13] 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 [13], constructing a category of domains. In Sections 4-6 we present the LAPL-structure. We first present a model of $\text{PILL}_Y$ 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 $\text{Lily}_{\text{strict}}$. In particular, we show how to solve recursive domain equations in $\text{Lily}_{\text{strict}}$. First, however, we present the language and sketch the results of [13].

In Appendix A we address the following question: It is well known [9] that in $\text{PILL}_Y$, 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 $\text{PILL}_Y$, then one obtains a language as expressive as the original $\text{PILL}_Y$? In particular, it could be that there were terms in $\text{PILL}_Y$, 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 $\text{Lily}_{\text{strict}}$.

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

## 2 Synthetic Domain Theory

In this section we recall the approach to synthetic domain theory (SDT) presented in [13]. 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 paper, we follow [13] and work informally in an intuitionistic set theory. Formally, the set theory may be taken to be IZF [14]. Below we define a notion of pointed set and assume that we are given a class of special sets called predomains satisfying certain conditions. We then define domains to be pointed pre-domains. Among our axioms will be that all endomaps on domains have fixed points.

We emphasize that there do exist models of SDT as presented here. For example, we can model SDT in any realizability topos satisfying the *strong completeness axiom* of [5], 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 $1 = \{\emptyset\}$:

$$\Omega = P(1).$$

Notice that since we work in intuitionistic set theory, $\Omega$ is not just the set $\{\emptyset, \{\emptyset\}\}$ as it is classically. In fact for every proposition $p$ we can associate the element

$$\{\emptyset \mid p\} \in \Omega$$

and for each element $X \in \Omega$ we can associate the proposition $\emptyset \in X$, and these associations are each others inverses up to provable equivalence of propositions. Motivated by this, we call $\Omega$ *the set of truth values*.

We assume that we are given a set $\Sigma \subseteq \Omega$ of truth values, which we think of as the truth values of propositions of the form “P terminates”, for programs P. We will not formalize what we mean by computation, but instead we will axiomatize the properties needed for $\Sigma$.

**Axiom 2.1.** *The subset $\Sigma \subseteq \Omega$ is a dominance [12], 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 $p \in \Sigma$ we define 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.** 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.** 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 $e \in X^p$

$$f(r_p(e)) = s_p(\{f(x) \mid x \in e\})$$

In this paper we will often leave the pointed structure implicit and simply write $X$ for a pointed set $(X, (r_p)_{p \in \Sigma})$. We write $f \colon X \multimap Y$ to denote that $f$ is a pointed map, and we write $X \multimap Y$ for the set of pointed maps from $X$ to $Y$. We say that $X' \subset X$ is a subpointed set of $X$ if for all $p \in \Sigma$ and $e \in (X')^p$ we have $r_p(e) \in X'$ , 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 $(Y_x, (r_p^{Y_x})_{p \in \Sigma})_{x \in X}$, we can define a pointed structure on $\prod_{x \in X} Y_x$ by defining*

$$r_p^{\prod_{x \in X} Y_x}(e) = (r_p^{Y_x}(\{\pi_x(z) \mid z \in e\}))_{x \in X}$$

*where $\pi_x \colon \prod_{x \in X} Y_x \to Y_x$ denotes the projection. 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

$$r_{p \wedge q}^{\prod_x Y_x}(e) = (r_{p \wedge q}^{Y_x}(\{\pi_x(z) \mid z \in e\}))_{x \in X} =$$
$$(r_p^{Y_x}(\{r_{p \wedge q}^{Y_x}(\{\pi_x(z) \mid z \in e\}) \mid p\}))_{x \in X} = r_p^{\prod_x Y_x}(\{r_{p \wedge q}^{\prod_x Y_x}(e) \mid p\}).$$

This proves that we have defined 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

$$\begin{array}{rcl} \phi(f)(e) & = & r_p^Y(\{f(x) \mid x \in e\}) \\ \psi(f)(e) & = & f(r_p^X(e)) \end{array}$$

for $e \in X^p$. By Lemma 2.4 it now suffices to show that $\phi, \psi$ are strict. Suppose $e \in X^p$ and $f \in (X \to Y)^q$. Then

$$\begin{array}{c} \phi(r_q^{X\to Y}(f))(e) = r_p^Y(\{r_q^{X\to Y}(f)(x) \mid x \in e\}) = \\ r_p^Y(\{r_q^Y(\{g(x) \mid g \in f\}) \mid x \in e\}. \end{array}$$

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 $y = r_q^Y(\{g(x) \mid g \in f\})$. Thus

$$\{r_q^Y(\{g(x) \mid g \in f\}) \mid x \in e\} = \{r_{p\wedge q}^Y(\{g(x) \mid g \in f \wedge x \in e\} \mid \exists x \in e\}$$

and so

$$\begin{array}{c} \phi(r_q^{X\to Y}(f))(e) = r_p^Y(\{r_{p\wedge q}^Y(\{g(x) \mid g \in f \wedge x \in e\} \mid \exists x \in e\}) = \\ r_{p\wedge q}^Y(\{g(x) \mid g \in f \wedge x \in e\}). \end{array}$$

Likewise

$$\begin{array}{c} r_q^{LX\to Y}(\{\phi(g) \mid g \in f\})(e) = r_q^Y(\{\phi(g)(e) \mid g \in f\}) = \\ r_q^Y(\{r_p^Y(\{g(x) \mid x \in e\}) \mid g \in f\}) = \\ r_q^Y(\{r_{p\wedge q}^Y(\{g(x) \mid x \in e \wedge g \in f\}) \mid \exists g \in f\}) = r_{p\wedge q}^Y(\{g(x) \mid x \in e \wedge g \in f\}) \end{array}$$

and so $\phi$ is strict.

To see that $\psi$ is strict, we compute

$$\begin{array}{c} \psi(r_q^{X\to Y}(f))(e) = r_q^{X\to Y}(f)(r_p^X(e)) = \\ r_q^Y(\{g(r_p^X(e)) \mid g \in f\}) \end{array}$$

and

$$\begin{array}{c} r_q^{LX\to Y}(\{\psi(g) \mid g \in f\})(e) = r_q^Y(\{\psi(g)(e) \mid g \in f\}) = \\ r_q^Y(\{g(r_p^Y(e)) \mid g \in f\}). \end{array}$$

So $\psi$ 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 $\to$ nor $\multimap$ define a cartesian closed structure on the category of pointed sets and strict maps. Clearly $\to$ defines a cartesian closed structure on the category of pointed sets and all maps, and we will see that $\multimap$ will be part of a symmetric monoidal closed structure on a category of domains.

For any set $X$, we introduce a free pointed structure $(LX, (\mu_p)_{p\in\Sigma})$ on $X$ 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 $\Sigma$ 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 $f\colon X \to Y$, we 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

$$r_p^Y(\{\hat{f}(e) \mid e \in E\}) = r_p^Y(\{r_{\exists x \in e}(\{f(x) \mid x \in e\}) \mid e \in E\}) = \\ r_p^Y(\{r_q^Y(\{f(x) \mid \exists e \in E \wedge x \in e\}) \mid p\}).$$

Since $q \supset p$, $p \wedge q = q$ and so by the last rule of Definition 2.2 the last part is simply

$$r_q^Y(\{f(x) \mid \exists e \in E \wedge x \in e\}),$$

which proves that $\hat{f}$ is pointed. □

**Lemma 2.9.** *$\Sigma \cong L1$ and so has a pointed structure.*

## 2.2 Domains and predomains

In this section we introduce our class of predomains, which will be the basis of the definition of the class of domains. Of course our main requirements for this class will be closure under certain constructions, a good supply of predomains, and that all endofunctions on domains have fixed points.

**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 $\Sigma$ are predomains.*

*Proof.* The set 1 is the equalizer of the identity and the constant map to 0 on the natural numbers. The set $\Sigma$ is isomorphic to $L1$. □

Since we will use predomains to model polymorphism it would be desirable to have a *set* of all predomains, such that we can define products over this set. This is unfortunately too much to ask for, so instead we will 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.** A domain is a pointed predomain. We denote by $\mathbf{Dom}_\perp$ the category of domains with strict maps and by $\mathbf{Dom}$ we denote the category of domains with all maps. By $\mathbf{D}$ we denote the set of pointed structures on objects of $\mathbf{P}$, i.e.,

$$\mathbf{D} = \{(B, (r_p)_{p \in \Sigma}) \mid B \in \mathbf{P}, (r_p)_{p \in \Sigma} \text{ is a pointed structure on } B\}$$

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 $\mathbf{Dom}_\perp$.

**Axiom 2.14.** *For every domain $A$ there is a function $\mathit{fix}_A \colon (A \to A) \to A$ such that*

- *$\mathit{fix}_A$ gives fixed points, i.e., for any $f \colon A \to A$,*

$$f(\mathit{fix}_A(f)) = \mathit{fix}_A$$

- *The maps $\mathit{fix}_A$ 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*

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"', -o] & A \arrow[d, "h", -o] \\
B \arrow[r, "g"] & B
\end{tikzcd}$$

*commutes, $h(\mathit{fix}_A(f)) = \mathit{fix}_B(g)$.*

## 3 The category of domains

In this section, we prove 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 $\mathbf{Dom}_\perp$.

**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, we notice that in the proof of Lemma 2.6 we show 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.** Suppose $f \colon A \times B \to C$ is a map between domains. We say that $f$ is *strict in the first variable* if for all $p \in \Sigma, e \in A^p, y \in B$

$$f(r_p^A(e), y) = r_p^C(\{f(x, y) \mid x \in e\}).$$

Likewise we define what it means for $f$ to be strict in the second variable. We say that $f$ is *bistrict* if $f$ 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

$$f(r_p^A(e), r_p^B(g)) = r_p^C(\{f(x, r_p^B(g)) \mid x \in e\}) = r_p^C(\{r_p^C\{f(x,y) \mid y \in g\} \mid x \in e\})$$

Since

$$\{r_p^C\{f(x,y) \mid y \in g\} \mid x \in e\} = \{f(x,y) \mid x \in e, y \in g\}$$

we get

$$f(r_p^A(e), r_p^B(g)) = r_p^C(\{f(x,y) \mid x \in e, y \in g\})$$

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 $\Sigma$). 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 $\eta$ 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 $\eta$. 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 [7, Definition 1.1]. We define the natural transformation $m^U$ to be $\eta$ of Lemma 3.10. We 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 [7, 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

$$\begin{tikzcd}
1 \times X \arrow[r, "\cong"] \arrow[d, "m_I^U \times id"'] & X \arrow[d, "\cong"] & X \times Y \arrow[r, "\cong"] \arrow[d, "\eta"'] & Y \times X \arrow[d, "\eta"] \\
I \times X \arrow[r, "\eta"] & I \otimes X & X \otimes Y \arrow[r, "\cong"] & Y \otimes X
\end{tikzcd}$$

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 $A, B, C$

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

so that $LA \otimes LB \cong L(A \times B)$ and by definition $L1 \cong I$. 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 [7, Definition 1.1], but this can be done exactly as in the proof of Lemma 3.11. □

**Lemma 3.13.** *The adjunction*

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

*is symmetric monoidal.*

*Proof.* The functors of the adjunction are symmetric monoidal and the left adjoint is strong, so the lemma follows from [7, 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 [7, 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 $f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Dom}_\perp$ 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.

The model described in this section will be modified to a parametric PILL$_Y$-model in Section 5.

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 $t$:

$$\begin{tikzcd}
(\pi^* f)(\vec{A}, D) \arrow[r, "t_{\vec{A},D}"] \arrow[d, "(\pi^* f)(id,i) = id"'] & g(\vec{A}, D) \arrow[d, "g(\vec{A},i)"] \\
(\pi^* f)(\vec{A}, D') \arrow[r, "t_{\vec{A},D'}"] & g(\vec{A}, D')
\end{tikzcd}$$

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

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

where $\pi_D$ 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

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

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

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

□

**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 [7]. □

**Lemma 4.6.** *The model*

$$\begin{tikzcd}
\mathbf{DFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=8] \arrow[dr] \arrow[rr, phantom, "\perp"] & & \mathbf{DFam}(\mathbf{Dom}) \arrow[ll, bend right=8] \arrow[dl] \\
& \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\} &
\end{tikzcd}$$

*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

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

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

We sum the above to the following

**Proposition 4.7.**

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

*is a PILL$_Y$-model.*

# 5 The parametric fibration

In this section, we basically apply a parametric completion process as in [11, 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 $B$. 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 $\mathbf{AdmRel}(\mathbf{Dom}_\perp)_{\mathrm{iso}}$ 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:

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

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 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

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

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 $x, y$

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

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

$$\begin{tikzcd}[column sep=small] & \bar{R} \arrow[dl] \arrow[dr] & & & \bar{S} \arrow[dl] \arrow[dr] & \\ A & & B & C & & D \end{tikzcd}$$

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

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

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

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

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 $\epsilon$:

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

The same reasoning applies for the rest of the linear 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

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

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 $\delta, e, m, m_I$ preserve relations is done likewise. The commutative diagrams of [7, 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 $\vec{R}\colon \mathbf{AdmRel}(\vec{\alpha}, \vec{\beta})$,

$$\begin{aligned} \mathit{dom}(s_{\vec{R}}) &= t_{\vec{\alpha}} \\ \mathit{codom}(s_{\vec{R}}) &= t_{\vec{\beta}} \\ s_{eq_{\vec{\alpha}}} &= (t_{\vec{\alpha}}, t_{\vec{\alpha}}) \end{aligned}$$

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 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

$$\begin{aligned} (f \otimes g)^r(\vec{R}) &= f^r(\vec{R}) \otimes g^r(\vec{R}) \\ (f \otimes g)^d(\vec{A}) &= f^d(\vec{A}) \otimes g^d(\vec{A}). \end{aligned}$$

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 $\epsilon$ 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. □

**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

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

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 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

$$\begin{tikzcd}
& R \arrow[ld] \arrow[rd] & \\
A & & B
\end{tikzcd}$$

correspond to strict maps preserving relations out of

$$\begin{tikzcd}
& !R \arrow[ld] \arrow[rd] & \\
!A & & !B
\end{tikzcd}$$

Since $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ has fibred products we may appeal to [7, Proposition 1.21]. □

**Lemma 5.7.** *The model*

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=5] \arrow[rr, phantom, "\perp"] \arrow[rd] & & \mathbf{PFam}(\mathbf{Dom}) \arrow[ll, bend right=5] \arrow[ld] \\
& \mathbf{PDom} &
\end{tikzcd}$$

*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

$$(\mathit{fix}_D)_{D\in\mathbf{D}}(\prod \alpha.\, (\alpha \to \alpha) \to \alpha)(\mathit{fix}_D)_{D\in\mathbf{D}},$$

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

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

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.**

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[rr, bend right=5] \arrow[dr] & & \mathbf{PFam}(\mathbf{Dom}) \arrow[ll, bend right=5] \arrow[ll, phantom, "\perp"] \arrow[dl] \\
& \mathbf{PDom} &
\end{tikzcd} \tag{4}$$

*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

$$(\mathbf{Dom}_\bot)^n_{\mathtt{iso}} \to \mathbf{Set},$$

and reindexing along a morphism from $m$ to $n$ in $\mathbf{PDom}$ is by composition with the functor

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

The category $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))$ is a fibred partial order over $\mathbf{DFam}(\mathbf{Set})$ and has as objects over

$$f\colon (\mathbf{Dom}_\bot)^n_{\mathtt{iso}} \to \mathbf{Set}$$

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 $f\colon (\mathbf{Dom}_\bot)^n_{\mathtt{iso}} \to \mathbf{Set}$ is an object of $\mathbf{DFam}(\mathbf{Set})_n$ and $h\colon (\mathbf{Dom}_\bot)^{n+1}_{\mathtt{iso}} \to \mathbf{Set}$ is a subfunctor of $\pi^* f = f \circ \pi$. We must define $(\prod h)\colon (\mathbf{Dom}_\bot)^n_{\mathtt{iso}} \to \mathbf{Set}$ 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

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

and on the other hand

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

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 pointed.*

*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 $\exists x \in e$. □

**Lemma 6.8.** *Suppose* $A, B$ *are domains, and* $\rho$ *is a sub-predomain of* $A\times B$ *then*

$$\{(x,y)\in A\times B \mid x\downarrow\supset (x,y)\in\rho\}$$

*is a subdomain of* $A\times B$.

*Proof.* For $f\colon A\multimap\Sigma$, define the set

$$\rho_f = \{(x,y)\in A\times B \mid f(x) \supset (x,y)\in\rho\}.$$

Since

$$\{(x,y)\in A\times B \mid x\downarrow\supset (x,y)\in\rho\} = \bigcap_{f\colon A\multimap\Sigma} \rho_f,$$

by Lemma 5.1 it suffices to show that $\rho_f$ is a subdomain of $A\times B$ for each $f$.

We first show that $\rho_f$ is a predomain. Consider the map $l_f\colon A\times B \to L(A\times B)$ defined as $l_f(x,y) = \{(x,y) \mid f(x)\}$. We have a pullback diagram

$$\begin{tikzcd}
\rho_f \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & L(\rho) \arrow[d] \\
A\times B \arrow[r, "l_f"] & L(A\times B)
\end{tikzcd}$$

because the pullback of the diagram is

$$\{(x,y) \in A \times B \mid \forall z \colon l_f(x,y).\, z \in \rho\},$$

but $z \in l_f(x,y)$ is equivalent to $f(x) \wedge z = (x,y)$. Since the category of predomains is closed under limits, $\rho_f$ is a predomain.

To show that $\rho_f$ is subpointed, suppose $e \in \rho_f^p$ for some $p \in \Sigma$. We must show that $r_p^{A\times B}(e) \in \rho_f$. So suppose $f(\pi(r_p^{A\times B}(e)))$ where $\pi \colon A \times B \multimap A$ is the projection. Using that $f, \pi$ are strict we see that

$$\begin{array}{l} f(\pi(r_p^{A\times B}(e))) = f(r_p^A(\{x \mid \exists y.\, (x,y) \in e\})) = \\ \mu_p(\{f(x) \mid \exists y \colon B.\, (x,y) \in e\}) = \bigcup_{(x,y)\in e} f(x), \end{array}$$

so that $f(\pi(r_p^{A\times B}(e)))$ implies that there exists $(x,y) \in e$ such that $f(x)$. Since $e \in \rho_f^p$, we conclude $p$ and $(x,y) \in \rho$ and so $r_p^{A\times B}(e) = r_\top^{A\times B}(e) = (x,y) \in \rho$. We have proved that $\rho_f$ is subpointed. □

*Proof of Lemma 6.5.* For readability, we will assume that everything here takes place in the fiber over $0 \in$ **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 $\rho$ is admissible. Then

$$(x,y).\, x \downarrow\supset \rho(x,y)$$

is admissible by 6.8. To see that

$$(x \colon LA, y \colon LB).\, x \downarrow\supset\subset y\downarrow$$

is admissible we simply notice that by Lemma 6.7 it is the equalizer of the maps $L(!) \circ \pi_A$ and $L(!) \circ \pi_B$, where $\pi_A, \pi_B$ denote the projections.

The subset

$$\{(x,y) \in A \times B \mid \phi\} = \bigcap_{z \in \{0 \mid \phi\}} A \times B$$

is a predomain, for $A, B$ domains, and so by Lemma 6.8

$$\{(x,y) \colon A \times B \mid x \downarrow \wedge y \downarrow\supset \phi\}$$

is admissible.

If $\phi$ is a proposition and $\rho$ 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 $\rho'$ 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 6.9.** *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

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

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:

$$\begin{tikzcd} \mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[d] & \mathbf{LinAdmRel} \arrow[d] \arrow[l] \\ \mathbf{PDom} & \mathbf{AdmRelCtx} \arrow[l] \end{tikzcd}$$

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 $\Phi$ 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 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. 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

$$(\exists x \in g \mathrel{\supset\!\subset} \exists y \in h) \wedge \forall x, y \,.\, x \in g, y \in h \supset (x, y) \in f^r(\vec{R}) \tag{8}$$

If we apply the ! in $\mathbf{LinAdmRel}$ to $\Phi(f^r, f^d)$ we obtain

$$\vec{R} \mapsto \{(g, h) \in Lf^d(\vec{A}) \times Lf^d(\vec{B}) \mid (g \downarrow \mathrel{\supset\!\subset} h \downarrow) \wedge g \downarrow \supset (\epsilon g, \epsilon h) \in f^r(\vec{R})\}$$

But since $g \downarrow \mathrel{\supset\!\subset} \exists x \in g$ 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 $\Phi$ 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 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 $\Phi$ preserves tensor. □

**Theorem 6.10.** *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. □

**Remark 6.11.** As mentioned earlier, the LAPL-structure (5) gives rise to a family of LAPL-structures, since we have one for each model of synthetic domain theory (SDT). To see how this work, let us focus on the case of the model being a realizability topos satisfying the strong completeness axiom [5] with predomains given as the well-complete objects.

Such a realizability topos is a large category (we have a full class of objects) and thus lives in a theory of classes. The well-complete objects constitute a large subcategory of the realizability topos, and thus the categories constructed above, such as $\mathbf{Dom}_\perp$ are large categories. When we talk about functors above, for example when we define types as being given by pairs of functors $(f^d, f^r)$ where $f^d \colon \mathbf{Dom}_\perp^n \to \mathbf{Dom}_\perp$, or when we talk about the lifting functor, we mean functors in the usual sense, i.e., maps of classes satisfying the usual requirements.

# 7 Proving consequences of parametricity for Lily$_{\text{strict}}$

In [13] a strict version of the language Lily [1] is given, which we shall call Lily$_{\text{strict}}$. By strict we mean intuitively that 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 (see Definition 7.5 and Theorem 7.10 below). 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 a model of 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 Lily$_{\text{strict}}$.

The model of PILL$_Y$ in (5) is based on the interpretation of Lily$_{\text{strict}}$ given in [13]. In this section we show that the two interpretations of PILL$_Y$ and Lily 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 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. One example of such a model is a realizability topos satisfying the strong completeness axiom [5] where we take predomains to be the well-complete objects.

In this section we will assume that we have chosen one such model, in which any $\prod_2^0$-sentence holds iff it holds in reality. This is for example the case for the realizability model mentioned above. The reason for this assumption is that adequacy (Theorem 7.9 below), is proved in the internal language of the model, and will only hold in the real world under this assumption (see also Section 8 of [13]).

We emphasize that the results we prove about Lily$_{\text{strict}}$ hold in general and are not relative to any model. To prove the results, however, we need to use a model satisfying the above conditions, and so the results only rely on the existence of such a model, which holds as argued above.

## 7.1 The language Lily$_{\text{strict}}$

This subsection sums up some definitions and results from [13]. In particular we define the language Lily$_{\text{strict}}$ with two operational semantics, a call-by-value a call-by-name semantics. For each of these semantics we can define a concept of ground contextual equivalence corresponding to observing termination at !- types. An interpretation of the language is defined, and shown to be adequate with respect to both notions of contextual equality, and using the interpretation it is proved that the two ground contextual equivalences coincide.

The types of Lily$_{\text{strict}}$ are

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

where $\alpha$ ranges over an infinite set of type variables. Except for $\otimes, I$ these are exactly the types of PILL$_Y$. We write $\vdash_\Xi \sigma \colon \mathsf{Type}$ to mean that $\sigma$ is a well formed type with free type variables contained in $\Xi$.

Typing judgements of Lily$_{\text{strict}}$ are of the form

$$\Gamma \mid \delta \vdash_\Xi t \colon \sigma$$

where $\Gamma$ is the context of free variables, i.e., a finite mapping of variables to types, 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 $\Gamma$, i.e., $\{x_1, \dots, x_n\}$. $\Xi$ is a finite set of free type variables containing the free type variables of $\sigma_1, \dots, \sigma_n, \sigma$. We write $\Xi, \alpha$ mean $\Xi \cup \alpha$ *and* $\alpha \notin \Xi$. $\delta$ is a labeling of the variables in the domain of $\Gamma$, i.e., a map from $\{x_1, \dots, x_n\}$ to $\{0, 1\}$. Intuitively $\delta(x_i) = 1$ means that $t$ is strict in $x_i$.

We use the notation $\vdash_\Xi \Gamma$ to mean that $\Gamma$ is a well-formed context with free variables contained in $\Xi$.

The term formation rules are given in Figure 1. 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 $\delta$ 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 $\delta, \delta'$ labellings of the same set of variables, the notation $\delta \vee \delta'$ is the labeling mapping $x\colon$ *dom*$(\delta)$ to $\max(\delta(x), \delta'(x))$. We use **0** to denote the constant zero labeling.

$$\frac{}{\Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_\Xi x\colon \sigma} \qquad \frac{\Gamma \mid \delta, x :_1 \sigma \vdash_\Xi t\colon \tau}{\Gamma \mid \delta \vdash_\Xi \lambda x :_1 \sigma.\, t\colon \sigma \multimap \tau}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi s\colon \sigma \multimap \tau \qquad \Gamma \mid \delta' \vdash_\Xi t\colon \sigma}{\Gamma \mid \delta \vee \delta' \vdash_\Xi s(t)\colon \tau} \quad \frac{\Gamma \mid \delta \vdash_\Xi t\colon \sigma}{\Gamma \mid \mathbf{0} \vdash_\Xi !t\colon !\sigma}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi s\colon !\sigma \qquad \Gamma \mid \delta', x :_{\_} \sigma \vdash_\Xi t\colon \tau}{\Gamma \mid \delta \vee \delta' \vdash_\Xi \text{let } !x \text{ be } s \text{ in } t} \quad \frac{\Gamma \mid \delta \vdash_{\Xi,\alpha} t\colon \sigma \qquad \vdash_\Xi \Gamma}{\Gamma \mid \delta \vdash_\Xi \Lambda\alpha.\, t\colon \prod \alpha.\, \sigma}$$

$$\frac{\Gamma \mid \delta \vdash_\Xi t\colon \prod \alpha.\, \sigma \qquad \vdash_\Xi \tau\colon \mathsf{Type}}{\Gamma \mid \delta \vdash_\Xi t(\tau)\colon \sigma[\tau/\alpha]} \quad \frac{\Gamma \mid \delta, x :_{\_} \sigma \vdash_\Xi t\colon \sigma}{\Gamma \mid \delta \vdash_\Xi \text{rec } x\colon \sigma.\, t\colon \sigma}$$

Figure 1: Term formation rules for $\text{Lily}_{\text{strict}}$

**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* $\textit{Lily}_{\textit{strict}}$*, and* $\Xi \vdash \sigma\colon \mathsf{Type}$ *is a type in* $\textit{Lily}_{\textit{strict}}$ *and* $x \notin$ *dom*$(\Gamma)$. *Then* $\Gamma \mid \delta, x :_0 \sigma \vdash_\Xi t\colon \tau$ *is typable in* $\textit{Lily}_{\textit{strict}}$.

In Figure 2 we define two operational semantics for $\text{Lily}_{\text{strict}}$. 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

$$v ::= \lambda x\colon \sigma.\, t \mid !t \mid \Lambda\alpha.\, t.$$

In Figure 2 we use the notation $t \Downarrow v$ in some rules. This means that each of these rules exist both in the definition of the $\Downarrow^n$ and the $\Downarrow^s$ semantics. We write $t \Downarrow^n$ 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 $\sigma$-context is a term $x :_{\_} \sigma \vdash C\colon !\tau$ for some type $\tau$, 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 $t \equiv^{\text{s}}_{\text{gnd}} t'$ if for all ground $\sigma$-contexts $C[-]$, $C[t] \Downarrow^s$ iff $C[t'] \Downarrow^s$. Likewise $t \equiv^{\text{n}}_{\text{gnd}} t'$ if for all ground $\sigma$-contexts $C[-]$, $C[t] \Downarrow^n$ iff $C[t'] \Downarrow^n$

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

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

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

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

Figure 2: Operational semantics of Lily$_{\text{strict}}$

The idea of $\equiv^{\text{n}}_{\text{gnd}}, \equiv^{\text{s}}_{\text{gnd}}$ 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. By observable behavior we have chosen 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 define an interpretation of Lily$_{\text{strict}}$. Each type $\vdash_{\alpha_1,\dots\alpha_n} \sigma$ has two interpretations. The first $\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^d$ 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 $\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^d$ as $\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^d_{\vec{D}}$. The second interpretation of a type is the relational interpretation $\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^r$. This is a family of relations

$$(\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^r_{\vec{R}} \colon \mathbf{AdmRel}(\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^d_{\vec{A}}, \llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^d_{\vec{B}}))_{\vec{R}\colon \mathbf{AdmRel}(\vec{A}, \vec{B})}$$

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.

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.** *Suppose $(f_i\colon A_i \multimap B_i)_{i \leq n}$ are isomorphisms in $(\mathbf{Dom}_\perp)_{\mathtt{iso}}$ and $\vdash_{\alpha_1,\dots\alpha_n} \sigma\colon \mathsf{Type}$ is a type in Lily$_{strict}$. Then there exists an isomorphism $\llbracket \sigma \rrbracket^d(\vec{f})$ in $\mathbf{Dom}_\perp$ such that $\llbracket \vdash_{\vec{\alpha}} \sigma \rrbracket^r_{(\langle f_i \rangle)_i}$ 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

$$\Gamma \mid \delta \vdash_{\Xi} t\colon \sigma$$

we define first the interpretation of the context $\Gamma$:

$$\llbracket \vdash_{\Xi} x_1\colon \sigma_1, \dots, \sigma_n\colon \sigma_n \rrbracket^d = (\textstyle\prod_i \llbracket \vdash_{\Xi} \sigma_i \rrbracket_{\vec{D}})_{\vec{D}}.$$

Since the labeling does not play a role in the interpretation of terms and is uniquely determined by $\Xi, \Gamma, t$, 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

$$(\llbracket \Gamma \vdash_{\Xi} t\colon \tau \rrbracket_{\vec{D}} \colon \llbracket \vdash_{\Xi} \Gamma \rrbracket^d_{\vec{D}} \to \llbracket \tau \rrbracket^d_{\vec{D}})_{\vec{D}}$$

$$
\begin{array}{rcl}
\llparenthesis \vdash_{\vec{\alpha}} \alpha_i \rrparenthesis^d_{\vec{D}} & = & (D_i) \\
\llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{D}} & = & \llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^d_{\vec{D}} \multimap \llparenthesis \vdash_{\vec{\alpha}} \tau \rrparenthesis^d \\
\llparenthesis \vdash_{\vec{\alpha}} !\sigma \rrparenthesis^d_{\vec{D}} & = & L\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis_{\vec{D}} \\
\llparenthesis \vdash_{\vec{\alpha}} \prod \alpha.\, \sigma \rrparenthesis^d_{\vec{D}} & = & \{\pi \in \prod_{D \in \mathbf{D}} \llparenthesis \vdash_{\Xi,\alpha} \sigma \rrparenthesis^d_{\vec{D},D} \mid \forall D, D' \in \mathbf{D}. \\
 & & \forall R\colon \mathbf{AdmRel}(D, D').\, \llparenthesis \vdash_{\Xi,\alpha} \sigma \rrparenthesis^r_{\vec{eq},R}(\pi_D, \pi_{D'})\}
\end{array}
$$

$$
\begin{array}{rcl}
\llparenthesis \vdash_{\vec{\alpha}} \alpha_i \rrparenthesis^r_{\vec{R}\colon \mathbf{AdmRel}(\vec{A},\vec{B})} & = & R_i \\
\llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^r_{\vec{R}\colon \mathbf{AdmRel}(\vec{A},\vec{B})} & = & \{(f, g) \in \llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{A}} \times \llparenthesis \vdash_{\vec{\alpha}} \sigma \multimap \tau \rrparenthesis^d_{\vec{B}} \mid \\
 & & \forall (x, y) \in \llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis^r_{\vec{R}}.\, (f(x), g(y)) \in \llparenthesis \vdash_{\vec{\alpha}} \tau \rrparenthesis^r_{\vec{R}}\} \\
\llparenthesis \vdash_{\vec{\alpha}} !\sigma \rrparenthesis^r_{\vec{R}\colon \mathbf{AdmRel}(\vec{A},\vec{B})} & = & \{(e, f)\colon L\llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}) \times L\llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}) \mid \\
 & & (\forall x\colon \llbracket \Xi \mid \sigma \rrbracket^d(\vec{A}).\, x \in e \supset \exists y \in f \supset \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})(x, y)) \wedge \\
 & & (\forall y\colon \llbracket \Xi \mid \sigma \rrbracket^d(\vec{B}).\, y \in f \supset \exists x \in e \supset \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})(x, y))\} \\
\llparenthesis \vdash_{\vec{\alpha}} \prod \alpha.\, \sigma \rrparenthesis^r_{\vec{R}\colon \mathbf{AdmRel}(\vec{A},\vec{B})} & = & \{(\pi, \pi') \in \llparenthesis \vdash_{\Xi} \prod \alpha.\, \sigma \rrparenthesis^d_{\vec{A}} \times \llparenthesis \vdash_{\Xi} \prod \alpha.\, \sigma \rrparenthesis^d_{\vec{B}} \mid \\
 & & \forall D, D' \in \mathbf{D}.\, \forall R\colon \mathbf{AdmRel}(D, D').\, \llparenthesis \vdash_{\Xi,\alpha} \sigma \rrparenthesis^r_{\vec{eq},R}(\pi_D, \pi_{D'})\}
\end{array}
$$

Figure 3: Interpretation of types

The interpretation of terms is defined in Figure 4. In the definition of the interpretation of the let-expression we have used the $(-)^*$-notation to denote the strict map

$$L\llparenthesis \vdash_{\Xi} \sigma \rrparenthesis \multimap \llparenthesis \vdash_{\Xi} \tau \rrparenthesis$$

corresponding to the set theoretic map

$$\llparenthesis \vdash_{\Xi} \sigma \rrparenthesis \to \llparenthesis \vdash_{\Xi} \tau \rrparenthesis$$

described. The notation

$$\llparenthesis \sigma \rrparenthesis^d(id_{\vec{D}}, i)$$

refers to the morphism of Lemma 7.7.

$$
\begin{array}{rcl}
\llparenthesis \Gamma \vdash_{\Xi} x_i\colon \sigma_i \rrparenthesis_{\vec{D}}(\vec{x}) & = & x_i \\
\llparenthesis \Gamma \vdash_{\Xi} \lambda x\colon \sigma.\, t\colon \sigma \multimap \tau \rrparenthesis_{\vec{D}}(\vec{x}) & = & d\colon \llparenthesis \vdash_{\Xi} \sigma \rrparenthesis^d_{\vec{D}} \mapsto \llparenthesis \Gamma, x\colon \sigma \vdash_{\Xi} t\colon \tau \rrparenthesis_{\vec{D}}(\vec{x}, d) \\
\llparenthesis \Gamma \vdash_{\Xi} s(t)\colon \tau \rrparenthesis_{\vec{D}}(\vec{x}) & = & \llparenthesis \Gamma \vdash_{\Xi} s\colon \sigma \multimap \tau \rrparenthesis_{\vec{D}}(\vec{x})(\llparenthesis \Gamma \vdash_{\Xi} t\colon \sigma \rrparenthesis_{\vec{D}}(\vec{x})) \\
\llparenthesis \Gamma \vdash_{\Xi} !t \rrparenthesis_{\vec{D}}(\vec{x}) & = & \{\llparenthesis \Gamma \vdash_{\Xi} t \rrparenthesis_{\vec{D}}(\vec{x})\} \\
\llparenthesis \Gamma \vdash_{\Xi} \text{let } !x \text{ be } s \text{ in } t \rrparenthesis_{\vec{D}}(\vec{x}) & = & (d\colon \llparenthesis \vdash_{\Xi} \sigma \rrparenthesis \mapsto \llparenthesis \Gamma, x\colon \sigma \vdash_{\Xi} t \rrparenthesis_{\vec{D}}(\vec{x}, d))^* \\
 & & (\llparenthesis \Gamma \vdash_{\Xi} s\colon !\sigma \rrparenthesis_{\vec{D}}(\vec{x})) \\
\llparenthesis \Gamma \vdash_{\Xi} \Lambda\alpha.\, t\colon \prod \alpha.\, \sigma \rrparenthesis_{\vec{D}}(\vec{x}) & = & ((\llparenthesis \Gamma \vdash_{\Xi,\alpha} t\colon \sigma \rrparenthesis_{\vec{D},D}(\vec{x}))_{D \in \mathbf{D}} \\
\llparenthesis \Gamma \vdash_{\Xi} t(\tau)\colon \sigma[\tau/\alpha] \rrparenthesis_{\vec{D}}(\vec{x}) & = & \llparenthesis \vdash_{\Xi} \sigma \rrparenthesis^d(id_{\vec{D}}, i)((\llparenthesis \Gamma \vdash_{\Xi} t\colon \prod \alpha.\, \sigma \rrparenthesis_{\vec{D}}(\vec{x}))_D \\
 & & \text{where } D \in \mathbf{D} \text{ and } i\colon D \multimap \llparenthesis \vdash_{\Xi} \tau \rrparenthesis^d_{\vec{D}} \text{ is an iso} \\
\llparenthesis \Gamma \vdash_{\Xi} \text{rec}\, x\colon \sigma.\, t\colon \sigma \rrparenthesis_{\vec{D}}(\vec{x}) & = & \mathit{fix}(d\colon \llparenthesis \vdash_{\Xi} \sigma \rrparenthesis \mapsto \llparenthesis \Gamma, x\colon \sigma \vdash_{\Xi} t\colon \sigma \rrparenthesis_{\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.** *Suppose* $\Gamma \vdash_\Xi t \colon \tau$. *Then*

- $(\!|\Gamma \vdash_\Xi t \colon \tau|\!)$ *is well defined*
- *If* $\Gamma = x_1 \colon \sigma_1, \ldots, x_n \colon \sigma_n$ *and* $\delta(x_i) = 1$ *then for each vector* $\vec{D}$ *of domains, the function* $(\!|\Gamma \mid \delta \vdash_\Xi t \colon \tau|\!)_{\vec{D}}$ *is strict in the* $i$*'th variable.*
- *If* $\vec{R} \colon \mathbf{AdmRel}(\vec{D}, \vec{D}')$ *is a vector of admissible relations, and* $x_1, \ldots x_n$, $x'_1, \ldots, x'_n$ *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 [13].

**Theorem 7.9 (Adequacy).** *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).** *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.** In [13] adequacy is proved using the intuitionistic set theory in which the category of domains live. This means that for a concrete model of Synthetic Domain Theory, adequacy as stated in [13] only holds in the internal logic of the model as stated using an encoding of the operational semantics of Lily$_{\mathrm{strict}}$ in natural numbers.

In this section, we have assumed that we are using a model of SDT in which $\prod^0_2$-sentences only hold if they hold in reality. Since statements of the form $t \Downarrow^s$, $t \Downarrow^n$ are $\Sigma^0_1$, statements $t \equiv^{\mathrm{s}}_{\mathrm{gnd}} t'$ and $t \equiv^{\mathrm{n}}_{\mathrm{gnd}} t'$ are $\prod^0_2$ and so Theorem 7.9 is not just a theorem in the internal language of the model, but a real theorem. The same holds of course for Theorem 7.10.

## 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* $\phi$ *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$_Y \setminus \otimes$, $\llbracket \vec{\alpha} \vdash \sigma \rrbracket^d$ is a functor from $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}}$ to $\mathbf{Dom}_\perp$. Since $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis$ is an indexed family of domains, it does not make sense to ask if $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis = \llbracket \vec{\alpha} \vdash \sigma \rrbracket$, but we can still compare the values of $\llbracket \vec{\alpha} \vdash \sigma \rrbracket$ on objects with $\llparenthesis \vdash_{\vec{\alpha}} \sigma \rrparenthesis$.

**Lemma 7.13.** *For all types $\Xi \vdash \sigma$ of PILL$_Y \setminus \otimes$, the object parts of $\llbracket - \rrbracket^d$ and $\llbracket - \rrbracket^r$ agree with $\llparenthesis - \rrparenthesis^d$ and $\llparenthesis - \rrparenthesis^r$. Moreover, for $\vec{i}$ a vector of strict isomorphism between domains, $\llbracket - \rrbracket^d(\vec{i})$ is equal to $\llparenthesis - \rrparenthesis^d(\vec{i})$ 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

$$\begin{array}{c} \llbracket \Xi \mid !\sigma \rrbracket^d(\vec{A}) = L(\llbracket \Xi \mid \sigma \rrbracket^d(\vec{A})) \\ \llparenthesis \Xi \mid !\sigma \rrparenthesis^d(\vec{A}) = L(\llparenthesis \Xi \mid \sigma \rrparenthesis^d(\vec{A})) \end{array}$$

For the relational interpretation, we have for $\vec{R}$: $\mathbf{AdmRel}(\vec{A}, \vec{B})$.

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

On the other hand, $\llbracket \Xi \vdash !\sigma \rrbracket^r(\vec{R})$ is the image of the span obtained by applying the lifting functor $L$ to both maps in the span

$$\begin{tikzcd} & \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R}) \arrow[dl] \arrow[dr] & \\ \llbracket \Xi \vdash \sigma \rrbracket^d(\vec{A}) & & \llbracket \Xi \vdash \sigma \rrbracket^d(\vec{B}). \end{tikzcd}$$

So $\llbracket \Xi \vdash !\sigma \rrbracket^r(\vec{R})$ consists of lifts of pairs from $\llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})$, i.e., pairs $(e, f)$ such that $e \downarrow \supset\subset f \downarrow$ and $x \in e, y \in f \supset (x, y) \in \llbracket \Xi \vdash \sigma \rrbracket^r(\vec{R})$.

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 $(\llbracket \sigma \rrbracket^d(\vec{i}), id_{\llbracket \sigma \rrbracket^d(\vec{B})})$ is an isomorphism from $\llbracket \sigma \rrbracket^r(\langle \vec{i} \rangle)$ to $eq_{\llbracket \sigma \rrbracket^d(\vec{B})}$, i.e.,

$$(\llbracket \sigma \rrbracket^d(\vec{i}), id_{\llbracket \sigma \rrbracket^d(\vec{B})})^* eq_{\llbracket \sigma \rrbracket^d(\vec{B})} = \llbracket \sigma \rrbracket^r(\langle \vec{i} \rangle).$$

We can use this to prove

$$\langle \llbracket \sigma \rrbracket^d(\vec{i}) \rangle = (\llbracket \sigma \rrbracket^d(\vec{i}), id_{\llbracket \sigma \rrbracket^d(\vec{B})})^* eq_{\llbracket \sigma \rrbracket^d(\vec{B})} = \llbracket \sigma \rrbracket^r(\langle \vec{i} \rangle) = \llparenthesis \sigma \rrparenthesis^r(\langle \vec{i} \rangle) = \langle \llparenthesis \sigma \rrparenthesis^d(\vec{i}) \rangle,$$

and so since their graphs agree, $\llbracket \sigma \rrbracket^d(\vec{i}) = \llparenthesis \sigma \rrparenthesis^d(\vec{i})$. □

*Proof of Theorem 7.12.* The interpretation $\phi$ 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.
- $\phi(t)$ 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.

$$\begin{gathered}\phi(\Xi \mid \Gamma; - \vdash Y\colon \prod \alpha.\, !(!\alpha \multimap \alpha) \multimap \alpha) = \\ \Gamma \mid \mathbf{0} \vdash_{\Xi} \Lambda\alpha.\, \lambda t\colon !(!\alpha \multimap \alpha).\, \mathrm{rec}\ x\colon \alpha.\ \mathrm{let}\ !u\ \mathrm{be}\ t\ \mathrm{in}\ u(!x)\end{gathered}$$

$$\phi(\Xi \mid \Gamma, x\colon \sigma; - \vdash x\colon \sigma) = \Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_{\Xi} x\colon \sigma$$

$$\phi(\Xi \mid \Gamma; x\colon \sigma \vdash x\colon \sigma) = \Gamma \mid \mathbf{0}, x :_1 \sigma \vdash_{\Xi} x\colon \sigma$$

$$\frac{\begin{gathered}\phi(\Xi \mid \Gamma; \Delta \vdash t\colon \sigma \multimap \tau) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(t)\colon \sigma \multimap\ \tau, \\ \phi(\Xi \mid \Gamma; \Delta' \vdash u\colon \sigma) = \Gamma, \Delta' \mid \delta' \vdash_{\Xi} \phi(u)\colon \sigma\end{gathered}}{\phi(\Xi \mid \Gamma; \Delta, \Delta' \vdash t\, u\colon \tau) = \Gamma, \Delta, \Delta' \mid \delta \vee \delta' \vdash_{\Xi} \phi(t)\phi(u)\colon \tau}$$

$$\frac{\phi(\Xi \mid \Gamma; \Delta, x\colon \sigma \vdash t\colon \tau) = \Gamma, \Delta \mid \delta, x :_1 \sigma \vdash_{\Xi} \phi(t)\colon \tau}{\phi(\Xi \mid \Gamma; \Delta \vdash \lambda^{\circ} x\colon \sigma.\, t\colon \tau) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \lambda x\colon \sigma.\, \phi(t)\colon \sigma \multimap \tau}$$

$$\frac{\phi(\Xi \mid \Gamma; - \vdash t\colon \sigma) = \Gamma \mid \delta \vdash_{\Xi} \phi(t)\colon \sigma}{\phi(\Xi \mid \Gamma; - \vdash !t\colon !\sigma) = \Gamma \mid \mathbf{0} \vdash_{\Xi} !\phi(t)\colon \sigma}$$

$$\frac{\phi(\Xi, \alpha \mid \Gamma; \Delta \vdash t\colon \sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi, \alpha} \phi(t)\colon \sigma \quad \Xi \mid \Gamma; \Delta}{\phi(\Xi \mid \Gamma; \Delta \vdash \Lambda\alpha.\, t\colon \prod \alpha.\, \sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \Lambda\alpha.\, \phi(t)\colon \prod \alpha\ .\, \sigma}$$

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

$$\frac{\begin{gathered}\phi(\Xi \mid \Gamma; \Delta \vdash s\colon !\sigma) = \Gamma, \Delta \mid \delta \vdash_{\Xi} \phi(s)\colon !\sigma \\ \phi(\Xi \mid \Gamma, x\colon \sigma; \Delta' \vdash t\colon \tau) = \Gamma, \Delta', x\colon \sigma \mid \delta' \vdash_{\Xi} \phi(t)\colon \tau\end{gathered}}{\phi(\Xi \mid \Gamma; \Delta, \Delta' \vdash \mathrm{let}\ !x\ \mathrm{be}\ s\ \mathrm{in}\ t\colon \tau) = \Gamma, \Delta, \Delta' \mid \delta \vee \delta' \vdash_{\Xi} \mathrm{let}\ !x\ \mathrm{be}\ \phi(s)\ \mathrm{in}\ \phi(t)\colon \tau}$$

Figure 5: Inductive definition of $\phi$.

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 $\mathrm{PILL}_Y \setminus \otimes$ with free variables

$$x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n; x_1'\colon \sigma_1', \ldots, x_m'\colon \sigma_m'$$

$\llbracket t \rrbracket_{\vec{A}}$ 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 $\eta$ 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 $\sigma$. 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

$$\begin{tikzcd} L(\llbracket \sigma \rrbracket^d(\vec{A})) \arrow[r, multimap, "\delta"] & LL(\llbracket \sigma \rrbracket^d(\vec{A})) \arrow[r, multimap, "L\llbracket t \rrbracket_{\vec{A}}"] & L(\tau^d(\vec{A})) \end{tikzcd}$$

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

$$\llbracket t \rrbracket_{\vec{A}}(\textstyle\bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j) = (\llbracket t' \rrbracket_{\vec{A},D} \bigotimes_i \{x_i\} \otimes \bigotimes_j x'_j)_{D \in \mathbf{D}} = (\llparenthesis t' \rrparenthesis_{\vec{A},D}(\vec{x}, \vec{x}'))_{D \in \mathbf{D}} = \llparenthesis t \rrparenthesis_{\vec{A}}(\vec{x}, \vec{x}').$$

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 $k\colon D \multimap \llbracket \tau \rrbracket^d(\vec{A})$, and the $\llparenthesis - \rrparenthesis$ 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

$$\llparenthesis \text{let } !x \text{ be } s \text{ in } t \rrparenthesis_{\vec{A}}(x, y, z) = \widehat{\llparenthesis t \rrparenthesis}_{\vec{A}}(x, \llparenthesis s \rrparenthesis_{\vec{A}}(x, y), z)$$

where

$$\widehat{\llparenthesis t \rrparenthesis}_{\vec{A}}\colon \llparenthesis \Gamma \rrparenthesis^d(\vec{A}) \times L(\llparenthesis \sigma \rrparenthesis^d(\vec{A})) \times \llparenthesis \Delta \rrparenthesis^d(\vec{A})) \to \llparenthesis \tau \rrparenthesis^d(\vec{A})$$

is the unique extension of

$$\llparenthesis t\rrparenthesis_{\vec{A}}\colon \llparenthesis\Gamma\rrparenthesis^d(\vec{A}) \times \llparenthesis\sigma\rrparenthesis^d(\vec{A}) \times \llparenthesis\Delta\rrparenthesis^d(\vec{A})) \to \llparenthesis\tau\rrparenthesis^d(\vec{A})$$

which is strict in the second variable. Since by induction, for all $u, v, w$

$$\llbracket t\rrbracket_{\vec{A}}(\{u\} \otimes \{v\} \otimes w) = \llparenthesis t\rrparenthesis_{\vec{A}}(u, v, w)$$

we get that also for all $v'$

$$\llbracket t\rrbracket_{\vec{A}}(\{u\} \otimes v' \otimes w) = \widehat{\llparenthesis t\rrparenthesis}_{\vec{A}}(u, v', w),$$

since $\llbracket t\rrbracket_{\vec{A}}$ is strict. Thus

$$\begin{array}{rcl} \llparenthesis \text{let } !x \text{ be } s \text{ in } t\rrparenthesis_{\vec{A}}(x, y, z) & = & \llbracket t\rrbracket_{\vec{A}}(\{x\} \otimes \llparenthesis s\rrparenthesis_{\vec{A}}(x, y) \otimes z) = \\ \llbracket t\rrbracket_{\vec{A}}(\{x\} \otimes \llbracket s\rrbracket_{\vec{A}}(\{x\} \otimes y) \otimes z) & = & \llbracket \text{let } !x \text{ be } s \text{ in } t\rrbracket_{\vec{A}}(\{x\} \otimes y \otimes z) \end{array}$$

For fixed points, we have defined $\llbracket Y\rrbracket = (\mathit{fix}_D)_{D\in\mathbf{D}}$ 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 $\llbracket Y\rrbracket = (a_D)_{D\in\mathbf{D}}$, 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 $\llparenthesis\phi(Y)\rrparenthesis$, consider first

$$\llparenthesis f :_1 !(!\alpha \multimap \alpha), x :_0 \alpha \vdash_\alpha \text{let } !u \text{ be } f \text{ in } u(!x)\rrparenthesis_D$$

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

$$\llparenthesis f :_1 !(!\alpha \multimap \alpha) \vdash_\alpha \text{rec } x :_0 \alpha.\, \text{let } !u \text{ be } f \text{ in } u(!x)\rrparenthesis_D$$

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

$$\begin{gathered} \llparenthesis\phi(Y)\rrparenthesis = \llparenthesis\Lambda\alpha.\, \lambda f\colon !(!\alpha \multimap \alpha).\, \text{rec } x :_0 !\alpha.\, \text{let } !u \text{ be } f \text{ in } u(x)\rrparenthesis = \\ (a_D)_D = \llbracket Y\rrbracket \end{gathered}$$

For the last statement of the theorem, suppose $t\colon \sigma \multimap \tau$, $u\colon \tau \multimap \omega$ are terms of 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 $\psi$ of PILL$_Y$ into PILL$_Y \setminus \otimes$ such that for any parametric PILL$_Y$-model $X$ the diagram*

$$\begin{tikzcd} PILL_Y \arrow[rr, "\psi"] \arrow[dr, "\llbracket - \rrbracket"'] & & PILL_Y \setminus \otimes \arrow[dl, "\llbracket - \rrbracket"] \\ & X & \end{tikzcd}$$

*commutes up to natural isomorphism. To be more precise, there exists a family of isomorphisms $f_\sigma \colon [\![\sigma]\!] \to [\![\psi(\sigma)]\!]$ 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 $\psi$ 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 $\beta, \eta$- 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 $\sigma$ to $\tau$ 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 $\sigma$ of Lily$_{strict}$, the objects $\sigma$ 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 [3]. Applying $\phi$ 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 $\alpha$ only appears positively in $\sigma$ induce functors on $\mathbf{Lily}$.

**Corollary 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 $\phi$.

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, $\phi$ 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* $\alpha \vdash \sigma(\alpha) \colon \mathsf{Type}$ *of Lily*$_{strict}$*, there exists a closed type* $\tau$ *of Lily*$_{strict}$ *such that* $\tau$ *and* $\sigma(\tau)$ *are isomorphic as objects of* $\mathbf{Lily}$.

*Proof.* We can define $\tau$ 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 $\tau'$ and morphisms $\sigma(\tau') \multimap \tau'$, $\tau' \multimap \sigma(\tau')$ definable in Lily$_{\text{strict}}$. By functoriality of $\phi$, 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 $\tau'$ and $\sigma(\tau')$ are isomorphic in $\mathbf{Lily}$. □

# 8 Conclusion

We have constructed an LAPL-structure based on the interpretation of Lily$_{\text{strict}}$ into models of synthetic domain theory presented in [13]. Comparing this with the concrete domain theoretic LAPL-structure of [7], the completion process for LAPL-structures of [6], 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 [13].

Using adequacy of the interpretation of Lily$_{\text{strict}}$, we have shown consequences of parametricity for 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 $\psi$ 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, $\psi$ 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 $\psi$ 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*

$$\begin{tikzcd}
\sigma \arrow[r, multimap, "f_\sigma"] \arrow[d, multimap, "t"'] & \psi(\sigma) \arrow[d, multimap, "\psi(t)"] \\
\tau \arrow[r, multimap, "f_\tau"] & \psi(\tau)
\end{tikzcd}$$

*commutes up to internal equality.*

We define $\psi$ 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 $t$. □

$$
\begin{gathered}
\psi(\alpha) = \alpha \qquad \psi(\sigma \multimap \tau) = \psi(\sigma) \multimap \psi(\tau) \qquad \psi(\textstyle\prod \alpha.\, \sigma) = \prod \alpha.\, \psi(\sigma) \\
\psi(!\sigma) = !\psi(\sigma) \qquad \psi(I) = \textstyle\prod \alpha.\, \alpha \multimap \alpha \text{ (for } \alpha \text{ fresh variable)} \\
\psi(\sigma \otimes \tau) = \textstyle\prod \alpha.\, (\psi(\sigma) \multimap \psi(\tau) \multimap \alpha) \multimap \alpha \text{ (for } \alpha \text{ fresh variable)} \\
\psi(x) = x, \text{ for } x \text{ variable} \qquad \psi(\star) = \Lambda\alpha.\, \lambda^\circ x\colon \alpha.\, x \qquad \psi(Y) = Y \\
\psi(t\, u) = \psi(t)\, \psi(u) \qquad \psi(\lambda^\circ x\colon \sigma.\, t) = \lambda^\circ x\colon \psi(\sigma).\, \psi(t) \\
\psi(t \otimes u\colon \sigma \otimes \tau) = \Lambda\alpha.\, \lambda^\circ h\colon \psi(\sigma) \multimap \psi(\tau) \multimap \alpha.\, h\, \psi(t)\, \psi(u) \qquad \psi(!t) = !\psi(t) \\
\psi(\Lambda\alpha.\, t) = \Lambda\alpha.\, \psi(t) \qquad \psi(t(\sigma)) = \psi(t)(\psi(\sigma)) \\
\psi(\text{let } x \otimes y\colon \sigma \otimes \sigma' \text{ be } t \text{ in } u\colon \tau) = \psi(t)\, \psi(\tau)\, (\lambda^\circ x\colon \psi(\sigma).\, \lambda^\circ y\colon \psi(\sigma').\, \psi(u)) \\
\psi(\text{let } !x \text{ be } t \text{ in } u) = \text{let } !x \text{ be } \psi(t) \text{ in } \psi(u) \qquad \psi(\text{let } \star \text{ be } t \text{ in } u\colon \tau) = \psi(t)\, \psi(\tau)\, \psi(u)
\end{gathered}
$$

Figure 6: Inductive definition of $\psi$.

**Lemma A.4.** *Suppose $\sigma, \tau$ 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 *$\sigma, \tau$, 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*

$$
\begin{gathered}
\lambda^\circ x\colon \textstyle\prod \alpha.\, \alpha \multimap \alpha.\, x\, I\, \star\colon (\prod \alpha.\, \alpha \multimap \alpha) \multimap I \\
\lambda^\circ x\colon I.\, \Lambda\alpha.\, \lambda^\circ y\colon \alpha.\, \textit{let } \star \textit{ be } x \textit{ in } y\colon I \multimap (\textstyle\prod \alpha.\, \alpha \multimap \alpha)
\end{gathered}
$$

*can be shown to be each others inverses up to internal equality using parametricity.*

*Proof.* We show that $i_{\sigma,\tau}$ is natural in $\sigma, \tau$. 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) $\sigma$ of PILL$_Y$ as described in Figure 7. Basically $f_\sigma, g_\sigma$ 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.

$$f_\alpha = id_\alpha \qquad f_{\sigma \multimap \tau} = \lambda^\circ t \colon \sigma \multimap \tau.\, f_\tau \circ t \circ g_\sigma \qquad f_{!\sigma} = !f_\sigma$$

$$f_{\prod \alpha. \sigma} = \lambda^\circ t \colon (\textstyle\prod \alpha. \sigma).\, \Lambda \alpha.\, f_\sigma(t\,\alpha)$$

$$f_I = \lambda^\circ t \colon I.\, \Lambda \alpha.\, \lambda^\circ u \colon \alpha.\, \text{let } \star \text{ be } t \text{ in } u.$$

$$f_{\sigma \otimes \tau} = i_{\psi(\sigma), \psi(\tau)} \circ f_\sigma \otimes f_\tau$$

$$g_\alpha = id_\alpha \qquad g_{\sigma \multimap \tau} = \lambda^\circ t \colon \psi(\sigma) \multimap \psi(\tau).\, g_\tau \circ t \circ f_\sigma \qquad g_{!\sigma} = !g_\sigma$$

$$g_{\prod \alpha. \sigma} = \lambda^\circ t \colon (\textstyle\prod \alpha. \sigma).\, \Lambda \alpha.\, g_\sigma(t\,\alpha)$$

$$g_I = \lambda^\circ t \colon \textstyle\prod \alpha. \alpha \multimap \alpha.\, t\, I \star$$

$$g_{\sigma \otimes \tau} = g_\sigma \otimes g_\tau \circ i^{-1}_{\psi(\sigma), \psi(\tau)}$$

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 $f_\sigma, g_\sigma$ are each others inverses.*

*Proof.* Simple induction over the structure of $\sigma$. □

**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 $\sigma$. □

**Lemma A.7.** *$\psi$ is the identity on PILL$_Y \setminus \otimes$. If $\alpha \vdash \sigma(\alpha)$ is a type in PILL$_Y \setminus \otimes$ and $\tau$ is any type of PILL$_Y$, then $\psi(\sigma(\tau)) = \sigma(\psi(\tau))$.*

*Proof.* Easy induction on $\sigma$. □

**Lemma A.8.** *If $\Xi \vdash \sigma$ is a type in PILL$_Y \setminus \otimes$ then $f_\sigma$ is the identity.*

*Proof.* Easy induction on the structure of $\sigma$. □

The next few pages until Lemma A.12 we prove a series of lemmas describing the behavior of $f_\sigma$ with respect to reindexing. Basically what makes this difficult is that since $f_\alpha = id_\alpha$ if $\alpha$ 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 $\alpha$ occurs only negatively and $\beta$ 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 $\sigma$. 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 $\alpha$ occurs only negatively and $\beta$ 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*

$$\Xi \mid - \mid - \vdash \sigma[eq_\Xi, \langle f \rangle, \langle f \rangle] \equiv \langle \sigma(f^{-1}, f) \rangle$$

*Proof.* Suppose $f\colon \tau \multimap \tau'$. Consider first the two commutative diagrams

$$\begin{tikzcd} \tau \arrow[r, "id_\tau", multimap] \arrow[d, "f"', multimap] & \tau \arrow[d, "id_\tau", multimap] \\ \tau' \arrow[r, "f^{-1}", multimap] & \tau \end{tikzcd} \qquad \begin{tikzcd} \tau \arrow[r, "id_\tau", multimap] \arrow[d, "id_\tau"', multimap] & \tau \arrow[d, "f", multimap] \\ \tau \arrow[r, "f", multimap] & \tau'. \end{tikzcd}$$

These diagrams imply that

$$(id_\tau, f^{-1})\colon \langle f \rangle \multimap eq_\tau,$$
$$(id_\tau, f)\colon eq_\tau \multimap \langle f \rangle.$$

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 $\sigma$ in the case

$$\alpha' = \langle f \rangle, \quad \alpha = eq_\tau, \quad \beta = eq_\tau, \quad \beta' = \langle f \rangle$$

we get

$$\sigma(id_\tau, id_\tau)(\sigma[eq_\tau, eq_\tau] \multimap \sigma[\langle f \rangle, \langle f \rangle])\sigma(f^{-1}, f).$$

which using identity extension and functoriality of $\sigma$ gives

$$(id_{\sigma(\tau,\tau)}, \sigma(f^{-1}, f))\colon eq_{\sigma(\tau,\tau)} \multimap \sigma[\langle f \rangle, \langle f \rangle].$$

Since $\forall x\colon \sigma(\tau, \tau).\, x(eq_{\sigma(\tau,\tau)})x$ 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

$$\begin{tikzcd} \tau' \arrow[r, "f^{-1}", multimap] \arrow[d, "id_{\tau'}"', multimap] & \tau \arrow[d, "f", multimap] \\ \tau' \arrow[r, "id_{\tau'}", multimap] & \tau' \end{tikzcd} \qquad \begin{tikzcd} \tau \arrow[r, "f", multimap] \arrow[d, "f"', multimap] & \tau \arrow[d, "id_{\tau'}", multimap] \\ \tau' \arrow[r, "id_{\tau'}", multimap] & \tau'. \end{tikzcd}$$

Thus,

$$(f^{-1}, id_{\tau'})\colon eq_{\tau'} \multimap \langle f \rangle,$$
$$(f, id_{\tau'})\colon \langle f \rangle \multimap eq_{\tau'}.$$

So by parametricity

$$(\sigma(f^{-1}, f), id_{\tau'})\colon \sigma[\langle f \rangle, \langle f \rangle] \multimap eq_{\tau'}.$$

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 $\alpha$ occurs only negatively and $\beta$ 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 $\sigma$. The base cases of $\sigma$ 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 $\sigma$ as $\Xi, \alpha, \beta \vdash \sigma(\Xi, \alpha, \beta)$ where we have split the appearences of $\alpha$ in $\sigma$ into negative ($\alpha$'s) and positive ($\beta$'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 $\sigma$. 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 $\omega, \omega'$ 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*

$$\begin{tikzcd}\prod \alpha.\, \sigma \arrow[r, multimap, "app_{\tau}"] \arrow[d, multimap, "f_{\prod \alpha.\sigma}"'] & \sigma[\tau/\alpha] \arrow[d, multimap, "f_{\sigma[\tau/\alpha]}"] \\ \prod \alpha.\, \psi(\sigma) \arrow[r, multimap, "app_{\psi(\tau)}"] & \psi(\sigma[\tau/\alpha])\end{tikzcd}$$

*commutes, where* $app_{\tau}$ *is the map* $\lambda^{\circ}x\colon \prod \alpha.\, \sigma.\, x\ \tau$ *and* $app_{\psi[\tau/\alpha]}$ *defined likewise.*

*Proof.* Since

$$app_{\tau} \circ f_{\prod \alpha.\sigma}(x) = app_{\tau}(\Lambda\alpha.\, f_{\sigma}(x\ \alpha)) = f_{\sigma}[\tau/\alpha](x\ \tau)$$

we have $app_{\tau} \circ f_{\prod \alpha.\sigma} = f_{\sigma}[\tau/\alpha] \circ app_{\tau}$. Parametricity tells us that for all $x\colon \prod \alpha.\, \psi(\sigma)$,

$$\psi(\sigma)[eq_{\Xi}, \langle f_{\tau}\rangle](app_{\tau}(x), app_{\psi(\tau)}(x))$$

By Lemma A.11 we thus conclude

$$f_{\psi(\sigma)[\tau/\alpha]} \circ app_{\tau} = app_{\psi(\tau)}$$

Now, using Lemma A.12 we get

$$app_{\psi(\tau)} \circ f_{\prod \alpha.\sigma} = f_{\psi(\sigma)[\tau/\alpha]} \circ app_{\tau} \circ f_{\prod \alpha.\sigma} = f_{\psi(\sigma)[\tau/\alpha]} \circ f_{\sigma}[\tau/\alpha] \circ app_{\tau} = f_{\sigma[\tau/\alpha]} \circ app_{\tau}$$

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 $\psi$ 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

$$\psi(t' \otimes t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \\ \Lambda\alpha.\, \lambda^\circ h \colon \psi(\tau') \multimap \psi(\tau'') \multimap \alpha.\, h\ (\psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))\ (\psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})))$$

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** $t =$ let $z \otimes z'$ be $t'$ in $t''$:

Suppose in this case that $t' \colon \tau' \otimes \tau''$ and $t'' \colon \tau$. Now,

$$\psi(\text{let } z \otimes z' \text{ be } t' \text{ in } t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y})) = \\ \psi(t')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}))\ \psi(\tau)\ (\lambda^\circ x \colon \psi(\tau').\, \lambda^\circ y \colon \psi(\tau'').\, \psi(t'')(f_{\vec{\sigma}}(\vec{x}), f_{\vec{\sigma}'}(\vec{y}), x, y)) \tag{9}$$

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 $\tau$. 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, 8

[2] L. Birkedal and R. E. Møgelberg. Categorical models of Abadi-Plotkin's logic for parametricity. 2004. Submitted. 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] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Math. Struct. in Comp. Science*, 11, 1996. 2, 6.11, 7

[6] 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

[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. 3, 3, 3, 3, 4, 5, 5, 8

[8] 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

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

[10] 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

[11] 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

[12] G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, University of Oxford, 1986. 2.1

[13] 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, 7, 7.1, 7.1, 7.11, 8

[14] 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