Synthetic domain theory and models of Linear Abadi & Plotkin logic

Rasmus Ejlers Møgelberg, Lars Birkedal [1,2]

Department of Theoretical Computer Science, IT University Copenhagen, Denmark

Giuseppe Rosolini [3]

DISI, Università di Genova Genova, Italy

Abstract

In a recent article [4] 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 $\mathtt{PILL}_Y$, 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 [4,5].

Based on recent work by Simpson and Rosolini [22] 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.

Key words: Synthetic domain theory, parametric polymorphism, categorical semantics, domain theory

1 Introduction

It was first realized by Plotkin [16] that $\mathtt{PILL}_Y$, 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

[1] This work was partly supported by the Danish Technical Research Council under grant no.: 56–00–0309.
[2] Email: `mogel@itu.dk, birkedal@itu.dk`
[3] Email: `rosolini@disi.unige.it`

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 [16] Plotkin also sketched a logic for reasoning about parametricity for $\mathtt{PILL}_Y$ (the logic is a variant of Abadi & Plotkin's logic for parametricity [17]) and how to solve domain equations for $\mathtt{PILL}_Y$ 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 [4,5]. 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 the first authors PhD-thesis [12] a parametric completion process along the lines of [20] is presented constructing parametric LAPL-structures out of a large class of models of $\mathtt{PILL}_Y$.

In recent work Simpson and Rosolini [22] have constructed an interpretation (or rather a family of interpretations) of $\mathtt{Lily}_{\mathtt{strict}}$ a strict version of $\mathtt{Lily}$ [2] 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, 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 $\mathtt{Lily}_{\mathtt{strict}}$. In particular, they prove a version of the strictness theorem for $\mathtt{Lily}$ [2] for the new language $\mathtt{Lily}_{\mathtt{strict}}$. 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 $\mathtt{Lily}_{\mathtt{strict}}$ of [22]. We have three motivations for this work. First of all, we would like to show that the concept of parametric LAPL-structure is general enough to incorporate many different models. As mentioned we have already constructed a concrete domain-theoretic parametric LAPL-structure and shown how to construct parametric LAPL-structures from $\mathtt{PILL}_Y$-models using a parametric completion process. In a future paper we intend to construct a parametric LAPL-structure using operational semantics of $\mathtt{Lily}$, showing that the parametric reasoning used in [2] can be presented as reasoning in an LAPL-structure.

Our second motivation is that the interpretation presented in [22] is parametric and thus one should be able to solve recursive domain equations in it. Proving that the interpretation gives rise to an LAPL-structure provides a formal proof of this.

Our third motivation is that we can use the LAPL-structure and the adequacy of the interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ to show formally consequences of parametricity for $\mathtt{Lily}_{\mathtt{strict}}$. This builds upon the idea from [22] of giving denotational proofs of the theorems in [2], and extends it to prove properties not included in [2].

No prior knowledge of LAPL-structures or synthetic domain theory is needed for this paper. In Section 2 we sketch the definition of parametric LAPL-structure, and in Section 3 we introduce synthetic domain theory constructing a category of domains. The material in sections 2 and 3 are taken from [4,5] and [22] respectively, and so the original contributions of this paper start in Section 4. In Sections 4-6 we present the LAPL-structure. We first present a model of $\mathtt{PILL}_Y$ based on the category of domains, and 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 $\mathtt{Lily}_{\mathtt{strict}}$. In particular, we show how to solve recursive domain equations in $\mathtt{Lily}_{\mathtt{strict}}$. First, however, we present the language and sketch the results of [22].

For reasons of space, many proofs have been left out of this paper. For these proofs, see [15,12] [4] .

Acknowledgments. We thank Alex Simpson for helpful discussions. Also thanks to the anonymous referees for constructive suggestions.

2 LAPL-structures

The equational theory $\mathtt{PILL}_Y$ is a variant of $\mathtt{DILL}$ [1] extended with polymorphism and fixed points given by a fixed point combinator of type $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$, where in general we use $\sigma \to \tau$ as notation for $!\sigma \multimap \tau$.

We start off by sketching the notion of LAPL-structure as described in [4]. LAPL-structures model a variant of Abadi & Plotkin's logic for parametricity [17,16] designed for reasoning about $\mathtt{PILL}_Y$. Propositions in the logic exist in contexts of free type variables, free variables of $\mathtt{PILL}_Y$ and free relational variables. The free relational variables may be simply relations or admissible relations. Propositions are written as

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

The vector $\overline{\alpha}$ is a list of free type variables. We will not describe the logic in details, but only mention a few main points. The variables $\overline{x} \colon \overline{\sigma}$ are treated intuitionistic in the logic. We may reason about linear terms by for example

[4] The reader can find an online copy of [15] at `www.itu.dk/people/birkedal/papers`

using variables of type $\sigma \multimap \tau$, but the reasoning about the terms is purely intuitionistic.

The logic comes equipped with a notion of admissible relations, which is required to be closed under certain constructions. For example, equality relations (relating equal elements of some type) are required to be admissible, and admissible relations must be closed under reindexing along *linear* maps and universal quantification.

For any type $\overline{\alpha} \vdash \sigma(\overline{\alpha})\colon \mathsf{Type}$ with n free variables, and any n-vector of *admissible* relations $\overline{\rho}\colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')$, we can form an admissible relation $\sigma[\overline{\rho}]\colon \mathsf{AdmRel}(\sigma(\overline{\tau}), \sigma(\overline{\tau}'))$. This is called the relational interpretation of σ, and it is important for reasoning about parametricity. For example we can form the *identity extension schema* [19] as $\sigma[eq_{\overline{\alpha}}] \equiv eq_{\sigma(\overline{\alpha})}$, which we use as our definition of parametricity.

A *pre-LAPL*-structure is a schema of categories and functors

$$\begin{tikzcd}
 & & \mathbf{Prop} \arrow[d, "r"] \\
\mathbf{LinType} \arrow[r, bend right] \arrow[rrd, "p"'] & \mathbf{Type} \arrow[l, bend right] \arrow[r, hook, "I"] \arrow[rd] & \mathbf{Ctx} \arrow[d, "q"] \\
 & & \mathbf{Kind}
\end{tikzcd} \tag{1}$$

such that the diagram

$$\begin{tikzcd}
\mathbf{LinType} \arrow[rr, bend right, "G"'] \arrow[rr, phantom, "\perp"] \arrow[rd, "p"'] & & \mathbf{Type} \arrow[ll, bend right, "F"'] \arrow[ld] \\
 & \mathbf{Kind} &
\end{tikzcd} \tag{2}$$

is a model of $\mathtt{PILL}_Y$ [14] (a fibred version of models of $\mathtt{DILL}$ [1], with generic object $\Omega \in \mathbf{Kind}$ for p, simple products modeling polymorphism in p, and a term modeling the fixed point combinator).

We further require that the fibration q has fibred products and that I is a faithful map preserving fibred products. The pair of fibrations (r, q) is an indexed first-order logic fibration which has products and coproducts with respect to projections $\Xi \times \Omega \to \Xi$ in $\mathbf{Kind}$ [3], meaning that each fibre of r over an object Ξ in $\mathbf{Kind}$ is a first-order logic fibration with structure preserved under reindexing, and that the logic models quantification along the mentioned projections in $\mathbf{Kind}$.

Finally, there should exist a fibred functor U mapping pairs of types σ, τ in the same fibre of $\mathbf{LinType}$ to an object $U(\sigma, \tau)$ in $\mathbf{Ctx}$ acting as an object of all relations from $IG(\sigma)$ to $IG(\tau)$ in the logic of $\mathbf{Prop}$.

A notion of admissible relations for a pre-LAPL-structure is a subfunctor V of U closed under the constructions for admissible relations in the logic.

A pre-LAPL-structure models Abadi & Plotkin's logic for parametricity, except for the relational interpretation of types. The contexts of the logic are modeled in **Ctx** using U, V to model the sets of all relations and admissible relations between types respectively. The propositions of the logic are modeled in **Prop**.

From a pre-LAPL-structure with a notion of admissible relations one can define a `PILL` model (a $\mathtt{PILL}_Y$ model that does not necessarily model Y)

$$\begin{tikzcd}
\mathbf{LinAdmRel} \arrow[rr, bend right=10] \arrow[dr] & \perp & \mathbf{AdmRelations} \arrow[ll, bend right=10] \arrow[dl] \\
& \mathbf{AdmRelCtx} &
\end{tikzcd} \tag{3}$$

of admissible relations. There exists two maps of `PILL`-models ∂_0, ∂_1 from (3) to (2) basically mapping a relation to its domain and codomain respectively. An LAPL-structure is a pre-LAPL-structure with a notion of admissible relations and a map of `PILL`-models J from (2) to (3) such that

$$\partial_0 \circ J = \partial_1 \circ J = id.$$

The functor J models the relational interpretation of types. It enables us to talk about parametricity at all types in the model, not just the interpretations of types in *pure* $\mathtt{PILL}_Y$.

A *parametric LAPL-structure* is an LAPL-structure satisfying the identity extension schema in the internal logic. Moreover the extensionality schemes

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

should hold and the model should have very strong equality. The latter means that if two terms of $\mathtt{PILL}_Y$ are provably equal in the logic, then they are in fact equal in the model.

Parametric LAPL-structures are interesting because we can reason about the contained $\mathtt{PILL}_Y$-model using parametricity. In particular, we can solve a large class of domain equations in parametric LAPL-structures, and show that a large class of endo-functors have initial algebras and final coalgebras. We present these results in a restricted form, which is sufficient for the purposes of this paper.

We distinguish between *pure* $\mathtt{PILL}_Y$ and other $\mathtt{PILL}_Y$-theories with added type-constants and term-constants, such as the internal languages of models of $\mathtt{PILL}_Y$. A type of pure $\mathtt{PILL}_Y$ $\alpha \vdash \sigma(\alpha) \colon \mathsf{Type}$ in which α occurs only positively induces by standard constructions a functor, in the sense that there exists a term of type

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

preserving composition and identities. In the model σ induces an endofunctor

$[\![\sigma]\!]$ on $\mathbf{LinType}_1$, the fibre over the terminal object 1, which is the model of the closed types.

For each such type there exists a closed type $\mu\alpha.\,\sigma(\alpha)$, and closed terms,

$$\begin{aligned} \mathit{in}\colon\ & \sigma(\mu\alpha.\,\sigma(\alpha)) \multimap \mu\alpha.\,\sigma(\alpha), \\ \mathit{fold}\colon\ & \textstyle\prod\alpha.\,(\sigma(\alpha) \multimap \alpha) \to (\alpha \multimap \mu\alpha.\,\sigma(\alpha)) \end{aligned}$$

such that for any algebra $f\colon \sigma(\alpha) \multimap \alpha$, $\mathit{fold}\,\alpha\,!f$ is a map of algebras from *in* to f in the sense that

$$f \circ \sigma(\mathit{fold}\,\alpha\,!f) = (\mathit{fold}\,\alpha\,!f) \circ \mathit{in}.$$

Likewise there exists a closed type $\nu\alpha.\,\sigma(\alpha)$ and closed terms,

$$\begin{aligned} \mathit{out}\colon\ & \nu\alpha.\,\sigma(\alpha) \multimap \sigma(\nu\alpha.\,\sigma(\alpha)), \\ \mathit{unfold}\colon\ & \textstyle\prod\alpha.\,(\alpha \multimap \sigma(\alpha)) \to (\nu\alpha.\,\sigma(\alpha) \multimap \alpha) \end{aligned}$$

such that for any coalgebra $g\colon \alpha \multimap \sigma(\alpha)$, $\mathit{unfold}\,\alpha\,!g$ is a map of coalgebras from g to *out*.

Theorem 2.1 ([4]) *Suppose $\alpha \vdash \sigma(\alpha)$ is a type in pure $\mathtt{PILL}_Y$ in which α occurs only positively. In any parametric LAPL-structure in is interpreted as an initial algebra and out as a final coalgebra for $[\![\sigma]\!]\colon \mathbf{LinType}_1 \to \mathbf{LinType}_1$.*

The next theorem provides solutions to recursive domain equations.

Theorem 2.2 ([4]) *Suppose $\alpha \vdash \sigma(\alpha)\colon \mathsf{Type}$ is a type in pure $\mathtt{PILL}_Y$ (α may appear both positively and negatively). There exists a closed type $\mathit{rec}\,\alpha.\,\sigma(\alpha)$ in $\mathtt{PILL}_Y$ and terms*

$$\begin{aligned} f\colon\ & \mathit{rec}\,\alpha.\,\sigma(\alpha) \multimap \sigma(\mathit{rec}\,\alpha.\,\sigma(\alpha)), \\ g\colon\ & \sigma(\mathit{rec}\,\alpha.\,\sigma(\alpha)) \multimap \mathit{rec}\,\alpha.\,\sigma(\alpha) \end{aligned}$$

such that in any parametric LAPL-structure, f, g are interpreted as each others inverses.

We refer to [4,5] for further details.

3 Synthetic Domain Theory

The parametric models of $\mathtt{PILL}_Y$ which we shall produce are based on (a refined version of) ω-cpo's. Since these must model polymorphism, we shall consider an effective version of these and the simplest way to handle such structures is to view them within a realizability topos. Yet, that requires becoming acquainted with the logical/category-theoretic notion of realizability interpretation.

Fortunately, (all these possible choices of) effective ω-cpo's have been studied in great details and, following an intuition of Dana Scott, these were synthesized in an "elementary" theory, namely Synthetic Domain Theory.

The simplification is quite essential: domains are simply special sets, and continuous maps between them are all the set-theoretic maps. As one expects that every (continuous) endofunction has a fixed point, it is clear that the underlying set-theory is not classical.

As every new theory, SDT has lived through many reincarnations which have always tried to achieve an underlying level of elementarity in order to make it palatable for non-intuitionistically prone readers, see *e.g.* the original paper [7], [25] for some interesting achievements with the theory, [6] for quick review, and [9] and [18].

In this section we briefly recall the Synthetic Domain Theory introduced recently by Alex Simpson and the third author and described, though yet unpublished, in [22] as we shall only need some results of SDT. Since all the realizability models we are interested in satisfy the axioms for SDT, those results are true in such models, but we believe that the presentation via SDT will make them more intuitive (as it should be).

As we said, classical set theory is inconsistent with the basic view of SDT, so all must be developed in an intuitionistic theory of classes such as Algebraic Set Theory [8]—in fact, we shall want to be slightly more refined, but we postpone this issue to section 4. Hence, it is no longer the case that the powerset of a singleton $P\{*\}$—the set of truth values which will be written Ω as usual—consists of exactly two elements. This gives enough elbowroom to consider a subset $\Sigma \subseteq \Omega$ whose truth values should be thought of as of the form "P terminates", for P some program.

The first axiom requires that Σ is a *dominance*, see [21]:

- $\top \in \Omega$.
- If $p \in \Sigma, q \in \Omega$ and $p \supset (q \in \Sigma)$ then $p \wedge q \in \Sigma$.

For any set X and $p \in \Sigma$ denote by X^p the set of subsets e of X with at most one point such that $\exists x \in e$ is p. Following [22], a set is *pointed* when it is endowed with operations $r_p : X^p \to X$, for each p in Σ, such that

- $r_\top(\{x \mid \top\}) = x$ for all $x \in X$,
- $r_{p\wedge q}(e) = r_p(\{r_{p\wedge q}(e) \mid p\})$ for all $p, q \in \Sigma$, $e\colon X^{p\wedge q} \to X$.

The interested reader is referred to the basic reference [22] for a cleaner presentation.

The definition above is an extension of the notion of non-empty set (a set with a chosen element) which allows for more than just the classic two possibilities: $p = \top$ and $p = \bot$. Indeed, in those cases one computes from the definition that $r_\top$ is the obvious isomorphism and $r_\bot : 1 \to X$ chooses the element witnessing that X is non-empty.

The first evidence that the extension is appropriate is that a homomorphism $f\colon X \to Y$ between pointed sets—*i.e.* such that for all $p \in \Sigma$ and $e \in X^{\{*|p\}}$

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

—extends the usual notion of a strict map. As in [22], we shall use the same adjective *strict* for such a map.

It duly happens that the forgetful functor from the category of such algebras creates all limits. Moreover, for X and Y pointed sets, the set $X \multimap Y$ of homomorphisms from X to Y is a pointed subset of the product pointed set $\prod_{x\in X} Y = X \to Y$, see [22].

As usual, neither $\to$ nor $\multimap$ define a cartesian closed structure on the category of pointed sets and strict maps. As one may expect, $\to$ defines a cartesian closed structure on the category of pointed sets and all maps, and $\multimap$ is part of a symmetric monoidal closed structure.

Finally, the *free* pointed set on the set Z is $(LZ, (\mu_p)_{p\in\Sigma})$ where $LZ = \{\{z \mid p\} \mid z \in Z, p \in \Sigma\}$ and $\mu_p(E) = \{z \mid z \in e, e \in E\}$ for E in $(LZ)^p$. By freeness, a map $f\colon Z \to W$ induces a strict map $L(f)\colon L(Z) \to L(W)$ defined by $L(f)(e) = \{f(z) \mid z \in e\}$.

It is easy to check that $\Sigma \cong L1$ and so Σ bears a pointed structure.

Always following the approach to SDT of *loc.cit.*, one requires axiomatically the existence a class of sets (whose elements are called predomains) closed under appropriate constructions. As we already mentioned, such an axiomatic approach is supported by a host of models, each with its own peculiarities, but all verifying the following properties.

The class of predomains is a class of sets which is closed under isomorphic copies, set-indexed products and equalizers (so singletons are all predomains). Moreover,

- if A is a predomain, so is LA (hence $\Sigma = L1$ is a predomain)
- the set $\mathbb{N}$ of natural numbers is a predomain.

Lastly, there is a set $\mathbf{P}$ collecting all predomains "up to iso": for any predomain A, there exists $B \in \mathbf{P}$ such that $A \cong B$.

A *domain* is declared in [22] to be a pointed predomain. Denote by $\mathbf{Dom}_\perp$ the category of domains with strict maps and by $\mathbf{Dom}$ the category of domains with all maps. If $\mathbf{D}$ denotes the set of pointed structures on objects of $\mathbf{P}$

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

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 $\mathbf{Dom}_\perp$.

The crucial axiom in [22] about the class of (pre)domains is the following.

AXIOM For every domain A there is a function $\mathrm{fix}_A\colon (A \to A) \to A$ with the following properties

fixed points: for any $f\colon A \to A$, $f(\mathrm{fix}_A(f)) = \mathrm{fix}_A$

uniformity: for any $f\colon A_1 \to A_1$, $g\colon A_2 \to A_2$, $h\colon A_1 \multimap A_2$ such that

$$\begin{tikzcd} A_1 \arrow[r, "f"] \arrow[d, "h"', -o] & A_1 \arrow[d, "h", -o] \\ A_2 \arrow[r, "g"] & A_2 \end{tikzcd}$$

commutes, $h(\mathrm{fix}_{A_1}(f)) = \mathrm{fix}_{A_2}(g)$.

We end this section stating some properties of the categories $\mathbf{Dom}_\perp$ and $\mathbf{Dom}$ which we shall need in the following sections.

Lemma 3.1 (i) *The category* $\mathbf{Dom}_\perp$ *is complete.*

(ii) *The category* $\mathbf{Dom}$ *is cartesian closed.*

From the axioms above, it is easy to check that $(-) \multimap (=)$ defines a functor ${\mathbf{Dom}_\perp}^{\mathrm{op}} \times \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$, and that for any domain A the functor $A \multimap (-)$ preserves limits. Since the solution set condition is guaranteed by the existence of the set $\mathbf{D}$, one applies the Special Adjoint Functor Theorem to obtain that $A \multimap (-)$ has a left adjoint $A \otimes (-)$. So there is a natural isomorphisms

$$(B_1 \multimap (A \multimap B_2)) \cong ((A \otimes B_1) \multimap B_2).$$

One can read the isomorphism as a certain universal property of the domain $A \otimes B$: say that a map $f\colon A \times B \to C$ between domains is *strict in the first variable* if for all $p \in \Sigma, e \in A^{\{*|p\}}$, $y \in B$ $f(r_p(e), y) = r_p(\{f(x, y) \mid x \in e\})$. Likewise one can say what being strict in the second variable means. A map f is *bistrict* if it is strict in both variables. Bistrict maps are strict, but strict maps need not be bistrict (in general, projections in $\mathbf{Dom}_\perp$ are not bistrict). A bistrict map $A \times B \to C$ can be extended to a unique strict map $A \otimes B \multimap C$.

Lemma 3.2 *There is a functor* $(-) \otimes (=)\colon \mathbf{Dom}_\perp \times \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$ *and a domain* $I = \Sigma$ *giving* $\mathbf{Dom}_\perp$ *a symmetric monoidal closed structure. Moreover, the adjunction*

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

is symmetric monoidal.

A situation, as above, of a symmetric monoidal adjunction where the category on the right is cartesian closed and the category on the left is symmetric monoidal closed gives rise to a linear category structure on the category on the left. Hence $\mathbf{Dom}_\perp$ is symmetric monoidal closed, has a symmetric monoidal comonad L and bears a linear structure, see *e.g.* [14,10,11] for details.

Lemma 3.3 *The functor* $L\colon \mathbf{Dom}_\perp \to \mathbf{Dom}_\perp$ *extends the symmetric monoidal closed category on* $\mathbf{Dom}_\perp$ *to a linear structure.*

4 The domains fibration

In this section we construct a $\mathtt{PILL}_Y$-model based on the linear structure on the category $\mathbf{Dom}_\perp$. A first attempt at such a model would model types with n free variables as maps $f\colon (\mathbf{Dom}_\perp)_0^n \to (\mathbf{Dom}_\perp)_0$ where $(\mathbf{Dom}_\perp)_0$ is the class of domains. But to be able to handle polymorphism, we change this model slightly, such that types become functors $f\colon (\mathbf{Dom}_\perp)_{\mathtt{iso}}^n \to \mathbf{Dom}_\perp$ where $(\mathbf{Dom}_\perp)_{\mathtt{iso}}$ is the restriction of $\mathbf{Dom}_\perp$ to isomorphisms. The model described in this section will be turned into a parametric $\mathtt{PILL}_Y$-model in Section 5.

Some of the constructions of the present section cannot be carried out in the set theoretic setting used in Section 3, since they involve constructions on classes. In particular, since $(\mathbf{Dom}_\perp)_0$ is a class and not a set, the collection of all class maps $(\mathbf{Dom}_\perp)_0 \to (\mathbf{Dom}_\perp)_0$ is not a class, and so since a category has a class of objects, we cannot use this collection to construct a category.

For the concerned reader, we sketch how these issues may be resolved. As the given model of SDT, we will assume that we have a category of classes satisfying the axioms of Joyal and Moerdijk's algebraic set theory [8] as refined in [24] with the notion of classic structure on a regular category with a universe and a small natural numbers object. Given such a setting, the categories $\mathbf{Dom}$ and $\mathbf{Dom}_\perp$ mentioned above are internal categories in the regular category of classes while the collection of all internal functors $(\mathbf{Dom}_\perp)_0 \to (\mathbf{Dom}_\perp)_0$ is a class in the external sense, since it is a subclass of the class of morphisms of the category of classes. Thus the fibrations in (4) below are defined externally. The examples of realizability toposes mentioned in Section 3 still provide models as they embed into categories of classes as described in [25].

The reader should keep in mind that we really construct a family of parametric LAPL-structures. Since the LAPL-structure is constructed using SDT, we get a parametric LAPL-structure for each model of SDT.

We now begin the detailed description of the model. We define the fibration

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

by defining the base category to have as objects natural numbers and as morphisms from n to m functors $(\mathbf{Dom}_\perp)_{\mathtt{iso}}^n \to (\mathbf{Dom}_\perp)_{\mathtt{iso}}^m$. Objects in $\mathbf{DFam}((\mathbf{Dom}_\perp)_{\mathtt{iso}})$ over n are functors $(\mathbf{Dom}_\perp)_{\mathtt{iso}}^n \to \mathbf{Dom}_\perp$ and morphisms are natural transformations.

Lemma 4.1 *The fibration*

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

has a fibred linear structure plus fibred products.

Proof. Suppose $f, g\colon (\mathbf{Dom}_\perp)_{\mathtt{iso}}^n \to \mathbf{Dom}_\perp$ are objects of $\mathbf{DFam}((\mathbf{Dom}_\perp)_{\mathtt{iso}})_n$. We define $(f \multimap g)(\overline{D}) = f(\overline{D}) \multimap g(\overline{D})$ and if $\bar{i}\colon \overline{D} \multimap \overline{D}'$ is a vector of isomorphisms, then $(f \multimap g)(\bar{i})(h\colon f(\overline{D}) \multimap g(\overline{D})) = g(\bar{i}) \circ h \circ f(\bar{i}^{-1})$.

The rest of the structure is defined pointwise in the same manner. □

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. We sketch the proof. The main idea of the proof is due to [22]. 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

$$\mathrm{RK}_\pi(g)(\overline{A}) = \{x \in \textstyle\prod_{D\in\mathbf{D}} g(\overline{A}, D) \mid \forall D, D' \in \mathbf{D} \forall i\colon D \multimap D' \text{ iso}.\, g(\overline{A}, i)x_D = x_{D'}\}$$

Intuitively, this acts as a product of $g(\overline{A}, B)$ over all domains B, since we can define projections as follows. If B is a domain and $D \in \mathbf{D}$ is a domain such that there exists an isomorphism $i\colon D \multimap B$, we define

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

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

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

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

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 $(\mathbf{Dom}_\perp)_{\mathtt{iso}} \to \mathbf{Dom}_\perp$. The simple products are given by the right Kan extensions. □

Remark 4.4 The sketch of the proof of 4.2 shows how type specialization is interpreted.

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.

Since **Dom** is the coKleisli category for the lift comonad on $\mathbf{Dom}_\perp$, $\mathbf{DFam}(\mathbf{Dom}) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$ is the coKleisli fibration for the fibred comonad on $\mathbf{DFam}(\mathbf{Dom}_\perp) \to \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\}$, and so it is easy to prove that

$$\begin{tikzcd}
\mathbf{DFam}(\mathbf{Dom}_\perp) \arrow[dr] \arrow[rr, phantom, "\perp"] & & \mathbf{DFam}(\mathbf{Dom}) \arrow[ll, bend right] \arrow[dl] \\
& \{(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \mid n\} &
\end{tikzcd} \qquad (4)$$

is a `PILL` model.

The element $(\mathrm{fix}_D)_{D \in \mathbf{D}}$ models the fixed point combinator of type $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$, which leads us to the following proposition.

Proposition 4.5 *The diagram (4) is a* $\mathtt{PILL}_Y$*-model.*

5 The parametric fibration

In this section, we basically apply a parametric completion process as in [20,12] 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 (i.e. a pointed subset which is itself a domain) of $A \times B$ and we write $\mathbf{AdmRel}(A, B)$ for the set of admissible relations from A to B. This is the same notion of admissible relations used in [22]. We shall often write $R(x, y)$ for $(x, y) \in R$. Since the category of domains with strict maps is complete we can show the following lemma.

Lemma 5.1 *Admissible relations are closed under reindexing by strict maps and arbitrary intersections.*

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.

We have canonical reflexive graphs of functors:

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

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

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 a linear category structure and products. The maps of the reflexive graph*

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

preserve this 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

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

An admissible relation can be considered as a jointly monic span in the usual sense. We write $\bar{R}$ for the domain of the maps of the span in the following, in order not to confuse this with the relation. A first attempt at defining

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

would be the span obtained by taking tensors of maps:

$$\begin{tikzcd}
& \bar{R} \otimes \bar{S} \arrow[dl] \arrow[dr] & \\
A \otimes C & & B \otimes D.
\end{tikzcd}$$

However, we do not know that this is a jointly monic span, so instead we define $R \otimes S$ to be the intersection of all subdomains of $(A \otimes C) \times (B \otimes D)$ containing the image of this span.

The lift of a relation is obtained by lifting both maps in the span. □

We define the category **PDom** to have as objects natural numbers, and as morphisms from n to m pairs of functors making the diagram

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

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

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

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 $\overline{R}\colon \mathbf{AdmRel}(\overline{\alpha}, \overline{\beta})$,

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

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 $\overline{R}\colon \mathbf{AdmRel}(\overline{\alpha}, \overline{\beta})$, $(t_{\overline{\alpha}}, t_{\overline{\beta}})$ is a map of relations $f^r(\overline{R}) \to g^r(\overline{R})$.

Reindexing in the fibration $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ is by composition.

Lemma 5.3 *The fibration* $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ *has a fibred linear structure and fibred products.*

The structure is defined pointwise.

Lemma 5.4 *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 $(\prod f)^d(\overline{A})$ to be the subset of $\prod_{D \in \mathbf{D}} f^d(\overline{A}, D)$ of elements x satisfying

$$\forall D, D' \in \mathbf{D}. \forall R \in \mathbf{AdmRel}(D, D').\, f^r(eq_{\overline{A}}, R)(x_D, x_{D'})$$

where we write x_D for $\pi_D(x)$. We define the relational interpretation as

$$(\textstyle\prod f)^r(\overline{R}\colon \mathbf{AdmRel}(\overline{A}, \overline{B}))(x, y)$$

for $x \in (\prod f)^d(\overline{A}), y \in (\prod f)^d(\overline{B})$ iff

$$\forall D, D' \in \mathbf{D}. \forall R' \in \mathbf{AdmRel}(D, D') f^r(\overline{R}, R')(x_D, y_{D'}).$$

Since this is an intersection of admissible relations it is admissible by Lemma 5.1.□

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 from f^d to g^d, as seen as functors with codomain $\mathbf{Dom}$ instead of $\mathbf{Dom}_\perp$, i.e., the components of t are not required to be strict, such that for all vectors of relations $\overline{R} \colon \mathbf{AdmRel}(\overline{A}, \overline{B})$, the pair $(t_{\overline{A}}, t_{\overline{B}})$ is a map of relations $f^r(\overline{R}) \to g^r(\overline{R})$. Reindexing in the fibration $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{PDom}$ is given by composition.

Again $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{PDom}$ is the co-Kleisli fibration for the fibred monad on $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$, and so we can prove that we have a PILL-model:

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

Proposition 5.5 *The diagram (5) is a* $\mathtt{PILL}_Y$*-model.*

For the proof of Proposition 5.5 we just have to show that the fixed point combinator is modeled. To do this, we basically have to show that for any pair of domains A, B, any pair of maps $f \colon A \to A, g \colon B \to B$ and any admissible relation $R \colon \mathbf{AdmRel}(A, B)$, such that $R(x, y)$ implies $R(f(x), g(y))$, we have $R(\mathrm{fix}_A f, \mathrm{fix}_B(g))$. This can be done using uniformity of fix.

6 The LAPL-structure

In this section we show that the $\mathtt{PILL}_Y$-model (5) 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 $\overline{R} \colon \mathbf{AdmRel}(\overline{A}, \overline{B})$ we have $f^r(\overline{R}) \colon \mathbf{AdmRel}(f^d(\overline{A}), f^d(\overline{B}))$. 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 $\mathbf{Ctx}$ of (1) 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

$$\begin{tikzcd}
& & \mathbf{DFam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{PFam}(\mathbf{Dom}_\perp) \arrow[r, shift right] \arrow[rrd] & \mathbf{PFam}(\mathbf{Dom}) \arrow[l, shift right] \arrow[r] \arrow[rd] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
& & \mathbf{PDom}.
\end{tikzcd}$$
(6)

The category $\mathbf{DFam}(\mathbf{Set})$ is fibred over $\mathbf{PDom}$. Its fibre over n has as objects functors $(\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$, and reindexing along a morphism from m to n in $\mathbf{PDom}$ is by composition with the functor $((\mathbf{Dom}_\perp)_{\mathtt{iso}})^m \to ((\mathbf{Dom}_\perp)_{\mathtt{iso}})^n$. The category $\mathbf{DFam}(\mathrm{Sub}(\mathbf{Set}))$ is a fibred partial order over $\mathbf{DFam}(\mathbf{Set})$ and has as objects over $f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ subfunctors of f ordered by inclusion. The map $\mathbf{PFam}(\mathbf{Dom}) \to \mathbf{DFam}(\mathbf{Set})$ is given by the inclusion of $\mathbf{Dom}$ into $\mathbf{Set}$.

Lemma 6.1 *The fibration* $\mathbf{DFam}(\mathbf{Set}) \to \mathbf{PDom}$ *has fibred products and products in the base.*

Proof. The fibred products are given pointwise. □

Lemma 6.2 *The fibred functor*

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{Dom}) \arrow[r] \arrow[rd] & \mathbf{DFam}(\mathbf{Set}) \arrow[d] \\
& \mathbf{PDom}
\end{tikzcd}$$

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 $\mathrm{Sub}(\mathbf{Set}) \to \mathbf{Set}$.

What remains to be shown is that the composable fibration has simple products [3, Appendix A], which means that the logic models quantification along projections in $\mathbf{PDom}$.

To be precise, suppose $f\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ is an object of $\mathbf{DFam}(\mathbf{Set})_n$ and $h\colon (\mathbf{Dom}_\perp)^{n+1}_{\mathtt{iso}} \to \mathbf{Set}$ is a subfunctor of $\pi^* f = f \circ \pi$. We must define $(\prod h)\colon (\mathbf{Dom}_\perp)^n_{\mathtt{iso}} \to \mathbf{Set}$ a subfunctor of f and prove that for any other subfunctor g of f

$$\forall \overline{A}.\, g(\overline{A}) \subseteq (\textstyle\prod h)(\overline{A}) \quad \text{iff} \quad \forall \overline{A}, B.\, g(\overline{A}) \subseteq h(\overline{A}, B). \tag{7}$$

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

$$(\prod h)(\overline{A}) = \bigcap_{D \in \mathbf{D}} h(\overline{A}, D).$$

Clearly, the right to left implication of (7) holds. Suppose on the other hand that

$$\forall \overline{A}.\, g(\overline{A}) \subseteq (\prod h)(\overline{A}).$$

If $\overline{A}, B$ are domains, we must show that $g(\overline{A}) \subseteq h(\overline{A}, B)$. We know that there exists $D \in \mathbf{D}$ and isomorphism $i \colon B \cong D$. Since $h(\overline{A}, i) \colon h(\overline{A}, B) \to h(\overline{A}, D)$ is an isomorphism of subobjects of $f(\overline{A})$ we must have $h(\overline{A}, B) = h(\overline{A}, D)$, so since clearly $g(\overline{A}) \subseteq h(\overline{A}, D)$, also $g(\overline{A}) \subseteq h(\overline{A}, B)$ as desired. □

Lemma 6.4 *The diagram (6) defines a pre-LAPL-structure.*

Proof. All that is missing in this proof is the definition of the fibred functor U mapping a pair of types in the same fibre to an object of all relations between them. We define

$$U((f^r, f^d), (g^r, g^d))(\overline{A}) = \mathbf{Rel}(f^d(\overline{A}), g^d(\overline{A})).$$

□

Lemma 6.5 *The subfunctor of U given by*

$$V((f^r, f^d), (g^r, g^d))(\overline{A}) = \mathbf{AdmRel}(f^d(\overline{A}), g^d(\overline{A}))$$

defines a notion of admissible relations for the APL-structure (6).

As mentioned in the introduction, one of our aims with this paper is to show that the notion of parametric LAPL-structures is a general notion of parametric models. Lemma 6.5 is important in this respect, since it shows that the concrete notion of admissible relations of the SDT-model interprets the abstract notion of admissible relations presented in the definition of parametric LAPL-structures.

Theorem 6.6 *The pre-LAPL-structure (6) is an LAPL-structure.*

Basically, what needs to be proved in Theorem 6.6 is that all types in the model have a relational interpretation. Since a type in the model is a pair (f^d, f^r) where $f^d \colon (\mathbf{Dom}_\bot)^n_{\mathtt{iso}} \to \mathbf{Dom}_\bot$ and f^r maps n-vectors of admissible relations $\overline{R} \colon \mathbf{AdmRel}(\overline{A}, \overline{B})$ to relations $f^r(\overline{R}) \colon \mathbf{AdmRel}(f^d(\overline{A}), f^d(\overline{B}))$, the map f^r can be taken as a relational interpretation of (f^d, f^r). Notice that the reason this works, is that in the logic of the pre-LAPL-structure (6) relations on types $(f^d, f^r), (g^d, g^r)$ are families of relations on $f^d(\overline{A}), g^d(\overline{A})$ for $\overline{A} \in \mathbf{Dom}^n$, since the inclusion of $\mathbf{PFam}(\mathbf{Dom})$ into $\mathbf{DFam}(\mathbf{Set})$ forgets the f^r-part of a type.

Since the relational interpretation of types in LAPL-structures is given by a map of `PILL`-models, the proof of Theorem 6.6 also checks that the linear structure $(\multimap, \otimes, !)$ and the polymorphic structure of $\mathbf{PFam}(\mathbf{Dom}_\perp) \to \mathbf{PDom}$ agrees with the abstractly defined structure on $\mathbf{LinAdmRel} \to \mathbf{AdmRelCtx}$.

Theorem 6.7 *The LAPL-structure (6) is a parametric LAPL-structure, i.e., satisfies identity extension, extensionality and very strong equality.*

Proof. Identity extension holds basically because we have required that f^r preserves identities. Very strong equality follows from very strong equality in the subobject fibration over **Set**. Extensionality is a consequence of very strong equality. □

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

In [22] a language, which we shall call $\mathtt{Lily}_{\mathtt{strict}}$ is introduced. This language is a modification of `Lily` [2], where the function type $\sigma \multimap \tau$ is interpreted as strict rather than linear functions. The reason for using strictness rather than linearity is that it is more general, i.e., gives types to more terms, and that it is exactly what is needed for call-by-value and call-by-name to give the same notion of ground contextual equivalence. This intuitively also corresponds more directly to strict functions in domain theory, since these are the functions that diverge if their input does.

Simpson and Rosolini define an interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ into models of synthetic domain theory, and use this to prove that call-by-value and call-by-name give the same notion of contextual equivalence. This has been proved for `Lily` in [2] using operational methods, but Simpson and Rosolini give a different semantic proof. In [2] 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 (6) to prove more advanced parametricity results for $\mathtt{Lily}_{\mathtt{strict}}$.

The model of $\mathtt{PILL}_Y$ in (6) is based on the interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ given in [22]. In this section we show that the two interpretations of $\mathtt{PILL}_Y$ 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 $\mathtt{PILL}_Y$ into $\mathtt{Lily}_{\mathtt{strict}}$, and show that the interpretations agree up to this inclusion.

As mentioned earlier, the LAPL-structure we have constructed using synthetic domain theory is really a family of LAPL-structures, since we have one LAPL-structure for each model of synthetic domain theory.

In this section we will assume that we have chosen one such model which is also 1-consistent in the sense of [23,25]: any sentence of the form $\exists n \colon \mathbb{N}.\, \phi(n)$, for ϕ a primitive recursive predicate,—a Σ^0_1-sentence—is true in the model iff there exists (in the external sense) a natural number n such that $\phi(n)$ is true. This is, for example, the case for a realizability topos satisfying the strong completeness axiom [9] where one takes predomains to be the well-complete objects. The reason for this assumption is that adequacy (Theorem 7.1 below), will be proved in the internal language of the model; it will hold in the real world only under the assumption of 1-consistency (and precisely when 1-consistency holds), as explained also in Section 8 of [22]. This technique was introduced in [23,25].

We emphasize that the results about $\texttt{Lily}_{\texttt{strict}}$ (Theorems 7.6, 7.7) hold in general and independently of any model. Yet, to prove the results we need to refer to a model of SDT satisfying 1-consistency (which is known to exist).

7.1 The language $\texttt{Lily}_{\texttt{strict}}$

This subsection sums up some definitions and results from [22]. In particular we review the language $\texttt{Lily}_{\texttt{strict}}$ and with two operational semantics, a call-by-value and a call-by-name semantics. Each of these semantics give rise to a concept of contextual equivalence corresponding to observing termination at !- types.

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

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

where α ranges over an infinite set of type variables. Except for $\otimes, I$ these are exactly the types of $\texttt{PILL}_Y$. The notation $\vdash_\Xi \sigma \colon \mathsf{Type}$ means that σ is a well formed type with free type variables contained in Ξ.

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

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

where Γ is the context of free variables, i.e., an assignment of types to a finite set of variables usually written as $x_1 \colon \sigma_1, \dots, x_n \colon \sigma_n$ such that the free variables of t are contained in the domain of Γ, i.e., $\{x_1, \dots, x_n\}$. Ξ is a finite set of free type variables containing the free type variables of $\sigma_1, \dots, \sigma_n, \sigma$. The notation Ξ, α means $\Xi \cup \alpha$ *and* $\alpha \notin \Xi$. δ is a labeling of the variables in the domain of Γ, i.e., a map from $\{x_1, \dots, x_n\}$ to $\{0, 1\}$. Intuitively $\delta(x_i) = 1$ means that t is strict in x_i.

The notation $\vdash_\Xi \Gamma$ means that Γ is a well-formed context with free variables contained in Ξ.

Figure 1 recalls the term formation rules as defined in [22]. The notation $\Gamma \mid \delta, x :_i \sigma \vdash_\Xi t \colon \tau$ for $i = 0, 1$ is short for $\Gamma, x \colon \sigma \mid \delta[x \mapsto i] \vdash_\Xi t \colon \tau$, where $\delta[x \mapsto i]$ is the extension of δ to $\mathrm{dom}(\delta) \cup \{x\}$ such that $\delta(x) = i$. The notation $x :_{\_} \sigma$ means that either $x :_0 \sigma$ or $x :_1 \sigma$. For δ, δ' labellings of the same set of variables, the notation $\delta \vee \delta'$ is the labeling mapping $x \colon \mathrm{dom}(\delta)$ to $\max(\delta(x), \delta'(x))$. The constant zero labeling is denoted $\mathbf{0}$.

Figure 2 recalls the two operational semantics for $\texttt{Lily}_{\texttt{strict}}$ as defined in [22]. 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

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

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

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

$$\frac{\Gamma \mid \delta \vdash_{\Xi,\alpha} t \colon \sigma \qquad \vdash_\Xi \Gamma}{\Gamma \mid \delta \vdash_\Xi \Lambda\alpha.\, t \colon \prod \alpha.\, \sigma}$$

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

Fig. 1. Term formation rules for $\texttt{Lily}_{\texttt{strict}}$

terms of closed types of the form

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

$$\frac{}{\lambda x \colon \sigma.\, t \Downarrow \lambda x \colon \sigma.\, t}$$

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

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

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

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

Fig. 2. Operational semantics of $\texttt{Lily}_{\texttt{strict}}$

In Figure 2 the notation $t \Downarrow v$ is used in some rules. This means that each of these rules exist both in the definition of the $\Downarrow^n$ and the $\Downarrow^s$ semantics. The notation $t \Downarrow^n$ is short for $\exists v.\, t \Downarrow^n v$ and likewise for $t \Downarrow^s$.

The two operational semantics give rise to two concepts of operational equivalence, by observing termination at !-types. To be more precise, a ground σ-context is a term $x :_{\_} \sigma \vdash C \colon !\tau$ for some type τ, and if $t \colon \sigma$ the notation $C[t]$ denotes the substitution $C[t/x]$. For $t, t' \colon \sigma$ closed terms of closed types, t, t' are equivalent, written as $t \equiv^{\mathrm{s}}_{\mathrm{gnd}} t'$, if for all ground σ-contexts $C[-]$,

$C[t] \Downarrow^s$ iff $C[t'] \Downarrow^s$. Likewise $t \equiv^{\mathrm{n}}_{\mathrm{gnd}} t'$ if for all ground σ-contexts $C[-]$, $C[t] \Downarrow^n$ iff $C[t'] \Downarrow^n$. These two relations are clearly equivalence relations and congruences.

In [22] an interpretation of $\mathtt{Lily}_{\mathtt{strict}}$ is defined. We shall denote this interpretation $(\!|-|\!)$. As with the interpretation $[\![-]\!]$ of $\mathtt{PILL}_Y$ in the model (5) above, the interpretation has an interpretation of types as domains denoted $(\!|-|\!)^d$ and an interpretation of types as relations denoted $(\!|-|\!)^r$. Terms are interpreted as maps preserving relations. Since the interpretation almost coincides with the model defined here, we will not repeat the definition of the interpretation, but only state the results that we need.

Theorem 7.1 (Adequacy [22]) *Suppose $t, t' \colon \tau$ are closed terms of closed types. If $(\!|t|\!) = (\!|t'|\!)$ then $t \equiv^s_{gnd} t'$ and $t \equiv^n_{gnd} t'$.*

In [22], basically as a consequence of Theorem 7.1, it is proved that $\equiv^{\mathrm{s}}_{\mathrm{gnd}}$ and $\equiv^{\mathrm{n}}_{\mathrm{gnd}}$ coincide. Therefore we shall denote either of them by $\equiv_{\mathrm{gnd}}$.

7.2 Translating $\mathtt{PILL}_Y$ *into* $\mathtt{Lily}$

Consider the language $\mathtt{PILL}_Y \setminus \otimes$ obtained by removing the type-constructors $\otimes, I$ from $\mathtt{PILL}_Y$ and removing the corresponding term constructors such as the corresponding let-expressions, $\star$, and $\otimes$ of terms.

Now, the language $\mathtt{PILL}_Y \setminus \otimes$ has the same types as $\mathtt{Lily}_{\mathtt{strict}}$ and so the only real difference between the two languages is that $\multimap$ in $\mathtt{PILL}_Y \setminus \otimes$ denotes linear function space and in $\mathtt{Lily}_{\mathtt{strict}}$ it denotes strict function space. Since linear functions are strict, we can basically include $\mathtt{PILL}_Y \setminus \otimes$ into $\mathtt{Lily}_{\mathtt{strict}}$.

Theorem 7.2 *There exists an interpretation ϕ of $\mathtt{PILL}_Y \setminus \otimes$ into $\mathtt{Lily}_{\mathtt{strict}}$ such that for all closed terms t of $\mathtt{PILL}_Y$, $[\![t]\!] = (\!|\phi(t)|\!)$. This translation is the identity on types.*

The translation is functorial in the following sense: ϕ preserves identities and for $u \colon \sigma \multimap \tau, t \colon \tau \multimap \omega$ closed terms of closed types of $\mathtt{PILL}_Y \setminus \otimes$, $\phi(t \circ u) = \phi(t) \circ \phi(u)$.

The restriction of the translation to $\mathtt{PILL}_Y \setminus \otimes$ in Theorem 7.2 is not essential as the next proposition shows.

Proposition 7.3 *There exists a translation ψ of $\mathtt{PILL}_Y$ into $\mathtt{PILL}_Y \setminus \otimes$ such that for any parametric $\mathtt{PILL}_Y$-model X the diagram*

$$\begin{tikzcd}
\mathtt{PILL}_Y \arrow[rr, "\psi"] \arrow[dr, "{[\![-]\!]}"'] & & \mathtt{PILL}_Y \setminus \otimes \arrow[dl, "{[\![-]\!]}"] \\
& X &
\end{tikzcd}$$

commutes up to natural isomorphism. To be more precise, there exists a family of isomorphisms $f_\sigma \colon [\![\sigma]\!] \to [\![\psi(\sigma)]\!]$ indexed by closed types of $\mathtt{PILL}_Y$, such that *for each closed term $t\colon \sigma \multimap \tau$ of closed type, 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, the restriction of ψ to $\texttt{PILL}_Y \setminus \otimes$ is the identity, for $\alpha \vdash \sigma(\alpha)$ a type in $\texttt{PILL}_Y \setminus \otimes$, $\psi(\sigma(\tau)) = \sigma(\psi(\tau))$, and ψ is functorial in the sense of Theorem 7.2.

The core of the proof of Proposition 7.3 is the well known theorem that using parametricity,

$$\sigma \otimes \tau \cong \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha,$$
$$I \cong \prod \alpha.\, \alpha \multimap \alpha,$$

see [16,4]. Using these isomorphisms, one can translate any type σ of $\texttt{PILL}_Y$ into a type $\psi(\sigma)$ of $\texttt{PILL}_Y \setminus \otimes$ and construct an isomorphism $f_\sigma \colon \sigma \multimap \psi(\sigma)$. However, this is not the complete proof of the proposition, since we need to show that the smaller language can still express all the terms of the larger language. To be more precise we need to translate each term t of $\texttt{PILL}_Y$ possibly involving let-constructions not present in $\texttt{PILL}_Y \setminus \otimes$ into a term $\psi(t)$ of $\texttt{PILL}_Y \setminus \otimes$, and show that the collection f_σ defines a natural transformation as described in Proposition 7.3. Details can be found in [15,12].

Corollary 7.4 *There exists a translation of $\texttt{PILL}_Y$ into $\texttt{Lily}_{\texttt{strict}}$ which commutes with interpretation up to natural isomorphism. The translation is an extension of the translation of Theorem 7.2, and it is functorial.*

Proof. This follows from Theorem 7.2 and Proposition 7.3. □

Lemma 7.5 *The translation of $\texttt{PILL}_Y$ into $\texttt{Lily}_{\texttt{strict}}$ maps β, η- equivalent terms to ground contextually equivalent terms.*

Proof. Externally equal terms of $\texttt{PILL}_Y$ are interpreted as equal terms in the model. Since the translation commutes with interpretation into the model, by adequacy (Theorem 7.1), the translated terms are ground contextually equal. □

7.3 Consequences of parametricity for $\texttt{Lily}_{\texttt{strict}}$

We end this section by showing how to use Corollary 7.4, computational adequacy of the interpretation $(\!|-|\!)$ and the results of [4] to obtain consequences of parametricity for the language $\texttt{Lily}_{\texttt{strict}}$.

Consider the category whose objects are the closed types of $\texttt{Lily}_{\texttt{strict}}$ and whose morphisms from σ to τ are closed terms of type $\sigma \multimap \tau$ of $\texttt{Lily}_{\texttt{strict}}$ identified up to ground contextual equivalence. We call this category $\mathbf{Lily}_{\mathbf{strict}}$.

As always, type expressions $\vdash_\alpha \sigma(\alpha)$ in $\texttt{Lily}_{\texttt{strict}}$ for which α only appears positively in σ induce endofunctors on $\mathbf{Lily}_{\mathbf{strict}}$.

Theorem 7.6 *All functors* $\mathbf{Lily}_{\mathbf{strict}} \to \mathbf{Lily}_{\mathbf{strict}}$ *induced by types* $\sigma(\alpha)$ *in* $\texttt{Lily}_{\texttt{strict}}$ *have initial algebras and final coalgebras.*

Proof. We define the initial algebra by applying the translation of Corollary 7.4 to $\mathit{in}\colon \sigma(\mu\alpha.\,\sigma(\alpha)) \multimap \mu\alpha.\,\sigma(\alpha)$ of Theorem 2.1. To show that this defines a weak initial algebra, consider $\phi(\mathit{fold})$, that is, ϕ applied to the term that takes an algebra and produces a map from the initial algebra. Since

$$\Lambda\alpha.\,\lambda f\colon \sigma(\alpha) \multimap \alpha.\, f \circ \sigma(\mathit{fold}\,\alpha\,!f) = \Lambda\alpha.\,\lambda f\colon \sigma(\alpha) \multimap \alpha.\,(\mathit{fold}\,\alpha\,!f) \circ \mathit{in}$$

using Lemma 7.5 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 $\texttt{Lily}_{\texttt{strict}}$. 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 $h \equiv_{\mathrm{gnd}} g$.

The proof for final coalgebras is exactly the same. $\square$

Theorem 7.7 *For all types* $\alpha \vdash \sigma(\alpha)\colon \mathsf{Type}$ *of* $\texttt{Lily}_{\texttt{strict}}$, *there exists a closed type* τ *of* $\texttt{Lily}_{\texttt{strict}}$ *such that* τ *and* $\sigma(\tau)$ *are isomorphic as objects of* $\mathbf{Lily}_{\mathbf{strict}}$.

Proof. From Theorem 2.2 we obtain a type τ and isomorphisms $\tau \cong \sigma(\tau)$ in pure $\texttt{PILL}_Y$. Now, applying the translation of Corollary 7.4 to these isomorphisms we get a type τ' and morphisms $\sigma(\tau') \multimap \tau'$, $\tau' \multimap \sigma(\tau')$ definable in $\texttt{Lily}_{\texttt{strict}}$. By functoriality, the interpretations of both compositions of the two maps are identities. Thus, by adequacy, the two compositions are ground contextual equivalent to the identity, and thus τ' and $\sigma(\tau')$ are isomorphic in $\mathbf{Lily}_{\mathbf{strict}}$. $\square$

8 Conclusion

We have constructed an LAPL-structure based on the interpretation of $\texttt{Lily}_{\texttt{strict}}$ into models of synthetic domain theory presented in [22]. Comparing this with the concrete domain theoretic LAPL-structure of [14], the completion process for LAPL-structures of [13,12], and the LAPL-structure based on the operational semantics of $\texttt{Lily}$ [2] 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 [22].

Using adequacy of the interpretation of $\texttt{Lily}_{\texttt{strict}}$, we have shown consequences of parametricity for $\texttt{Lily}_{\texttt{strict}}$ up to ground contextual equivalence.

These consequences include encodings of inductive, coinductive and recursive types.

References

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

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

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

[4] 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.

[5] L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. In *Proceedings of the Twenty-first Conference on the Mathematical Foundations of Programming Semantics*, 2005. To appear.

[6] M. Fiore, A. Jung, E. Moggi, P. O'Hearn, J. Riecke, G. Rosolini, and I. Stark. Domains and denotational semantics: History, accomplishments and open problems. *Bulletin of the EATCS*, 59:227–256, jun 1996. Also published as Technical Report CSR-96-2, University of Birmingham School of Computer Science.

[7] J.M.E. Hyland. First steps in synthetic domain theory. In A. Carboni, M.C. Pedicchio, and G. Rosolini, editors, *Proceedings of the 1990 Como Category Theory Conference*, volume 1488 of *Lecture Notes in Mathematics*, pages 131–156. Springer, Berlin, 1991.

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

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

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

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

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

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

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

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

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

[17] 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.

[18] B. Reus and T. Streicher. General synthetic domain theory — a logical approach. *Mathematical Structures in Computer Science*, 1998.

[19] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983.

[20] 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.

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

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

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

[24] A. Simpson. Elementary axioms for categories of classes. In *14th Symposium on Logic in Computer Science (LICS'99)*, pages 77–87, Washington - Brussels - Tokyo, July 1999. IEEE.

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