Parametric Domain-theoretic Models of Polymorphic Intuitionistic / Linear Lambda Calculus

Lars Birkedal, Rasmus E. Møgelberg, Rasmus L. Petersen [1,2]

Department of Theoretical Computer Science, IT University Copenhagen, Denmark

Abstract

We present a formalization of a version of Abadi and Plotkin's logic for parametricity for a polymorphic dual intuitionistic / linear type theory with fixed points, and show, following Plotkin's suggestions, that it can be used to define a wide collection of types, including solutions to recursive domain equations. We further define a notion of parametric LAPL-structure and prove that it provides a sound and complete class of models for the logic, and conclude that such models have solutions for a wide class of recursive domain equations. Finally, we present a concrete parametric LAPL-structure based on suitable categories of partial equivalence relations over a universal model of the untyped lambda calculus.

Key words: Parametric polymorphism, Categorical semantics, domain theory

1 Introduction

In this paper we show how to define parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. The work is motivated by two different observations, due to Reynolds and Plotkin.

In 1983 Reynolds argued that parametric models of the second-order lambda calculus are very useful for modeling data abstraction in programming [24] (see also [18] for a recent textbook description). For real programming, one is of course not just interested in a strongly terminating calculus such as the second-order lambda calculus, but also in a language with full recursion. Thus in *loc. cit.* Reynolds also asked for a parametric *domain-theoretic* model

[1] This work was partly supported by the Danish Technical Research Council under grant no.: 56–00–0309.

[2] Email: `birkedal@itu.dk`, `mogel@itu.dk`, `rusmus@itu.dk`

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

of polymorphism [24]. Informally, what is meant [25] by this is a model of an extension of the polymorphic lambda calculus [23,9], with a polymorphic fixed-point operator $Y\colon \prod \alpha.\, (\alpha \to \alpha) \to \alpha$ such that

(i) types are modelled as domains, the sublanguage without polymorphism is modelled in the standard way and $Y\sigma$ is the least fixed-point operator for the domain σ;

(ii) the logical relations theorem (also known as the abstraction theorem) is satisfied when the logical relations are admissible, i.e., strict and closed under limits of chains;

(iii) every value in the domain representing some polymorphic type is parametric in the sense that it satisfies the logical relations theorem (even if it is not the interpretation of any expression of that type).

Of course, this informal description leaves room for different formalizations of the problem. Even so, it has proved to be a non-trivial problem. Unpublished work of Plotkin [21] indicates one way to solve the problem model-theoretically by using strict, admissible partial equivalence relations over a domain model of the untyped lambda calculus but, as far as we know, the details of this relationally parametric model have not been worked out in detail before. (We do that here.) In *loc. cit.* Plotkin also suggested that one should consider parametric domain-theoretic models not only of polymorphic lambda calculus but of polymorphic intuitionistic / linear lambda calculus, since this would give a way to distinguish, in the calculus, between strict and possibly non-strict continuous functions, and since some type constructions, e.g., coproducts, should not be modeled in a cartesian closed category with fixed points [10]. Indeed Plotkin argued that such a calculus could serve as a very powerful metalanguage for domain theory in which one could also encode recursive types, using parametricity. To prove such consequences of parametricity, Plotkin suggested to use a variant of Abadi and Plotkin's logic for parametricity [22] with fixed points.

Thus parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus are important both from a programming language perspective (for modeling data abstraction) and from a purely domain-theoretic perspective.

Recently, Pitts and coworkers [19,2] have presented a syntactic approach to Reynolds' challenge, where the notion of domain is essentially taken to be equivalence classes of terms modulo a particular notion of contextual equivalence derived from an operational semantics for a language called Lily, which is essentially polymorphic intuitionistic / linear lambda calculus endowed with an operational semantics.

In parallel with the work presented here, Rosolini and Simpson [27] have shown how to construct parametric domain-theoretic models using synthetic domain-theory in intuitionistic set-theory. Moreover, they have shown how to give a computationally adequate denotational semantics of Lily.

In the present paper we make the following contributions to the study of parametric domain-theoretic models of intuitionistic / linear lambda calculus:

- We present a formalization of Linear Abadi-Plotkin Logic with fixed points (LAPL). The term language, called PILL_Y for polymorphic intuitionistic / linear logic (the Y stands for the fixed point combinator), is a simple extension of Barber and Plotkin's calculus for dual intuitionistic / linear lambda calculus (DILL) with polymorphism and fixed points and the logic is an extension of Abadi-Plotkin's logic for parametricity with rules for forming admissible relations. The logic allows for intuitionistic reasoning over PILL_Y terms; i.e., the terms can be linear but the reasoning about terms is always done intuitionistically. In LAPL we can give detailed proofs of consequences of parametricity, including the solution of recursive domain equations; these results and proofs have not been presented formally in the literature before. We omit them here for reasons of space; they can be found in the accompanying technical report.
- We give a definition of a *parametric LAPL-structure*, which is a categorical notion of a parametric model of LAPL, with associated soundness and completeness theorems.
- We present a definition of a concrete parametric LAPL-structure based on suitable categories of partial equivalence relations over a universal model of the untyped lambda calculus, thus confirming the folklore idea that one should be able to get a parametric domain-theoretic model using partial equivalence relations over a universal model of the untyped lambda calculus.

We remark that one can see our notion of parametric LAPL-structure as a suitable categorical axiomatization of a good category of domains. In Axiomatic Domain Theory much of the earlier work has focused on axiomatizing the adjunction between the category of predomains and continuous functions and the category of predomains and partial continuous functions [5, Page 7] – here we axiomatize the adjunction between the category of domains and strict functions and the category of domains and all continuous functions and extend it with parametric polymorphism, which then suffices to model also recursive types.

In the technical development, we make use of a notion of admissible relations, which we axiomatize, since admissible may mean different things in different models. We believe our axiomatization is reasonable in that it accomodates several different kinds of models, such as the classical one described here and models based on synthetic domain theory [16].

The work presented here builds upon our previous work on categorical models of Abadi-Plotkin's logic for parametricity [3], which includes detailed proofs of consequences of parametricity for polymorphic lamdba calculus and also includes a description of a parametric completion process that given an internal model of polymorphic lambda calculus produces a parametric model. It is not necessary to be familiar with the details of [3] to read the present paper (except for Appendix A of [3], which contains some definitions and theory concerning composable fibrations), but, for readers unfamiliar with parametricity, it may be helpful to start with [3], since the proofs of consequences of parametricity given here are slightly more sophisticated than the ones in [3] because of the use of linearity.

In subsequent papers we intend to show how one can define a computationally adequate model of Lily and how to produce parametric LAPL-structures from Rosolini and Simpson's models based on intutionistic set theory [27] (this has been worked out at the time of writing [16]) and from Pitts and coworkers operational models [2] (we conjecture that this is possible, but have not checked all the details at the time of writing). As a corollary one then has that the encodings of recursive types mentionend in [27] and [2] really do work out (these properties were not formally proved in *loc. cit.*). We will also extend the parametric completion process of [3] to produce a parametric LAPL-structure given a model of polymorphic intuitionistic / linear lambda calculus, see [15].

For reasons of space, we have omitted many proofs from this paper. Further details can be found in the accompanying technical report [4], which includes proofs of all the properties stated herein. [3]

1.1 Outline

The remainder of this paper is organized as follows. In Section 2 we present LAPL, the logic for reasoning about parametricity over polymorphic intuitionistic / linear lambda calculus (PILL$_Y$). In Section 3 we state some of the main consequences of parametricity (see [4] for detailed proofs). In Section 4 we present our definition of an LAPL-structure and show soundness and completeness. In Section 5 we present our definition of a *parametric* LAPL-structure and prove that one may solve recursive domains equations in such. In Section 6 we present a concrete parametric LAPL-structure based on partial equivalence relations over a universal domain model. To make the model easier to understand, we first present a model of PILL$_Y$ (without parametricity) and then show how to make it into a parametric LAPL-structure. Moreover, as an example of how to calculate in the model, we characterize the definable natural numbers object.

2 Linear Abadi-Plotkin Logic

In this section we define a logic for reasoning about parametricity for Polymorphic Intuitionistic Linear Lambda calculus with fixed points (PILL$_Y$). The logic is based on Abadi and Plotkin's logic for parametricity [22] for the second-order lambda calculus and thus we refer to the logic as Linear Abadi-Plotkin Logic (LAPL).

[3] The reader can find an online copy of the technical report at `www.itu.dk/people/birkedal/papers`.

The logic for parametricity is basically a higher-order logic over PILL_Y. Expressions of the logic are formulas in contexts of variables of PILL_Y and relations among types of PILL_Y. Thus we start by defining PILL_Y.

2.1 $PILL_Y$

PILL_Y is essentially Barber and Plotkin's DILL [1] extended with polymorphism and a fixed point combinator.

Well-formed type expressions in PILL_Y are expressions of the form:

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

where σ is built using the syntax

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

and all the free type variables of σ appear on the left hand side of the turnstile. The list of α's is called the kind context, and is often denoted simply by Ξ or $\overline{\alpha}$. We have included $\otimes, I$ in the type system even though Plotkin's original idea was to define them using parametricity, since in models of the type system these constructions are always required to exist.

The terms of PILL_Y are of the form:

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

where the σ_i, σ'_j, and τ are well-formed types in the kind context Ξ. The list $\overline{x}$ is called the intuitionistic type context and is often denoted Γ, and the list $\overline{x'}$ is called the linear type context, often denoted Δ. No repetition of variable names is allowed in any of the contexts, but permutation akin to having an exchange rule is. Due to the nature of the formation rules, weakening and contraction can be derived for all but the linear context.

The grammar for terms is:

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

We use λ°, which bear some graphical resemblance to $\multimap$, to denote linear function abstraction. And we use s, t, $u \ldots$ to range over terms.

The formation rules given are the standard ones for DILL [1], extended to contexts with type variables, plus the standard rules for type abstraction and type application and the axiom

$$\Xi \mid \Gamma; - \vdash Y \colon \prod \alpha.\, !(!\alpha \multimap \alpha) \multimap \alpha.$$

$\Xi \mid \Gamma; \Delta$ is considered well-formed if for all types σ appearing in Γ and Δ, $\Xi \vdash \sigma \colon \mathsf{Type}$ is a well-formed type construction. Δ and Δ' are considered disjoint if the set of variables appearing in Δ is disjoint from the set of variables appearing in Δ'. We use $-$ to denote an empty context. As the types of variables in the let-constructions and function abstractions are often apparent from the context, these will just as often be omitted.

The *external equality* relation on PILL_Y terms is the least equivalence relation given by the rules of DILL [1] extended with an obvious rule for Y and β- and η-rules for polymorphic types. We encode ordinary lambda abstraction in the usual way by defining $\sigma \to \tau = !\sigma \multimap \tau$ and $\lambda x\colon \sigma.\, t = \lambda^\circ y\colon !\sigma.\, \mathrm{let}\ !x\ \mathrm{be}\ y\ \mathrm{in}\ t$, where y is a fresh variable. Using this notation the constant Y appears with the more familiar looking type $Y\colon \Pi\alpha.\, (\alpha \to \alpha) \to \alpha$.

2.2 The logic

As mentioned, expressions of LAPL live in contexts of variables of PILL_Y and relations among types of PILL_Y. The contexts look like this:

$$\begin{aligned}\Xi \mid \Gamma \mid\ & R_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \dots, R_n\colon \mathsf{Rel}(\tau_n, \tau_n'),\\ & S_1\colon \mathsf{AdmRel}(\omega_1, \omega_1'), \dots, S_m\colon \mathsf{AdmRel}(\omega_m, \omega_m')\end{aligned}$$

where $\Xi \mid \Gamma; -$ is a context of PILL_Y and the $\tau_i, \tau_i', \omega_i, \omega_i'$ are well-formed types in context Ξ, for all i. The list of R's and S's is called the relational context and is often denoted Θ. As for the other contexts we allow permutation, but no repetion of variables.

The concept of admissible relations is taken from domain theory. Intuitively admissible relations relate $\perp$ to $\perp$, and are closed under least upper bounds of chains.

It is important to note that there is no linear component Δ in the contexts — the point is that the logic only allows for *intuitionistic reasoning* about terms of PILL_Y, whereas PILL_Y terms can behave linearly.

Propositions in the logic are given by the syntax:

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

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

2.2.1 Definable relations

Definable relations, ranged over by ρ, are defined by rules below. Definable relations always have a domain and a codomain, just as terms always have types. The basic formation rules for definable relations are:

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

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

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

Notice that in the second rule we can only abstract *intuitionistic* variables to obtain definable relations. In the last rule, $\rho\colon \mathsf{AdmRel}(\sigma,\tau)$ is an admissible relation, to be discussed below. The rule says that the admissible relations constitute a subset of the definable relations.

An example of a definable relation is the graph relation of a function: $\langle f\rangle = (x\colon \sigma, y\colon \tau).\, fx =_\tau y$, for $f : \sigma \multimap \tau$. The equality relation eq_σ is defined as the graph of the identity map.

If $\rho\colon \mathsf{Rel}(\sigma,\tau)$ is a definable relation, and we are given terms of the right types, then we may form the proposition stating that the two terms are related by the definable relation:

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau) \quad \Xi \mid \Gamma; - \vdash t\colon \sigma, s\colon \tau}{\Xi \mid \Gamma \mid \Theta \vdash \rho(t,s)} \tag{1}$$

We shall also write $t\rho s$ for $\rho(t,s)$.

We introduce some shorthand notation for reindexing of relations. For $f\colon \sigma' \multimap \sigma, g\colon \tau' \multimap \tau$ and $\rho\colon \mathsf{Rel}(\sigma,\tau)$, we write $(f,g)^*\rho$ for the definable relation

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

2.2.2 Constructions on definable relations

In this subsection we present some constructions on definable relations, which will be used to give a relational interpretation of the types of PILL_Y. We define $\multimap$, $\forall$, !, and $\otimes$ on relations in such a way that they make **LinAdmRelations** — a category of relations introduced in Section 4 — into a linear category, i.e., a model of PILL_Y.

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

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

defined by

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

If

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

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

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

as

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

For $\rho\colon \mathsf{Rel}(\sigma, \tau)$, we seek to define a relation $!\rho\colon \mathsf{Rel}(!\sigma, !\tau)$. First we define for any type σ the proposition $(-)\downarrow$ on σ as

$$x \downarrow \equiv\ \exists f\colon \sigma \multimap I.\, f(x) =_I \star.$$

This definition is due to [27]. The intuition here is that since we have fixed points we may think of types as domains, and so $x\downarrow$ is thought of as $x \neq \bot$.

We further define the map $\epsilon\colon\, !\sigma \multimap \sigma$ as $\lambda^\circ x\colon !\sigma.$ let $!y$ be x in $y = \lambda x\colon \sigma.\, x$. We can now define

$$!\rho = (x\colon !\sigma, y\colon !\tau).\, x \downarrow \supset\subset y \downarrow \wedge (x \downarrow \supset \rho(\epsilon x, \epsilon y)).$$

Following the intuition of domains, ! is to be thought of as lifting, and ϵ the unit providing the unlifted version of an element. The formula then expresses that either both x and y are $\bot$ or they are both lifted elements whose unlifted versions are related.

Next we will define the tensor product of ρ and ρ'

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

for $\rho\colon \mathsf{Rel}(\sigma, \tau)$ and $\rho'\colon \mathsf{Rel}(\sigma', \tau')$. The basic requirement on the definition is that $\otimes$ should become a left adjoint to $\multimap$ in the category of relations **LinAdmRelations** to be introduced in Section 4. To give a concrete definition satisfying this requirement, we take a slightly long route. We first introduce the map

$$f : \sigma \otimes \tau \multimap \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$$

defined as

$$f\ x = \text{let } x' \otimes x''\colon \sigma \otimes \tau \text{ be } x \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\ x'\ x''.$$

Then we define

$$\rho \otimes \rho' = (f, f)^*(\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, (\rho \multimap \rho' \multimap R) \multimap R),$$

or, if we write it out, $\rho \otimes \rho' =$

$$\begin{aligned}&(x\colon \sigma \otimes \sigma', y\colon \tau \otimes \tau').\, \forall \alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta).\, \forall t\colon \sigma \multimap \tau \multimap \alpha, t'\colon \sigma' \multimap \tau' \multimap \beta.\\ &\quad (\rho \multimap \rho' \multimap R)(t, t') \supset R(\text{let } x' \otimes x'' \text{ be } x \text{ in } t\, x'\, x'', \text{let } y' \otimes y'' \text{ be } y \text{ in } t'\, y'\, y'').\end{aligned}$$

The reason for this seemingly convoluted definition, is that we will later prove, using parametricity, that $\sigma \otimes \tau$ is isomorphic to $\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha$, and we already have a relational interpretation of the latter. The idea of using this definition of $\otimes$ is due to Alex Simpson.

Similarly, to define the relation $I_{Rel}\colon \mathsf{AdmRel}(I, I)$ we define the map $f\colon I \multimap \prod \alpha.\, \alpha \multimap \alpha$ as $\lambda^\circ x\colon I.\,\text{let } \star \text{ be } x \text{ in } id$, where $id = \Lambda\alpha.\,\lambda^\circ x\colon \alpha.\, x$ and define

$$I_{Rel} = (f, f)^*(\forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, R \multimap R),$$

which, if we write it out, is

$$(x\colon I, y\colon I).\, \forall(\alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, \forall z\colon \alpha, w\colon \beta.$$
$$zRw \supset (\text{let } \star \text{ be } x \text{ in } z)R(\text{let } \star \text{ be } y \text{ in } w).$$

2.2.3 Admissible relations

The key notion used in Reynolds definition of relational parametricity [24] is the relational interpretation of a type. The relational interpretation of a type with n free variables is a function taking n relations and returning a new relation. However, we will not require that this function is defined on all vectors of relations, but only that it is defined on vectors of "admissible relations". On the other hand this function should also return admissible relations. Since "admissible" might mean different things in different settings, we axiomatize the concept of admissible relations.

The axioms for admissible relations are formulated in Figure 1. In the last of these rules $\rho \equiv \rho'$ is a shorthand for $\forall x, y.\, \rho(x, y) \supset\subset \rho'(x, y)$.

Proposition 2.1 *The admissible relations contains all graphs and are closed under the constructions of Section 2.2.2.*

Now, finally, we may give the last formation rule for definable relations:

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

We call $\sigma[\overline{\rho}]$ the *relational interpretation of the type* σ.

Remark 2.2 Observe that $\sigma[\overline{\rho}]$ is a syntactic construction and is not obtained by substitution as in [3]. Still $\sigma[\rho_1/\alpha_1, \ldots, \rho_n/\alpha_n]$ might be a more complete notation, but this quickly becomes overly verbose. In [22] $\sigma[\overline{\rho}]$ is to some extent defined inductively on the structure of σ, but in our case that is not enough, since we will need to form $\sigma[\overline{\rho}]$ for type constants (when using the internal language of a model of LAPL). We capture the inductive definitions in axioms.

2.2.4 Axioms and Rules

Having specified the language of LAPL, it is time to specify the axioms and inference rules. We have all the usual axioms and rules of predicate logic plus

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

$$\frac{}{\Xi \mid \Gamma \mid \Theta \vdash eq_\sigma\colon \mathsf{AdmRel}(\sigma,\sigma)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma; - \vdash t\colon \sigma' \multimap \sigma, u\colon \tau' \multimap \tau \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma', y\colon \tau').\, \rho(t\,x, u\,y)\colon \mathsf{AdmRel}(\sigma',\tau')}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho, \rho'\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \rho(x,y) \wedge \rho'(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (y\colon \tau, x\colon \sigma).\, \rho(x,y)\colon \mathsf{AdmRel}(\tau,\sigma)}$$

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

$$\frac{x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \top\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop} \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi \supset \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi, \alpha \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \vdash \tau\colon \mathsf{Type} \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall \alpha\colon \mathsf{Type}.\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma, z\colon \omega \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall z\colon \omega.\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{AdmRel}(\omega,\omega') \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall R\colon \mathsf{AdmRel}(\omega,\omega').\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\omega,\omega') \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau) \quad x,y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \forall R\colon \mathsf{Rel}(\omega,\omega').\, \rho(x,y)\colon \mathsf{AdmRel}(\sigma,\tau)}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau), \rho'\colon \mathsf{Rel}(\sigma,\tau) \qquad \Xi \mid \Gamma \mid \Theta \mid \top \vdash \rho \equiv \rho'}{\Xi \mid \Gamma \mid \Theta \vdash \rho'\colon \mathsf{AdmRel}(\sigma,\tau)}$$

Fig. 1. Rules for admissible relations

the axioms and rules specified below. There are obvious rules for substitution of terms, definable relations, and types for variables, relation variables, and type variables.

The rules for universal and existential quantification over types, terms, and relations, are standard.

As usual, external equality implies internal equality, and there are obvious rules expressing that internal equality is an equivalence relation.

Intuitively admissible relations should relate $\perp$ to $\perp$ and we need an axiom stating this. In general, we will use $(-)\downarrow$ as the test for $x \neq \bot$.

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(!\sigma, !\tau), \rho' \colon \mathsf{AdmRel}(!\sigma, !\tau) \quad x, y \notin \Gamma}{\begin{array}{l}\Xi \mid \Gamma \mid \Theta \mid \forall x \colon \sigma, y \colon \tau.\, \rho(!x, !y) \supset \rho'(!x, !y) \vdash \\ \qquad\qquad \forall x \colon !\sigma, y \colon !\tau.\, x \downarrow \supset\subset y \downarrow \supset (\rho(x,y) \supset \rho'(x,y))\end{array}} \tag{2}$$

We have rules concerning the interpretation of types as relations:

$$\frac{\overline{\alpha} \vdash \alpha_i \colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \alpha_i[\overline{\rho}] \equiv \rho_i}$$

$$\frac{\overline{\alpha} \vdash \sigma \multimap \sigma' \colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \multimap \sigma')[\overline{\rho}] \equiv (\sigma[\overline{\rho}] \multimap \sigma'[\overline{\rho}])}$$

$$\frac{\overline{\alpha} \vdash \sigma \otimes \sigma' \colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \otimes \sigma')[\overline{\rho}] \equiv (\sigma[\overline{\rho}] \otimes \sigma'[\overline{\rho}])}$$

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash I[\overline{\rho}] \equiv I_{Rel}}$$

$$\frac{\overline{\alpha} \vdash \prod \beta.\, \sigma(\overline{\alpha}, \beta) \colon \mathsf{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \top \vdash (\prod \beta.\, \sigma(\overline{\alpha}, \beta))[\overline{\rho}] \equiv \forall (\beta, \beta', R \colon \mathsf{AdmRel}(\beta, \beta')).\, \sigma[\overline{\rho}, R])}$$

$$\frac{\overline{\alpha} \vdash !\sigma \colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho} \colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (!\sigma)[\overline{\rho}] \equiv !(\sigma[\overline{\rho}])}$$

Here $\rho \equiv \rho'$ is shorthand for $\forall x, y.\, x\rho y \supset\subset x\rho' y$.

If the definable relation ρ is of the form $(x \colon \sigma, y \colon \tau).\, \phi(x, y)$, then $\rho(t, u)$ should be equivalent to ϕ with x, y substituted by t, u:

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

Finally, we need an axiom stating parametricity of the fixed point combinator:

$$\frac{}{\Xi \mid \Gamma; - \mid \Theta \vdash Y (\prod \alpha.\, (\alpha \to \alpha) \to \alpha) Y}$$

2.2.5 Extensionality and Identity Extension Schemas

The following two schemas are called **extensionality schemas**:

$$(\forall x \colon \sigma.\, t\, x =_\tau u\, x) \supset t =_{\sigma \multimap \tau} u$$

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

Lemma 2.3 *It is provable in the logic that*

$$\forall f, g \colon \sigma \to \tau.\, (\forall x \colon \sigma.\, f(!x) =_\tau g(!x)) \supset \forall x \colon !\sigma.\, f(x) =_\tau g(x).$$

In particular, extensionality implies

$$\forall f, g \colon \sigma \to \tau.\, (\forall x \colon \sigma.\, f(!x) =_\tau g(!x)) \supset f =_{\sigma\to\tau} g$$

The schema

$$- \mid - \mid - \vdash \forall \overline{\alpha} \colon \mathsf{Type}.\, \sigma[eq_{\overline{\alpha}}] \equiv eq_{\sigma(\overline{\alpha})}$$

is called the **identity extension schema**. Here σ ranges over all type expressions.

For any type $\beta, \alpha_1, \ldots, \alpha_n \vdash \sigma(\beta, \overline{\alpha})$ we can form the **parametricity schema**:

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

where, for readability, we have omitted $: \mathsf{Type}$ after β, β', and $eq_{\overline{\alpha}}$ is short notation for $eq_{\alpha_1}, \ldots, eq_{\alpha_n}$. It is easy to show that the identity extension schema implies the parametricity schema.

3 Proofs in LAPL

In LAPL one can make formal proofs of the definability of a wide collection of types, including recursive types, as suggested by Plotkin [21]. We have written out all the proofs in detail, but do not have space to include them here. They can be found in the accompanying technical report [4].

The main result is

Theorem 3.1 *Suppose $\alpha \vdash \sigma(\alpha) \colon \mathsf{Type}$ is a type in pure PILL$_Y$. There exists a closed type $rec\ \alpha.\, \sigma$ and a pair of terms $f \colon rec\ \alpha.\, \sigma \multimap \sigma(rec\ \alpha.\, \sigma)$, $g \colon \sigma(rec\ \alpha.\, \sigma) \multimap rec\ \alpha.\, \sigma$, such that the identity extension schema implies that $f \circ g =_{\sigma(rec\ \alpha.\sigma) \multimap \sigma(rec\ \alpha.\sigma)} id_{\sigma(rec\ \alpha.\sigma)}$ and $g \circ f =_{rec\ \alpha.\sigma \multimap rec\ \alpha.\sigma} id_{rec\ \alpha.\sigma}$.*

4 LAPL-structures

In this section we introduce the notion of an LAPL-structure. An LAPL-structure is a model of LAPL.

First, however, we call to mind what a model of PILL (PILL is PILL$_Y$ without the term Y) is and how PILL is interpreted in such a model (for a full description of models for PILL and interpretations in these, see e.g. [17,13,1,12]).

A model of PILL is a fibred symmetric monoidal adjunction

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

such that **LinType** is fibred symmetric monoidal closed; the tensor in **Type** is a fibred cartesian product; **Type** is equivalent to the category of finite products of free coalgebras; **Kind** is cartesian; p has a generic object and simple products with respect to projections forgetting Ω, where Ω is p of the generic object. See [13] for detailed explanation of this definition.

Recall that PILL is interpreted in such models as follows. A type σ is interpreted as an object $[\![\sigma]\!] \in \mathbf{LinType}$ and we interpret a term $\overline{\alpha} \mid \overline{x}\colon \overline{\sigma}; \overline{x}'\colon \overline{\sigma}' \vdash t\colon \tau$ as a morphism

$$![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_n]\!] \otimes [\![\sigma_1']\!] \otimes \ldots \otimes [\![\sigma_m']\!] \multimap [\![\tau]\!]$$

in **LinType**, where $! = FG$. Of course, $[\![!\sigma]\!] = ![\![\sigma]\!]$. Notice that we denote the morphisms in **LinType** by $\multimap$. Further recall that the intuitionistic part of the calculus, that is, the terms in the calculus with no free linear variables, can be interpreted in **Type**. For suppose we are given such a term $\Xi \mid \overline{x}\colon \overline{\sigma}; - \vdash t\colon \tau$. Then the interpretation of this term in **LinType** is

$$[\![\Xi \mid \overline{x}\colon \overline{\sigma}; - \vdash t\colon \tau]\!]\colon \otimes_i ![\![\Xi \mid \sigma_i]\!] \multimap [\![\Xi \mid \tau]\!].$$

Since $\otimes_i ![\![\Xi \mid \sigma_i]\!] \cong F(\prod_i G([\![\Xi \mid \sigma_i]\!]))$ (F can be proved to be strong) and $! = FG$, we have, using the adjunction $F \dashv G$, that such a term corresponds to

$$[\![\Xi \mid \overline{x}\colon \overline{\sigma}; - \vdash t]\!]_{\mathbf{Type}}\colon \textstyle\prod_i G([\![\sigma_i]\!]) \to G([\![\tau]\!])$$

in **Type**.

Finally, a model of PILL$_Y$ is a model of PILL, which models a fixed point operator

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

Definition 4.1 *A **pre-LAPL**-structure is*

(i) *a schema of categories and functors*

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

such that

- *the diagram*

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

is a model of PILL$_Y$.

- *q is a fibration with fibred finite products*
- *(r, q) is an indexed first-order logic fibration which has products and coproducts with respect to projections $\Xi \times \Omega \to \Xi$ in* **Kind** *[3], where Ω is p applied to the generic object of p. See Remark 4.2 below.*
- *I is a faithful product-preserving map of fibrations.*

(ii) *a contravariant morphism of fibrations:*

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

(iii) *a family of bijections*

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

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

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

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

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

Notice that Ψ is only defined on vertical morphisms.

Remark 4.2 We ask for the pair (r, q) to be an indexed first order logic fibration. This means that for each object Ξ in **Kind**, the restriction of r to the fibre over Ξ is a first order logic fibration, and the structure commutes with reindexing. We further require that (r, q) have simple products and coproducts, which means that the logic models quantification over types. Further note that, really, U is uniquely defined by the requirements on the rest of the structure so we will often refer to a pre-LAPL structure simply as the diagram in item 1.

By contravariance of the fibred functor U we mean that U is contravariant in each fibre.

We now explain how to interpret a subset of LAPL in a pre-LAPL structure. The subset of LAPL we consider at this stage is LAPL without admissible relations and without the relational interpretation of types.

We interpret the full contexts of the considered subset of LAPL in the category **Ctx** as follows. A context

$$\Xi \mid x_1 \colon \sigma_1, \ldots x_n \colon \sigma_n \mid R_1 \colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, R_m \colon \mathsf{Rel}(\tau_m, \tau_m')$$

is interpreted as

$$\prod_i IG(\llbracket\sigma_i\rrbracket) \times \prod_j U(\llbracket\tau_j\rrbracket, \llbracket\tau'_j\rrbracket),$$

where the interpretations of the types is the usual interpretation of types in $\mathbf{LinType} \to \mathbf{Kind}$.

For notational convenience we shall write $\llbracket\Xi \mid \Gamma \mid \Theta \vdash t\colon \tau\rrbracket$ for the interpretation of t in $\mathbf{Ctx}$, that is for $I(\llbracket\Xi \mid \Gamma; - \vdash t\colon \tau\rrbracket_{\mathbf{Type}}) \circ \pi$ (note the subscript $\mathbf{Type}$), where π is the projection $\pi\colon \llbracket\Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket\Xi \mid \Gamma \mid -\rrbracket$ in $\mathbf{Ctx}$.

The propositions in the logic are interpreted in $\mathbf{Prop}$ in the standard manner of categorical logic.

Definable relations with domain σ and codomain τ in contexts $\Xi \mid \Gamma \mid \Theta$ are interpreted as maps from $\llbracket\Xi \mid \Gamma \mid \Theta\rrbracket$ into $U(\llbracket\sigma\rrbracket, \llbracket\tau\rrbracket)$. The definable relation $\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma, \tau) \vdash R\colon \mathsf{Rel}(\sigma, \tau)$ is interpreted as the projection, and $\llbracket\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi\colon \mathsf{Rel}(\sigma, \tau)\rrbracket$

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

We now define the interpretation of $\rho(t, s)$, for a definable relation ρ and terms t, s of the right types. First, for $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma, \tau)$, we define

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

Next, if $\Xi \mid \Gamma \vdash t\colon \sigma, s\colon \tau$, then $\llbracket\Xi \mid \Gamma \mid \Theta \vdash \rho(t, s)\rrbracket$ equals

$$\langle\langle\pi, \langle\llbracket\Xi \mid \Gamma \mid \Theta \vdash t\rrbracket, \llbracket\Xi \mid \Gamma \mid \Theta \vdash s\rrbracket\rangle\rangle, \pi'\rangle^* \llbracket\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x, y)\rrbracket,$$

where π, π' are the projections $\pi\colon \llbracket\Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket\Xi \mid \Gamma\rrbracket$ and $\pi'\colon \llbracket\Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket\Xi \mid - \mid \Theta\rrbracket$.

To interpret admissible relations, we will assume that we are given a subfunctor V of U, i.e., a contravariant functor V with domain and codomain as U and a natural transformation $V \Rightarrow U$ whose components are all monomorphic. Thus, for all σ, τ, we can consider $V(\sigma, \tau)$ as a subobject of $U(\sigma, \tau)$. We think of $V(\sigma, \tau)$ as the subset of all admissible relations (since the isomorphism Ψ allows us to think of $U(\sigma, \tau)$ as the set of all definable relations).

We may interpret the logic containing admissible relations by interpreting $S\colon \mathsf{AdmRel}(\sigma, \tau)$ as $V(\llbracket\sigma\rrbracket, \llbracket\tau\rrbracket)$. Admissible relations are interpreted as maps into $V(\llbracket\sigma\rrbracket, \llbracket\tau\rrbracket)$. For this to make sense we need, of course, to make sure that the admissible relations in the model (namely the relations that factor through the object of admissible relations) in fact contain the relations that are admissible in the logic. We need to assume that of the functor V.

Definition 4.3 *A pre-LAPL structure together with a subfunctor V of U is said to* ***model admissible relations****, if V is closed under the rules of Figure 1 and Rule 2 holds.*

Given a pre-LAPL structure modelling admissible relations, we may define a fibration

$$\begin{array}{c}\mathbf{LinAdmRelations}\\ \downarrow\\ \mathbf{AdmRelCtx}\end{array},$$

which we think of as a model of admissible relations. We first define the category **AdmRelCtx** by the pullback

$$\begin{tikzcd}
\mathbf{AdmRelCtx} \arrow[r] \arrow[d, "\langle \partial_0, \partial_1 \rangle"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Ctx} \arrow[d] \\
\mathbf{Kind} \times \mathbf{Kind} \arrow[r, "\times"] & \mathbf{Kind}.
\end{tikzcd}$$

We write an object Θ in **AdmRelCtx** over (Ξ, Ξ') as $\Xi, \Xi' \mid \Theta$. The fiber of **LinAdmRelations** over an object $\Xi, \Xi' \mid \Theta$ is

objects triples (ϕ, σ, τ) where σ and τ are objects in **LinType** over Ξ and Ξ' respectively and ϕ is an admissible relation, i.e. a vertical map

$$\phi \colon \Theta \to V(\pi^*\sigma, \pi'^*\tau)$$

in **Ctx**.

morphisms A morphism $(\phi, \sigma, \tau) \to (\psi, \sigma', \tau')$ is a pair of morphisms

$$(t \colon \sigma \multimap \sigma', u \colon \tau \multimap \tau')$$

in $\mathbf{LinType}_{\Xi}$ and $\mathbf{LinType}_{\Xi'}$ respectively, such that

$$\Psi(\phi) \leq \Psi(V(t, u) \circ \psi)),$$

where we have left the inclusion of V into U implicit.

Reindexing with respect to vertical maps $\rho \colon \Theta \to \Theta'$ in **Ctx** is done by composition. Reindexing objects of **LinAdmRelations** with respect to lifts of maps in **Kind**$\times$**Kind** is done by reindexing in the fibration **Ctx** $\to$ **Kind**. Reindexing of morphisms in **LinAdmRelations** with respect to maps in **Kind**$\times$**Kind** is done by reindexing each map in **LinType** $\to$ **Kind**. This defines all reindexing since all maps in **AdmRelCtx** can be written as a vertical map followed by a cartesian map.

Remark 4.4 In the internal language, objects of **LinAdmRelations** are admissible relations

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

A vertical morphims in **LinAdmRelations** from $\rho \colon \mathsf{AdmRel}(\sigma, \tau)$ to $\rho' \colon \mathsf{AdmRel}(\sigma', \tau')$ is a pair of morphisms $f \colon \sigma \multimap \sigma'$, $g \colon \tau \multimap \tau'$ in **LinType** such that in the internal language the formula

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

holds.

There exist two canonical maps of fibrations:

$$\begin{tikzcd}\left(\begin{array}{c}\mathbf{LinAdmRelations}\\ \downarrow\\ \mathbf{AdmRelCtx}\end{array}\right) \arrow[r, shift left, "\partial_0"] \arrow[r, shift right, "\partial_1"'] & \left(\begin{array}{c}\mathbf{LinType}\\ \downarrow\\ \mathbf{Kind}\end{array}\right)\end{tikzcd}.$$

On the base category ∂_0, ∂_1 map an object $\Xi, \Xi' \mid \Theta$ to Ξ and Ξ' respectively. On the total category they map (ϕ, σ, τ) to σ and τ respectively. In words, ∂_0 and ∂_1 map a relation to its domain and codomain respectively.

Lemma 4.5 *If we define* **AdmRelations** *to be the category of finite products of coalgebras [13], we obtain a PILL-model*

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

and two maps of PILL-models ∂_0, ∂_1.

Proof. The proof proceeds essentially by verifying that the constructions on definable relations given in Section 2.2.2 make the fibration

$$\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$$

into a fibred linear category with generic object $V([\![\alpha \vdash \alpha\colon \mathsf{Type}]\!], [\![\alpha \vdash \alpha\colon \mathsf{Type}]\!])$. Notice that since V is closed under the rules of Figure 1, Proposition 2.1 tells us that the constructions on definable relations of Section 2.2.2 indeed do define operations on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$. □

Definition 4.6 *An* ***LAPL-structure*** *is a pre-LAPL-structure modeling admissible relations, together with a map of PILL-models J from*

$$\begin{tikzcd}\mathbf{LinType} \arrow[rr, bend right] \arrow[dr] & \perp & \mathbf{Type} \arrow[ll, bend right] \arrow[dl] \\ & \mathbf{Kind} & \end{tikzcd}$$

to

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

such that when restricting to the fibred linear categories, J together with ∂_0, ∂_1 is a reflexive graph, i.e., $\partial_0 \circ J = \partial_1 \circ J = id$.

In the following, we will often confuse J with the map of fibred linear categories from $\mathbf{LinType} \to \mathbf{Kind}$ to $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$.

We need to show how to interpret the rule

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

in LAPL-structures.

One can show that, since J preserves products in the base and generic objects, $J(\llbracket \overline{\alpha} \vdash \sigma(\overline{\alpha}) \rrbracket)$ is a relation from $\sigma(\overline{\alpha})$ to $\sigma(\overline{\beta})$ in context $\llbracket \overline{\alpha}; \overline{\beta} \mid \overline{R}\colon \mathsf{AdmRel}(\overline{\alpha}, \overline{\beta}) \rrbracket$. It thus makes sense to define $\llbracket \overline{\alpha}, \overline{\beta} \mid - \mid \overline{R}\colon \mathsf{AdmRel}(\overline{\alpha}, \overline{\beta}) \vdash \sigma[\overline{R}] \rrbracket$ to be $J(\llbracket \overline{\alpha} \mid \sigma(\overline{\alpha}) \rrbracket)$, so all we need to do now is to reindex this object. We reindex it to the right Kind context using $\langle \overline{\tau}, \overline{\tau}' \rangle : \llbracket \Xi \rrbracket \to \Omega^{2n}$, thus obtaining

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

For $\Xi \mid \Gamma \mid \Theta \vdash \overline{\rho}\colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}')$, we define

$$\begin{aligned} &\llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\overline{\rho}]\colon \mathsf{AdmRel}(\sigma(\overline{\tau}), \sigma(\overline{\tau}')) \rrbracket = \\ &\quad \llbracket \Xi \mid - \mid \overline{R}\colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}') \vdash \sigma[\overline{R}] \rrbracket \circ \llbracket \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho}\colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}') \rrbracket. \end{aligned}$$

where by $\llbracket \Xi \mid \Gamma \mid \Theta \vdash \overline{\rho}\colon \mathsf{AdmRel}(\overline{\tau}, \overline{\tau}') \rrbracket$ we mean the pairing

$$\langle \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho_1 \rrbracket, \ldots, \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho_n \rrbracket \rangle.$$

Soundness and completeness can then be proved by lengthy but fairly standard calculations.

Theorem 4.7 (Soundness) *The interpretation given above of LAPL in LAPL-structures is sound.*

Theorem 4.8 (Completeness) *There exists an LAPL-structure with the property that any formula of LAPL over pure PILL$_Y$ holds in this model iff it is provable in LAPL.*

5 Parametric LAPL-structures

Definition 5.1 *A **parametric LAPL-structure** is an LAPL-structure with very strong equality in which identity extension holds in the internal logic.*

Recall that very strong equality implies extensionality. We ask that identity extension and extensionality hold because this means that all the results from Section 3 apply to the internal logic of the LAPL-structure. Strong equality is used to conclude that properties proved in the internal logic also hold externally. This means that we can solve recursive domain equations in parametric LAPL-structures. In the following we will explain what this means exactly.

Suppose $\alpha \vdash \sigma$ is a type in pure PILL$_Y$. We may split the occurences of α in σ into positive and negative obtaining a type $\alpha, \beta \vdash \sigma(\alpha, \beta)$ such that α

occurs only negatively and β only positively. Such a type induces a functor which is contravariant in the first variable and covariant in the second, in the sense that there exists a term

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

preserving composition and identities (this is much as in [22]). Such a term induces a fibred functor

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

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

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

Theorem 5.2 *For parametric LAPL-structures every polymorphically strong fibred functor as above has a family of fixed points closed under reindexing.*

The fixed points are constructed as follows. In the special case of $\alpha \vdash \sigma$ where α occurs only positively in σ, σ can simply be considered a fibred functor

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

As a result of parametricity, such a functor has an initial algebra $\mu\alpha.\, \sigma = \prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ and a final coalgebra $\nu\alpha.\, \sigma = \prod \beta.\, (\prod \alpha.\, (!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta)) \multimap \beta$. Plotkin noticed that using the fixed point combinator Y one can show that initial algebras and final coalgebras coincide. This situation called is algebraic compactness and has been studied by Freyd [7,6,8], who has shown that in such categories general bifree solutions to recursive domain equations exist, and are given as the type τ constructed by

$$\begin{aligned} \tau''(\alpha) &= \mu\beta.\, \sigma(\alpha, \beta) \\ \tau' &= \nu\alpha.\, \sigma(\tau''(\alpha), \alpha) \\ \tau &= \tau''(\tau') \end{aligned}$$

The generalisation to polymorphically strong fibred functors means that we can solve recursive domain equations involving types modelled in the language which may not exist in pure PILL_Y. This could be interesting for example in the case of models with a type for real numbers.

6 A parametric domain-theoretic model of $\mathbf{PILL}_Y$

In this section we present a concrete parametric LAPL-structure based on partial equivalence relations over a universal domain model. To make it easier to understand the model, we first present a model of PILL_Y (without parametricity) and then show how to make it into a parametric LAPL-structure.

Let D be a pointed ω-chain-complete partial order such that we have $\Phi\colon D \to [D \to D]$ and $\Psi\colon [D \to D] \to D$, both Scott-continuous and satisfying $\Phi \circ \Psi = id_{[D\to D]}$, where $[D \to D]$ denotes the cpo of continuous functions from D to D. An *admissible partial equivalence relation on* D is a partial equivalence relation R on D that is *strict* (relates $\bot_D$ to $\bot_D$) and ω-chain complete.

We write $\mathbf{PER}(D)$ for the standard category of partial equivalence relations over D and $\mathbf{AP}(D)$ for the full subcategory of $\mathbf{PER}(D)$ on the admissible pers. The category $\mathbf{AP}(D)_\bot$ of admissible pers and strict continuous functions is the full-on-objects subcategory of $\mathbf{AP}(D)$ with only those morphisms $[f]\colon R \to S$ that have a strict continuous realizer.

Theorem 6.1 *The category $\mathbf{AP}(D)$ is a cartesian closed category, with ccc-structure defined as in $\mathbf{PER}(D)$, and $\mathbf{AP}(D)_\bot$ is a cartesian sub-category of $\mathbf{AP}(D)$. The category $\mathbf{AP}(D)_\bot$ is symmetric monoidal closed. There is an obvious forgetful functor $U\colon \mathbf{AP}(D)_\bot \to \mathbf{AP}(D)$, which has a left adjoint L. The adjunction $L \dashv U$ is monoidal.*

The L functor is of course a lifting functor. Define two fibrations

$$\mathbf{UFam}(\mathbf{AP}(D)) \to \mathbf{Set} \text{ and } \mathbf{UFam}(\mathbf{AP}(D)_\bot) \to \mathbf{Set}$$

of uniform families of admissible pers in the same way as one standardly defines the fibration of uniform families of per's (see, e.g., [11]). The functors L and U easily lift to fibred functors between these two fibrations:

$$\begin{tikzcd} \mathbf{UFam}(\mathbf{AP}(D)_\bot) \arrow[rr, bend right, "U"'] \arrow[dr, "q"'] & \perp & \mathbf{UFam}(\mathbf{AP}(D)) \arrow[ll, bend right, "L"'] \arrow[dl, "p"] \\ & \mathbf{Set}. & \end{tikzcd} \tag{3}$$

The set $\Omega = \mathrm{Obj}(\mathbf{AP}(D)_\bot) = \mathrm{Obj}(\mathbf{AP}(D))$ is a split generic object of the fibration q, and the fibration q has Ω-products satisfying the Beck-Chevalley condition given by intersections of admissible pers.

Theorem 6.2 *The diagram (3) constitutes a model of $PILL_Y$.*

Proof. Given the preceding results it only remains to verify that (1) the structure in the diagram models the polymorphic fixed point combinator and that (2) $\mathbf{UFam}(\mathbf{AP}(D))$ is equivalent to the category of products of free coalgebras of $\mathbf{UFam}(\mathbf{AP}(D))_\perp$. For (1), the required follows, as expected, by taking Y to be the equivalence class of the fixed-point operator on D, since the pers are strict and complete. For (2), observe that by general considerations it suffices to show that $\mathbf{UFam}(\mathbf{AP}(D))$ is equivalent to the coKleisli category of the adjunction $L \dashv U$, but this follows from the fact that U is a forgetful functor and since $\mathbf{UFam}(\mathbf{AP}(D))$ has products. $\square$

6.1 The LAPL-structure

In this section, we introduce a parametric version of the thus far constructed model. It is essentially obtained through a parametric completion process such as the one described in [26,3] for internal λ_2 models. We end up with the following diagram of categories and functors:

$$
\begin{tikzcd}
 & & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[r, bend right, "U"'] \arrow[dr] & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[l, bend right, "L"'] \arrow[r, hook, "I"] \arrow[d] & \mathbf{Fam}(\mathbf{Set}) \arrow[dl] \\
 & \mathbf{PAP}(D) &
\end{tikzcd}
\tag{4}
$$

which, together with a functor V, constitutes a parametric LAPL structure.

The two leftmost fibrations are defined from the two fibrations in (3) in essentially the same manner as the fibration in [11, Proposition 8.6.3] is defined from the standard fibration of uniform families of pers. The objects of $\mathbf{PAP}(D)$ are natural numbers. Objects in fibres are now families of admissible pers together with families of *admissible* relations, which are taken to be regular subobjects in $\mathbf{AP}(D)_\perp$. To make it a bit more precise, first recall that a regular subobject in $\mathbf{AP}(D)_\perp$ of an object R in $\mathbf{AP}(D)_\perp$ is a set of equivalence classes of R such that, when considered as a per, it forms an admissible per. A type in $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$ with n free type variables is then a pair (f^p, f^r) where

- f^p is a map of objects $(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n \to \mathrm{Obj}(\mathbf{AP}(D)_\perp)$
- f^r is a map, that to two vectors of objects of $\mathbf{AP}(D)_\perp$ associates maps of subobjects $f^r \in$

  $$\Pi_{\overline{R},\overline{S}\in(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n}\left(\Pi_{j\in\{1,\dots,n\}}\,\mathrm{RegSub}(R_j \times S_j) \to \mathrm{RegSub}(f^p(\overline{R}) \times f^p(\overline{S}))\right)$$

  satisfying that it maps equality relations to equality relations, that is:

  $$\forall \overline{R} \in (\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n.\; f^r(\overline{R},\overline{R})(\overline{Eq_{R_j}}) = Eq_{f^p(\overline{R})}.$$

Over n, the fibration $\mathbf{Fam}(\mathbf{Set})$ contains $(\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n$-indexed families of sets and functions between such. The functor I takes a family of admissible pers and relations (f^p, f^r) to the family of sets of equivalence classes of the admissible pers, i.e., $I(f^p, f^r)\colon (\mathrm{Obj}(\mathbf{AP}(D)_\perp))^n \to \mathbf{Set}$ maps $\overline{R}$ to the set of equivalence classes of $f^p(R)$. Using $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set})$ to model families of predicates corresponds to the intuitive idea that predicates on types, i.e., on admissible pers and relations, are simply modelled by subsets of equivalence classes of the admissible pers. The V functor takes admissible pers and relations to the set of subsets of equivalence classes of the admissible pers that correspond to regular subobjects in $\mathbf{AP}(D)_\perp$. Long calculations then verify:

Theorem 6.3 *The diagram (4) together with the functor V is a parametric LAPL-structure.*

Observe that the fibre $\mathbf{PFam}(\mathbf{AP}(D)_\perp)_0$ over the terminal object in $\mathbf{PAP}(D)$, which corresponds to closed types and terms, is equivalent to the fibre $\mathbf{UFam}(\mathbf{AP}(D)_\perp)$ (likewise for $\mathbf{PFam}(\mathbf{AP}(D))$). Thus the closed types in the parametric model are modeled in the same way as in the simple model in (3).

6.2 Example of Calculations in the Model

We consider the type

$$\mathrm{Nat} = [\![\textstyle\prod \alpha.\, (\alpha \multimap \alpha) \to \alpha \multimap \alpha]\!].$$

Arguing as for Theorem 5.2 we can show that Nat is a natural numbers object in the fibre of $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PAP}(D)$ over the terminal object. We present a concrete description of Nat. By definition $d(\mathrm{Nat})d'$ iff for all admissible pers R, S and admissible relations $A \subset R \times S$, all $f, g\colon (A \multimap A)$ and all $(x, y) \in A$

$$(d\ f\ x, d'\ g\ y) \in A.$$

The domain of Nat contains the elements $\perp = \lambda f\, \lambda x.\, \perp$ and $\underline{n} = \lambda f.\, \lambda x.\, f^n(x)$, in particular $\underline{0} = \lambda f\, \lambda x.\, x$.

Lemma 6.4 *Suppose $\underline{n} \leq \underline{m}$. Then $n = m$.*

Proof. Consider the two functions $f, g\colon D \to D$ given by $f(d) = \langle d, \iota\rangle$, and g being the first projection. Both are continuous and since $g \circ f = id$, f is injective. Define the sequence of elements $x_n = f^n(\perp)$. This sequence is strictly increasing. Now, if $\underline{n} \leq \underline{m}$ then

$$x_n = \underline{n}\, f\, \perp \leq \underline{m}\, f\, \perp = x_m$$

so $n \leq m$. Further,

$$x_{m-n} = \underline{n}\, g\, x_m \leq \underline{m}\, g\, x_m = \perp$$

so $m = n$. □

Corollary 6.5 *The per*

$$\{\{\bot\}\} \cup \{\{\underline{n}\} \mid n\}$$

is admissible.

Proposition 6.6 *Suppose $d(\mathrm{Nat})d$. Then either $d = \bot$ or $d = \underline{n}$.*

Proof. Consider the discrete admissible per $D = \{\{d\} \mid d \in D\}$. Then given f, x consider the admissible relation $R\colon \mathsf{AdmRel}(\mathrm{Nat}, D)$ given by

$$\bot R \bot, \qquad \forall n.\, \underline{n} R (f^n(x)).$$

R is admissible, simply because it contains no interesting increasing chains. Clearly $(succ, f)\colon R \multimap R$, so

$$R(d\; succ\; 0, d\; f\; x),$$

i.e., if $d\; succ\; 0 = \bot$, then $d\; f\; x = \bot$ for all f, x, and if $d\; succ\; 0 = \underline{n}$ for some n, then $d\; f\; x = f^n(x)$, for all f, x. In both cases, $d\; succ\; 0 = d$. From the definition of R we see that there are no other possibilities for $d\; succ\; 0$. □

Proposition 6.7 *Suppose $d(\mathrm{Nat})d'$, then $d = d'$.*

Proof. We saw in the above proof that $d = d\; succ\; 0$. By considering the admissible relation $R\colon \mathsf{AdmRel}(\mathrm{Nat}, \mathrm{Nat})$ given by

$$\bot R \bot, \qquad \forall n.\, \underline{n} R \underline{n}$$

we conclude that $d\; succ\; 0 = d'\; succ\; 0$, and so $d = d'$. □

Acknowledgments

We gratefully acknowledge discussions with Milly Maietti, Gordon Plotkin, John Reynolds, Pino Rosolini and Alex Simpson.

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, 2005.

[5] M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps.* Distinguished Dissertations in Computer Science. Cambridge University Press, 1996.

[6] P.J. Freyd. Algebraically complete categories. In A. Carboni, M. C. Pedicchio, and G. Rosolini, editors, *Category Theory. Proceedings, Como 1990*, volume 1488 of *Lecture Notes in Mathematics*, pages 95–104. Springer-Verlag, 1990.

[7] P.J. Freyd. Recursive types reduced to inductive types. In *Proceedings of the fifth IEEE Conference on Logic in Computer Science*, pages 498–507, 1990.

[8] P.J. Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P.T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1991.

[9] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.* Thèse d'Etat, Université Paris VII, 1972.

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

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

[12] 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.

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

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

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

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

[17] R.L. Petersen and J. Thamsborg. Polymorphism and linearity all in one PILL. Student Project, 2003.

[18] B.C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

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

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

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

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

[23] J.C. Reynolds. Towards a theory of type structure. In *Colloquium sur La Programmation*, volume 19 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1974.

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

[25] J.C. Reynolds. Private communication, June 2000.

[26] 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.

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