

A  
Concrete  
Parametric  
Domain-theoretic  
Model of PILL $\gamma$

## Introduction

This is a qualification report concluding the first half of my Ph.D-enrollment at the IT-University of Copenhagen. As such it is supposed to document some of my work during that period as well as suggest possibilities for future work. To document my work I have included a technical report providing the main result of my work so far as well as an article presenting this result in a theoretical context. In addition to the herein presented material I am coauthor on the technical report “Categorical models of PILL”, which can be seen as a preliminary discussion to the article presented here, but my part in that work is neither large nor isolated enough to include here. Future work is discussed in a later section.

## Outline

This qualification report consists of three parts:

1. A conference article on which I am a co-author, that was presented at MFPS XXI. This article tries to tell at least three different stories at once, one of them being that a concrete model can be obtained in the way people believe it can. I have written out the details proving this belief to be sound.
2. An excerpt from the technical report behind the conference article which provides all the technical details. The excerpt includes only the chapter on concrete models, as this is what I have written.
3. As the technical report is aimed researchers and other people experienced in the field of domain theory, even this document skip over some details of the more fundamental calculations. Therefor a document, aimed at students, containing many of these has been included. Since this level of detail mostly make sense in the first part (very few students know of LAPL-structures yet) this last text only deals with the first part of the technical report. This also serves to shorten this qualification report.

## Future work

I here list a few possible directions for future work. I have ordered them such that the first listed are the most concrete and accessible, with which I might start right away, and the later listed require more conceptual development and are more general and speculative.

### LILY as an LAPL structure

In “Operational Properties of LILY, a Polymorphic Linear Lambda Calculus with Recursion” Pitts and coworkers presented a version of PILL<sub>Y</sub> called LILY that has an operational semantics. This allows a definition of ground contextual equivalence, which they manage to characterize through a concept called

$\top\top$ -closed relations. This characterization is particularly neat, from our point of view, as it includes constructions on these relations corresponding to the different constructions on types in the language. Moreover they obtain a parametricity result (albeit only for closed terms of closed types). Thus  $\top\top$ -closed relations seem ideal as a notion of admissible relations in an LAPL-structure based on terms identified by ground contextual equivalence. The existence of such an LAPL-structure would prove at least two points: Firstly, the parametricity result would be extended to open terms and types and encompass all the consequences of parametricity treated in our article. Secondly, the notion of LAPL-model is rich enough to contain this model (which is not the image of a completion process). This is already ongoing work.

### Other types of effects

$\text{PILL}_Y$  can be seen as linear logic with the added effect  $!$ , providing the possibility of non-linear terms. In order to move closer towards real life programming languages, one would need to look at other types of effect. It is known, that most effects can be encoded through continuations, and Andrzej Filinski has shown that this is reflected at the level of monads. Thus an obvious step would be to look at a language with continuations.

Another step, more directly aimed at reasoning about pointers, would be to consider the connection to separation logic. A recent publication, “Semantics of Separation-logic Typing and Higher-order Frame Rules”, of Birkedal Torp-Smith and Yang featured a model of promising resemblance to what is presented here.

Then one could move on to other languages, with sub typing in the type system of  $F_\omega$ .

### Study the concrete model

Since we are now provided with a concrete parametric model, we might as well use it to answer as many questions as it can. During the ongoing work mentioned above we have managed to add tensor to the language  $\text{LILY}$  (and to its operational semantics) effectively turning it into  $\text{PILL}_Y$ . We thus have an operational semantics of  $\text{PILL}_Y$  and can ask the questions of a given model being adequate or fully abstract.

Since parametricity and full abstraction both have the effect of excluding programs, it is an interesting question how close a parametric model is to being fully abstract. We should thus ask if our concrete model is fully abstract.

We could also ask if it is adequate, and we could study the representation of natural numbers. We proved those to be the Church numerals, but what about functions of natural numbers or the rest of the hierarchy?

### Building a bridge to operational semantics

A recent trend in compiler implementation is to propagate typing information to lower and lower levels, the ultimate example being Typed Assembly Lan-

guage, in order to prove behavioral properties of the final code using techniques such as parametricity. In “Stack-based typed assembly language” Morrisett and coworkers presents a version of TAL using stacks, and in “A Simple Proof Technique for Certain Parametricity Results” Crary proves a stack preservation result using a simplified version of parametricity. It would be interesting to attempt to streamline such arguments via our machinery. This would (at least) involve an adequacy result and a translation from LILY (perhaps with tensor) to something like TAL, preserving the parametricity result.

It would also be interesting to go the other way and look at translations into LILY. LILY is Turing complete and its rich type theory allows encoding of many commonly used type-constructs. It would be good to get a clear understanding of which reasoning principles could be lifted from parametricity in LILY to the encodings. Induction and co-induction principles seem likely candidates.

Another important task, that might greatly raise the impact of our work, is to demonstrate the usefulness of PILLY as a metalanguage for domain theory. Having provided a parametric model, the parametricity axiom of LAPL logic has been proved sound. Thus one might enjoy the consequences of parametricity whilst avoiding the heavy machinery of category theory and domain theory by interpreting languages into PILLY (and showing adequacy). The task would be to do this for an appropriate (interesting) language.

## Understanding Parametricity

A conceptual goal would be to study the very formalization of parametricity. Until now the effort at ITU has used a version of Abadi and Plotkin logic which has been adapted to each individual language to reflect relational parametricity. But relational parametricity seems to be an inadequate formalization once programs are not defined by their input-output relations alone, as happens when effects are added (unless some sort of cps-translation can circumvent this). Thus a comparison with other approaches, that are not built on relations or reflexive graphs, such as that of Grossman, Morrisett and Zdancewic in “Syntactic Type Abstraction” or Melliès and Vouillon in “Recursive polymorphic types and parametricity in an operational framework”, would be interesting. The connection of the latter article with the  $\top\top$ -closed relations of Pitts should also be studied.

## Acknowledgments

I would like to thank Rasmus Møgelberg and Lars Birkedal for collaboration on the article and technical report as well as proofreading and comments of my technical material.

I would also like to thank Lars Birkedal for being my supervisor and the theory department of the IT-University of Copenhagen for being a nice bunch of people to work around.

*The following article is included to provide context for the main result in chapter 6 of the technical report found later. I am responsible for the chapter on concrete models.*

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

Lars Birkedal, Rasmus E. Møgelberg, Rasmus L. Petersen<sup>1,2</sup>

*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 [19] for a recent textbook description). For real programming, one is of course not just interested in a strongly terminating calculus such as the second-order lambda calculus, but also in a language with full recursion. Thus in *loc. cit.* Reynolds also asked for a parametric *domain-theoretic* model of polymorphism [24]. Informally, what is meant [25] by this is a model of an extension

---

<sup>1</sup> This work was partly supported by the Danish Technical Research Council under grant no.: 56-00-0309.

<sup>2</sup> Email: birkedal@itu.dk, mogel@itu.dk, rusmus@itu.dk

of the polymorphic lambda calculus [23,10], with a polymorphic fixed-point operator  $Y: \prod \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha$  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  $\sigma$ ;
- (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 [11]. Indeed Plotkin argued that such a calculus could serve as a very powerful metalanguage for domain theory in which one could also encode recursive types, using parametricity. To prove such consequences of parametricity, Plotkin suggested to use a 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 [20,2] 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  $\text{PILL}_Y$  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  $\text{PILL}_Y$  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 accommodates several different kinds of models, such as the classical one described here and models based on synthetic domain theory [17].

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 lambda 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 intuitionistic set theory [27] (this has been worked out at the time of writing [17]) 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 mentioned 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 [16].

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.<sup>3</sup>

### 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<sub>Y</sub>). 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<sub>Y</sub> (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<sub>Y</sub>). 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).

---

<sup>3</sup> The reader can find an online copy of the technical report at [www.itu.dk/people/birkedal/papers](http://www.itu.dk/people/birkedal/papers).

The logic for parametricity is basically a higher-order logic over  $\text{PILL}_Y$ . Expressions of the logic are formulas in contexts of variables of  $\text{PILL}_Y$  and relations among types of  $\text{PILL}_Y$ . Thus we start by defining  $\text{PILL}_Y$ .

### 2.1 $\text{PILL}_Y$

$\text{PILL}_Y$  is essentially Barber and Plotkin's DILL [1] extended with polymorphism and a fixed point combinator.

Well-formed type expressions in  $\text{PILL}_Y$  are expressions of the form:

$$\alpha_1 : \text{Type}, \dots, \alpha_n : \text{Type} \vdash \sigma : \text{Type}$$

where  $\sigma$  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  $\sigma$  appear on the left hand side of the turnstile. The list of  $\alpha$ 's is called the kind context, and is often denoted simply by  $\Xi$  or  $\alpha$ . 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  $\text{PILL}_Y$  are of the form:

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

where the  $\sigma_i, \sigma'_j$ , and  $\tau$  are well-formed types in the kind context  $\Xi$ . The list  $\mathbf{x}$  is called the intuitionistic type context and is often denoted  $\Gamma$ , and the list  $\mathbf{x}'$  is called the linear type context, often denoted  $\Delta$ . 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:

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

We use  $\lambda^\circ$ , which bear some graphical resemblance to  $\multimap$ , to denote linear function abstraction. And we use  $s, t, u \dots$  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

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

$\Xi \mid \Gamma; \Delta$  is considered well-formed if for all types  $\sigma$  appearing in  $\Gamma$  and  $\Delta$ ,  $\Xi \vdash \sigma : \text{Type}$  is a well-formed type construction.  $\Delta$  and  $\Delta'$  are considered

disjoint if the set of variables appearing in  $\Delta$  is disjoint from the set of variables appearing in  $\Delta'$ . 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  $\text{PILL}_Y$  terms is the least equivalence relation given by the rules of DILL [1] extended with an obvious rule for  $Y$  and  $\beta$ - and  $\eta$ -rules for polymorphic types. We encode ordinary lambda abstraction in the usual way by defining  $\sigma \rightarrow \tau = !\sigma \multimap \tau$  and  $\lambda x: \sigma. t = \lambda^\circ y: !\sigma. \text{let } !x \text{ be } y \text{ in } t$ , where  $y$  is a fresh variable. Using this notation the constant  $Y$  appears with the more familiar looking type  $Y: \Pi\alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha$ .

## 2.2 The logic

As mentioned, expressions of LAPL live in contexts of variables of  $\text{PILL}_Y$  and relations among types of  $\text{PILL}_Y$ . The contexts look like this:

$$\begin{aligned} \Xi \mid \Gamma \mid R_1: \text{Rel}(\tau_1, \tau'_1), \dots, R_n: \text{Rel}(\tau_n, \tau'_n), \\ S_1: \text{AdmRel}(\omega_1, \omega'_1), \dots, S_m: \text{AdmRel}(\omega_m, \omega'_m) \end{aligned}$$

where  $\Xi \mid \Gamma; -$  is a context of  $\text{PILL}_Y$  and the  $\tau_i, \tau'_i, \omega_i, \omega'_i$  are well-formed types in context  $\Xi$ , for all  $i$ . The list of  $R$ 's and  $S$ 's is called the relational context and is often denoted  $\Theta$ . As for the other contexts we allow permutation, but no repetition 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  $\Delta$  in the contexts — the point is that the logic only allows for *intuitionistic reasoning* about terms of  $\text{PILL}_Y$ , whereas  $\text{PILL}_Y$  terms can behave linearly.

Propositions in the logic are given by the syntax:

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

where  $\rho$  is a definable relation (to be defined below).

### 2.2.1 Definable relations

Definable relations, ranged over by  $\rho$ , 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:

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

$$\frac{\frac{\Xi \mid \Gamma, x: \sigma, y: \tau \mid \Theta \vdash \phi: \mathbf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \phi: \mathbf{Rel}(\sigma, \tau)}}{\Xi \mid \Gamma \mid \Theta \vdash \rho: \mathbf{AdmRel}(\sigma, \tau)}}{\Xi \mid \Gamma \mid \Theta \vdash \rho: \mathbf{Rel}(\sigma, \tau)}$$

Notice that in the second rule we can only abstract *intuitionistic* variables to obtain definable relations. In the last rule,  $\rho: \mathbf{AdmRel}(\sigma, \tau)$  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:  $\langle f \rangle = (x: \sigma, y: \tau). fx =_{\tau} y$ , for  $f: \sigma \multimap \tau$ . The equality relation  $eq_{\sigma}$  is defined as the graph of the identity map.

If  $\rho: \mathbf{Rel}(\sigma, \tau)$  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:

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

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

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

$$(x: \sigma', y: \tau'). \rho(fx, gy).$$

### 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  $\mathbf{PILL}_Y$ . 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  $\mathbf{PILL}_Y$ .

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

$$(\rho \multimap \rho'): \mathbf{Rel}((\sigma \multimap \sigma'), (\tau \multimap \tau')),$$

defined by

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

If

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

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

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

as

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

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

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

This definition is due to [9]. 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  $x \neq \perp$ .

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

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

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

Next we will define the tensor product of  $\rho$  and  $\rho'$

$$\rho \otimes \rho' : \mathbf{Rel}((\sigma \otimes \sigma'), (\tau \otimes \tau')),$$

for  $\rho : \mathbf{Rel}(\sigma, \tau)$  and  $\rho' : \mathbf{Rel}(\sigma', \tau')$ . 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

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

Then we define

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

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

$$(x : \sigma \otimes \sigma', y : \tau \otimes \tau'). \forall \alpha, \beta, R : \mathbf{AdmRel}(\alpha, \beta). \forall t : \sigma \multimap \tau \multimap \alpha, t' : \sigma' \multimap \tau' \multimap \beta. (\rho \multimap \rho' \multimap R)(t, t') \supset R(\text{let } x' \otimes x'' \text{ be } x \text{ in } t x' x'', \text{let } y' \otimes y'' \text{ be } y \text{ in } t' y' y'').$$

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

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

which, if we write it out, is

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

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

**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:

$$\frac{\alpha_1, \dots, \alpha_n \vdash \sigma(\boldsymbol{\alpha}) : \mathbf{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho_1 : \mathbf{AdmRel}(\tau_1, \tau_1'), \dots, \rho_n : \mathbf{AdmRel}(\tau_n, \tau_n')}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\boldsymbol{\rho}] : \mathbf{AdmRel}(\sigma(\boldsymbol{\tau}), \sigma(\boldsymbol{\tau}'))}$$

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

**Remark 2.2** Observe that  $\sigma[\boldsymbol{\rho}]$  is a syntactic construction and is not obtained by substitution as in [3]. Still  $\sigma[\rho_1/\alpha_1, \dots, \rho_n/\alpha_n]$  might be a more complete notation, but this quickly becomes overly verbose. In [22]  $\sigma[\boldsymbol{\rho}]$  is to some extent defined inductively on the structure of  $\sigma$ , but in our case that is not enough, since we will need to form  $\sigma[\boldsymbol{\rho}]$  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

$$\begin{array}{c}
 \hline
 \Xi \mid \Gamma \mid \Theta, R: \text{AdmRel}(\sigma, \tau) \vdash R: \text{AdmRel}(\sigma, \tau) \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad \Xi \mid \Gamma; - \vdash t: \sigma' \multimap \sigma, u: \tau' \multimap \tau \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma', y: \tau'). \rho(t x, u y): \text{AdmRel}(\sigma', \tau') \\
 \Xi \mid \Gamma \mid \Theta \vdash \rho, \rho': \text{AdmRel}(\sigma, \tau) \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \rho(x, y) \wedge \rho'(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (y: \tau, x: \sigma). \rho(x, y): \text{AdmRel}(\tau, \sigma) \\
 \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash !\rho: \text{AdmRel}(!\sigma, !\tau) \\
 x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \top: \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad \Xi \mid \Gamma \mid \Theta \vdash \phi: \text{Prop} \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \phi \supset \rho(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi, \alpha \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad \Xi \mid \Gamma \mid \Theta \quad \Xi \vdash \sigma: \text{Type} \quad \Xi \vdash \tau: \text{Type} \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \forall \alpha: \text{Type}. \rho(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma, z: \omega \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \forall z: \omega. \rho(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma \mid \Theta, R: \text{AdmRel}(\omega, \omega') \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \forall R: \text{AdmRel}(\omega, \omega'). \rho(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma \mid \Theta, R: \text{Rel}(\omega, \omega') \vdash \rho: \text{AdmRel}(\sigma, \tau) \quad x, y \notin \Gamma \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \forall R: \text{Rel}(\omega, \omega'). \rho(x, y): \text{AdmRel}(\sigma, \tau) \\
 \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\sigma, \tau), \rho': \text{Rel}(\sigma, \tau) \quad \Xi \mid \Gamma \mid \Theta \mid \top \vdash \rho \equiv \rho' \\
 \hline
 \Xi \mid \Gamma \mid \Theta \vdash \rho': \text{AdmRel}(\sigma, \tau)
 \end{array}$$

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  $x \neq \perp$ .

$$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho: \text{Rel}(!\sigma, !\tau), \rho': \text{AdmRel}(!\sigma, !\tau) \quad x, y \notin \Gamma}{\Xi \mid \Gamma \mid \Theta \mid \forall x: \sigma, y: \tau. \rho(!x, !y) \supset \rho'(!x, !y) \vdash} \quad (2)$$

$$\forall x: !\sigma, y: !\tau. x \downarrow \mathfrak{X} y \downarrow \supset (\rho(x, y) \supset \rho'(x, y))$$

We have rules concerning the interpretation of types as relations:

$$\frac{\alpha \vdash \alpha_i: \text{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\tau, \tau')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \alpha_i[\rho] \equiv \rho_i}$$

$$\frac{\alpha \vdash \sigma \multimap \sigma': \text{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\tau, \tau')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \multimap \sigma')[\rho] \equiv (\sigma[\rho] \multimap \sigma'[\rho])}$$

$$\frac{\alpha \vdash \sigma \otimes \sigma': \text{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\tau, \tau')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \otimes \sigma')[\rho] \equiv (\sigma[\rho] \otimes \sigma'[\rho])}$$

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

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

$$\frac{\alpha \vdash !\sigma: \text{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho: \text{AdmRel}(\tau, \tau')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (!\sigma)[\rho] \equiv !(\sigma[\rho])}$$

Here  $\rho \equiv \rho'$  is shorthand for  $\forall x, y. x\rho y \mathfrak{X} x\rho'y$ .

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

$$\frac{\Xi \mid \Gamma, x: \sigma, y: \tau \mid \Theta \vdash \phi: \text{Prop} \quad \Xi \mid \Gamma; - \vdash t: \sigma, u: \tau}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash ((x: \sigma, y: \tau). \phi)(t, u) \mathfrak{X} \phi[t, u/x, y]}$$

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

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

### 2.2.5 Extensionality and Identity Extension Schemas

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

$$(\forall x: \sigma. t x =_{\tau} u x) \supset t =_{\sigma \rightarrow \tau} u$$

$$(\forall \alpha: \text{Type}. t \alpha =_{\tau} u \alpha) \supset t =_{\prod \alpha: \text{Type}. \tau} u.$$

**Lemma 2.3** *It is provable in the logic that*

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

In particular, extensionality implies

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

The schema

$$- \mid - \mid - \vdash \forall \alpha: \mathbf{Type}. \sigma[eq_{\alpha}] \equiv eq_{\sigma(\alpha)}$$

is called the **identity extension schema**. Here  $\sigma$  ranges over all type expressions.

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

$$- \mid - \mid - \vdash \forall \alpha \forall u: (\prod \beta. \sigma). \forall \beta, \beta'. \forall R: \mathbf{AdmRel}(\beta, \beta'). (u \beta) \sigma[R, eq_{\alpha}](u \beta'),$$

where, for readability, we have omitted  $: \mathbf{Type}$  after  $\beta, \beta'$ , and  $eq_{\alpha}$  is short notation for  $eq_{\alpha_1}, \dots, eq_{\alpha_n}$ . 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  $\alpha \vdash \sigma(\alpha): \mathbf{Type}$  is a type in pure  $PILL_Y$ . There exists a closed type  $rec \alpha. \sigma$  and a pair of terms  $f: rec \alpha. \sigma \multimap \sigma(rec \alpha. \sigma)$ ,  $g: \sigma(rec \alpha. \sigma) \multimap rec \alpha. \sigma$ , 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. [18,14,1,13]).

A model of PILL is a fibred symmetric monoidal adjunction

$$\begin{array}{ccc} \mathbf{LinType} & \begin{array}{c} \xleftarrow{F} \\ \perp \\ \xrightarrow{G} \end{array} & \mathbf{Type} \\ & \searrow p & \swarrow \\ & \mathbf{Kind}, & \end{array}$$

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  $\Omega$ , where  $\Omega$  is  $p$  of the generic object. See [14] for detailed explanation of this definition.

Recall that **PILL** is interpreted in such models as follows. A type  $\sigma$  is interpreted as an object  $\llbracket \sigma \rrbracket \in \mathbf{LinType}$  and we interpret a term  $\alpha \mid \mathbf{x} : \sigma; \mathbf{x}' : \sigma' \vdash t : \tau$  as a morphism

$$\llbracket \sigma_1 \rrbracket \otimes \dots \otimes \llbracket \sigma_n \rrbracket \otimes \llbracket \sigma'_1 \rrbracket \otimes \dots \otimes \llbracket \sigma'_m \rrbracket \multimap \llbracket \tau \rrbracket$$

in **LinType**, where  $! = FG$ . Of course,  $\llbracket !\sigma \rrbracket = \llbracket \sigma \rrbracket$ . 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  $\Xi \mid \mathbf{x} : \sigma; - \vdash t : \tau$ . Then the interpretation of this term in **LinType** is

$$\llbracket \Xi \mid \mathbf{x} : \sigma; - \vdash t : \tau \rrbracket : \otimes_i \llbracket \Xi \mid \sigma_i \rrbracket \multimap \llbracket \Xi \mid \tau \rrbracket.$$

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

$$\llbracket \Xi \mid \mathbf{x} : \sigma; - \vdash t \rrbracket_{\mathbf{Type}} : \prod_i G(\llbracket \sigma_i \rrbracket) \rightarrow G(\llbracket \tau \rrbracket)$$

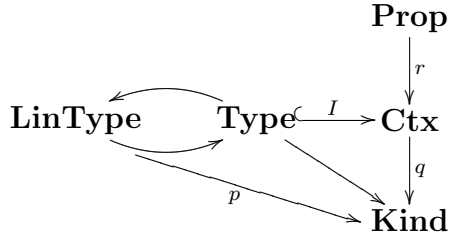
in **Type**.

Finally, a model of **PILL<sub>Y</sub>** is a model of **PILL**, which models a fixed point operator

$$Y : \Pi \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha$$

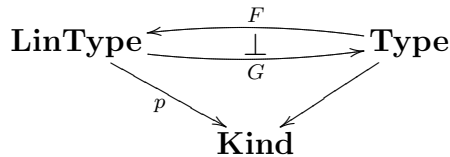
**Definition 4.1** A *pre-LAPL-structure* is

- (i) a schema of categories and functors



such that

- the diagram



is a model of **PILL<sub>Y</sub>**.

- $q$  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  $\Xi \times \Omega \rightarrow \Xi$  in **Kind** [3], where  $\Omega$  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:

$$\begin{array}{ccc}
 \mathbf{LinType} \times_{\mathbf{Kind}} \mathbf{LinType} & \xrightarrow{U} & \mathbf{Ctx} \\
 & \searrow & \swarrow \\
 & & \mathbf{Kind}
 \end{array}$$

(iii) a family of bijections

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

for  $\sigma$  and  $\tau$  in  $\mathbf{LinType}_{\Xi}$  and  $\xi$  in  $\mathbf{Ctx}_{\Xi}$ , which

- is natural in the domain variable  $\xi$
- is natural in  $\sigma, \tau$
- commutes with reindexing functors; that is, if  $\rho : \Xi' \rightarrow \Xi$  is a morphism in **Kind** and  $u : \xi \rightarrow U(\sigma, \tau)$  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  $\rho$ .

Notice that  $\Psi$  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  $\Xi$  in **Kind**, the restriction of  $r$  to the fibre over  $\Xi$  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  $\mathbf{Ctx}$  as follows. A context

$$\Xi \mid x_1 : \sigma_1, \dots, x_n : \sigma_n \mid R_1 : \mathrm{Rel}(\tau_1, \tau'_1), \dots, R_m : \mathrm{Rel}(\tau_m, \tau'_m)$$

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  $\mathbf{LinType} \rightarrow \mathbf{Kind}$ .

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

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

Definable relations with domain  $\sigma$  and codomain  $\tau$  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  $\Xi \mid \Gamma \mid \Theta, R: \mathbf{Rel}(\sigma, \tau) \vdash R: \mathbf{Rel}(\sigma, \tau)$  is interpreted as the projection, and  $\llbracket \Xi \mid \Gamma \mid \Theta \vdash (x: \sigma, y: \tau). \phi: \mathbf{Rel}(\sigma, \tau) \rrbracket$

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

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

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

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

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

where  $\pi, \pi'$  are the projections  $\pi: \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket \rightarrow \llbracket \Xi \mid \Gamma \rrbracket$  and  $\pi': \llbracket \Xi \mid \Gamma \mid \Theta \rrbracket \rightarrow \llbracket \Xi \mid - \mid \Theta \rrbracket$ .

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  $\sigma, \tau$ , 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  $\Psi$  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  $S: \mathbf{AdmRel}(\sigma, \tau)$  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  $\mathbf{AdmRelCtx}$  by the pullback

$$\begin{array}{ccc} \mathbf{AdmRelCtx} & \longrightarrow & \mathbf{Ctx} \\ \langle \partial_0, \partial_1 \rangle \downarrow & \lrcorner & \downarrow \\ \mathbf{Kind} \times \mathbf{Kind} & \xrightarrow{\times} & \mathbf{Kind}. \end{array}$$

We write an object  $\Theta$  in  $\mathbf{AdmRelCtx}$  over  $(\Xi, \Xi')$  as  $\Xi, \Xi' \mid \Theta$ . The fiber of  $\mathbf{LinAdmRelations}$  over an object  $\Xi, \Xi' \mid \Theta$  is

**objects**      triples  $(\phi, \sigma, \tau)$  where  $\sigma$  and  $\tau$  are objects in  $\mathbf{LinType}$  over  $\Xi$  and  $\Xi'$  respectively and  $\phi$  is an admissible relation, i.e. a vertical map

$$\phi: \Theta \rightarrow V(\pi^* \sigma, \pi'^* \tau)$$

in  $\mathbf{Ctx}$ .

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

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

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

**Remark 4.4** In the internal language, objects of  $\mathbf{LinAdmRelations}$  are admissible relations

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

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

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

holds.

There exist two canonical maps of fibrations:

$$\left( \begin{array}{c} \mathbf{LinAdmRelations} \\ \downarrow \\ \mathbf{AdmRelCtx} \end{array} \right) \begin{array}{c} \xrightarrow{\partial_0} \\ \xrightarrow{\partial_1} \end{array} \left( \begin{array}{c} \mathbf{LinType} \\ \downarrow \\ \mathbf{Kind} \end{array} \right).$$

On the base category  $\partial_0, \partial_1$  map an object  $\Xi, \Xi' \mid \Theta$  to  $\Xi$  and  $\Xi'$  respectively. On the total category they map  $(\phi, \sigma, \tau)$  to  $\sigma$  and  $\tau$  respectively. In words,  $\partial_0$  and  $\partial_1$  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 [14], we obtain a PILL-model*

$$\begin{array}{ccc} \mathbf{LinAdmRelations} & \begin{array}{c} \xleftarrow{\quad} \\ \perp \\ \xrightarrow{\quad} \end{array} & \mathbf{AdmRelations} \\ & \searrow & \swarrow \\ & \mathbf{AdmRelCtx} & \end{array}$$

and two maps of PILL-models  $\partial_0, \partial_1$ .

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

$$\mathbf{LinAdmRelations} \rightarrow \mathbf{AdmRelCtx}$$

into a fibred linear category with generic object  $V(\llbracket \alpha \vdash \alpha : \mathbf{Type} \rrbracket, \llbracket \alpha \vdash \alpha : \mathbf{Type} \rrbracket)$ . 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  $\mathbf{LinAdmRelations} \rightarrow \mathbf{AdmRelCtx}$ .  $\square$

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

$$\begin{array}{ccc} \mathbf{LinType} & \begin{array}{c} \xleftarrow{\quad} \\ \perp \\ \xrightarrow{\quad} \end{array} & \mathbf{Type} \\ & \searrow & \swarrow \\ & \mathbf{Kind} & \end{array}$$

to

$$\begin{array}{ccc} \mathbf{LinAdmRelations} & \begin{array}{c} \xleftarrow{\quad} \\ \perp \\ \xrightarrow{\quad} \end{array} & \mathbf{AdmRelations} \\ & \searrow & \swarrow \\ & \mathbf{AdmRelCtx} & \end{array}$$

such that when restricting to the fibred linear categories,  $J$  together with  $\partial_0, \partial_1$  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  $\mathbf{LinType} \rightarrow \mathbf{Kind}$  to  $\mathbf{LinAdmRelations} \rightarrow \mathbf{AdmRelCtx}$ .

We need to show how to interpret the rule

$$\frac{\alpha_1, \dots, \alpha_n \vdash \sigma(\boldsymbol{\alpha}) : \text{Type} \quad \Xi \mid \Gamma \mid \Theta \vdash \rho_1 : \text{AdmRel}(\tau_1, \tau'_1), \dots, \rho_n : \text{AdmRel}(\tau_n, \tau'_n)}{\Xi \mid \Gamma \mid \Theta \vdash \sigma[\boldsymbol{\rho}] : \text{AdmRel}(\sigma(\boldsymbol{\tau}), \sigma(\boldsymbol{\tau}'))}$$

in LAPL-structures.

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

$$\llbracket \Xi \mid - \mid \mathbf{R} : \text{AdmRel}(\boldsymbol{\tau}, \boldsymbol{\tau}') \vdash \sigma[\mathbf{R}] : \text{Rel}(\sigma(\boldsymbol{\tau}), \sigma(\boldsymbol{\tau}')) \rrbracket.$$

For  $\Xi \mid \Gamma \mid \Theta \vdash \boldsymbol{\rho} : \text{AdmRel}(\boldsymbol{\tau}, \boldsymbol{\tau}')$ , we define

$$\begin{aligned} \llbracket \Xi \mid \Gamma \mid \Theta \vdash \sigma[\boldsymbol{\rho}] : \text{AdmRel}(\sigma(\boldsymbol{\tau}), \sigma(\boldsymbol{\tau}')) \rrbracket = \\ \llbracket \Xi \mid - \mid \mathbf{R} : \text{AdmRel}(\boldsymbol{\tau}, \boldsymbol{\tau}') \vdash \sigma[\mathbf{R}] \rrbracket \circ \llbracket \Xi \mid \Gamma \mid \Theta \vdash \boldsymbol{\rho} : \text{AdmRel}(\boldsymbol{\tau}, \boldsymbol{\tau}') \rrbracket. \end{aligned}$$

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

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

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  $\text{PILL}_Y$  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  $\text{PILL}_Y$ . We may split the occurrences of  $\alpha$  in  $\sigma$  into positive and negative obtaining a type  $\alpha, \beta \vdash \sigma(\alpha, \beta)$  such that  $\alpha$

occurs only negatively and  $\beta$  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

$$M: \prod \alpha, \alpha', \beta, \beta'. (\alpha' \multimap \alpha) \rightarrow (\beta \multimap \beta') \rightarrow (\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta'))$$

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

$$\begin{array}{ccc} \mathbf{LinType}^{\text{op}} \times_{\mathbf{Kind}} \mathbf{LinType} & \longrightarrow & \mathbf{LinType} \\ & \searrow & \swarrow \\ & \mathbf{Kind} & \end{array}$$

The category  $\mathbf{LinType}^{\text{op}} \times_{\mathbf{Kind}} \mathbf{LinType}$  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  $\sigma$  and term as above in the internal language of the model (i.e. not necessarily in pure  $\text{PILL}_Y$ ).

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

**Theorem 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  $\alpha$  occurs only positively in  $\sigma$ ,  $\sigma$  can simply be considered a fibred functor

$$\begin{array}{ccc} \mathbf{LinType} & \longrightarrow & \mathbf{LinType} \\ & \searrow & \swarrow \\ & \mathbf{Kind} & \end{array}$$

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  $\tau$  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  $\text{PILL}_Y$ . This could be interesting for example in the case of models with a type for real numbers.

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

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  $\text{PILL}_Y$  (without parametricity) and then show how to make it into a parametric LAPL-structure.

Let  $D$  be a pointed  $\omega$ -chain-complete partial order such that we have  $\Phi: D \rightarrow [D \rightarrow D]$  and  $\Psi: [D \rightarrow D] \rightarrow D$ , both Scott-continuous and satisfying  $\Phi \circ \Psi = id_{[D \rightarrow D]}$ , where  $[D \rightarrow D]$  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  $\perp_D$  to  $\perp_D$ ) and  $\omega$ -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  $\mathbf{AP}(D)_\perp$  of admissible pers and strict continuous functions is the full-on-objects subcategory of  $\mathbf{AP}(D)$  with only those morphisms  $[f]: R \rightarrow S$  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  $\mathbf{AP}(D)_\perp$  is a cartesian sub-category of  $\mathbf{AP}(D)$ . The category  $\mathbf{AP}(D)_\perp$  is symmetric monoidal closed. There is an obvious forgetful functor  $U: \mathbf{AP}(D)_\perp \rightarrow \mathbf{AP}(D)$ , 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  $\mathbf{UFam}(\mathbf{AP}(D)) \rightarrow \mathbf{Set}$  and  $\mathbf{UFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{Set}$  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., [12]). The functors  $L$  and  $U$  easily lift to fibred functors between these two fibrations:

$$\begin{array}{ccc}
 \mathbf{UFam}(\mathbf{AP}(D)_\perp) & \begin{array}{c} \xleftarrow{L} \\ \xrightarrow{\perp} \\ \xrightarrow{U} \end{array} & \mathbf{UFam}(\mathbf{AP}(D)) \\
 & \begin{array}{c} \searrow q \\ \swarrow p \end{array} & \\
 & \mathbf{Set}. & 
 \end{array} \tag{3}$$

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

**Theorem 6.2** *The diagram (3) constitutes a model of  $\text{PILL}_Y$ .*

**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  $\lambda_2$  models. We end up with the following diagram of categories and functors:

$$\begin{array}{ccc}
 & & \mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) \\
 & & \downarrow \\
 \mathbf{PFam}(\mathbf{AP}(D)_{\perp}) & \xleftarrow{L} & \mathbf{PFam}(\mathbf{AP}(D)) \xrightarrow{I} \mathbf{Fam}(\mathbf{Set}) \\
 & \searrow U & \downarrow \\
 & & \mathbf{PAP}(D)
 \end{array} \tag{4}$$

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

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

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

$$\forall \mathbf{R} \in (\mathbf{Obj}(\mathbf{AP}(D)_{\perp}))^n. f^r(\mathbf{R}, \mathbf{R})(\mathbf{Eq}_{\mathbf{R}_j}) = \mathbf{Eq}_{f^p(\mathbf{R})}.$$

Over  $n$ , the fibration  $\mathbf{Fam}(\mathbf{Set})$  contains  $(\mathbf{Obj}(\mathbf{AP}(D)_{\perp}))^n$ -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.,  $I(f^p, f^r): (\text{Obj}(\mathbf{AP}(D)_\perp))^n \rightarrow \mathbf{Set}$  maps  $\mathbf{R}$  to the set of equivalence classes of  $f^p(R)$ . Using  $\mathbf{Fam}(\text{Sub}(\mathbf{Set})) \rightarrow \mathbf{Fam}(\mathbf{Set})$  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

$$\text{Nat} = \llbracket \prod \alpha. (\alpha \multimap \alpha) \rightarrow \alpha \multimap \alpha \rrbracket.$$

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

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

The domain of  $\text{Nat}$  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  $f, g: D \rightarrow D$  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*

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

is admissible.

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

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

$$\perp R \perp, \quad \forall n. \underline{n}R(f^n(x)).$$

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

$$R(d \text{ succ } 0, d f x),$$

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

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

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

$$\perp R \perp, \quad \forall n. \underline{n}Rn$$

we conclude that  $d \text{ succ } 0 = d' \text{ succ } 0$ , and so  $d = d'$ .  $\square$

## 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. Møgelberg. Categorical models for Abadi-Plotkin's Logic for parametricity. *Submitted for publication*, 2004.
- [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] A. Simpson G. Rosolini. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Submitted, 2004.
- [10] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'Etat, Université Paris VII, 1972.
- [11] H. Huwig and A. Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73:101–112, 1990.
- [12] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999.
- [13] 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.
- [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. *Categorical and domain theoretic models of parametric polymorphism*. PhD thesis, IT University of Copenhagen, 2005.
- [16] R.E. Møgelberg. Parametric completion for models of polymorphic intuitionistic / linear lambda calculus. Technical Report TR-2005-60, IT University of Copenhagen, February 2005.
- [17] 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.
- [18] R.L. Petersen and J. Thamsborg. Polymorphism and linearity all in one PILL. Student Project, 2003.
- [19] B.C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

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

*The following excerpt of the technical report “Parametric Domain-theoretic models of Linear Abadi & Plotkin Logic” contains chapter 6 of that report in which a concrete model is described. This chapter was written entirely by me. But the proof of Lemma 6.26, albeit mine tex’ing is due to Rasmus Møgelberg.*

# Parametric Domain-theoretic models of Linear Abadi & Plotkin Logic

Lars Birkedal  
Rasmus Ejlers Møgelberg  
Rasmus Lerchedahl Petersen

## Abstract

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>93</b>
1.1	Outline . . . . .	95
<b>2</b>	<b>Linear Abadi-Plotkin Logic</b>	<b>95</b>
2.1	PILL <sub>Y</sub> . . . . .	95
2.1.1	Equality . . . . .	96
2.1.2	Ordinary lambda abstraction . . . . .	99
2.2	The logic . . . . .	100
2.2.1	Definable relations . . . . .	101
2.2.2	Constructions on definable relations . . . . .	101
2.2.3	Admissible relations . . . . .	103
2.2.4	Axioms and Rules . . . . .	105
2.2.5	Admissible relations preserved by structure maps . . . . .	108
2.2.6	Extensionality and Identity Extension Schemes . . . . .	109
<b>3</b>	<b>Proofs in LAPL</b>	<b>110</b>
3.1	Logical Relations Lemma . . . . .	110
3.2	A category of linear functions . . . . .	113
3.3	Tensor types . . . . .	115

3.4	Unit object . . . . .	116
3.5	Initial objects and coproducts . . . . .	117
3.6	Terminal objects and products . . . . .	118
3.7	Natural Numbers . . . . .	120
3.7.1	Induction principle . . . . .	121
3.8	Types as functors . . . . .	121
3.9	Existential types . . . . .	123
3.10	Initial algebras . . . . .	125
3.11	Final Coalgebras . . . . .	126
3.12	Recursive type equations . . . . .	129
3.12.1	Parametrized initial algebras . . . . .	130
3.12.2	Dialgebras . . . . .	131
3.12.3	Compactness . . . . .	132
3.13	Recursive type equations with parameters . . . . .	135
<b>4</b>	<b>LAPL-structures</b>	<b>136</b>
4.1	Soundness . . . . .	146
4.2	Completeness . . . . .	149
<b>5</b>	<b>Parametric LAPL-structures</b>	<b>152</b>
5.1	Solving recursive domain equations in parametric LAPL-structures . . . . .	152
5.2	Parametrized recursive type equations . . . . .	155
<b>6</b>	<b>Concrete Models</b>	<b>156</b>
6.1	The connection to CUPERS . . . . .	159
6.2	Lifting . . . . .	159
6.3	Going fibred . . . . .	161
6.4	A domain-theoretic model of PILL . . . . .	162
6.5	A parametric domain-theoretic model of PILL . . . . .	163

## 6 Concrete Models

In this section we describe a parametric LAPL structure based on admissible pers over a universal domain as advocated by Plotkin [22]. Pers are known to model the typed  $\lambda$ -calculus, and admissible pers further facilitates a fixed point operator.

As noted in Section 4, to model PILL one has to provide a fibred symmetric monoidal adjunction. We do this by constructing a regular symmetric monoidal adjunction and then define the fibration pointwise.

The standard example is a lifting functor and a forgetful functor. This is also the case here albeit slightly obfuscated, as lifting is coded in the language of the universal domain. This is an adaptation of [13].

Finally, to be able to model polymorphism, the entire construction is done fibred. Parametricity is then ensured by a yet-to-be-described completion process, but first we present the “clean” version:

Let  $D$  be a reflexive cpo, i.e. a pointed  $\omega$ -chain-complete partial order such that we have

$$\Phi: D \rightarrow [D \rightarrow D] \quad \text{and} \quad \Psi: [D \rightarrow D] \rightarrow D,$$

both Scott-continuous and satisfying

$$\Phi \circ \Psi = id_{[D \rightarrow D]}$$

where  $[D \rightarrow D]$  denotes the cpo of continuous functions from  $D$  to  $D$ . We assume, without loss of generality, that both  $\Phi$  and  $\Psi$  are strict. It is standard that there exists strict continuous functions

$$\langle \cdot, \cdot \rangle: D \times D \rightarrow D, \quad \pi: D \rightarrow D \quad \text{and} \quad \pi': D \rightarrow D,$$

such that for all  $d, d' \in D$ :

$$\pi \langle d, d' \rangle = d \quad \text{and} \quad \pi' \langle d, d' \rangle = d'.$$

We use  $1$  to denote  $\Psi(id_{[D \rightarrow D]})$ . Notice that  $\Phi(1) = id_{[D \rightarrow D]}$ .

**Definition 6.1.** An *admissible partial equivalence relation on  $D$*  is a partial equivalence relation  $R$  on  $D$  satisfying

**strict**  $\perp_D R \perp_D$ ,

**$\omega$ -chain complete** For  $(x_n)_{n \in \omega}$  and  $(y_n)_{n \in \omega}$   $\omega$ -chains in  $D$ :

$$(\forall n \in \omega. x_n R y_n) \Rightarrow \bigsqcup_{n \in \omega} x_n R \bigsqcup_{n \in \omega} y_n,$$

**Definition 6.2.** For  $R$  and  $S$  pers on  $D$ , define the set of **equivariant functions from  $R$  to  $S$**  as

$$\mathcal{F}(R, S) = \{f \in [D \rightarrow D] \mid x R y \Rightarrow f(x) S f(y)\}$$

and the set of **strict equivariant functions from  $R$  to  $S$**  as

$$\mathcal{F}(R, S)_\perp = \{f \in \mathcal{F}(R, S) \mid f(\perp_D) = \perp_D\}.$$

Note  $\mathcal{F}(R, S)_\perp \subseteq \mathcal{F}(R, S)$ .

**Definition 6.3.** For  $R$  and  $S$  pers on  $D$ , define on  $\mathcal{F}(R, S)$  or  $\mathcal{F}(R, S)_\perp$  the equivalence relation  $\simeq_{R, S}$  by

$$f \simeq_{R, S} g \Leftrightarrow \forall d \in D. d R d \Rightarrow f(d) S g(d)$$

We write  $\mathbf{PER}(D)$  for the category of partial equivalence relations over  $D$ . Recall that it has partial equivalence relations over  $D$  as objects and that a morphism  $[f]: R \rightarrow S$  is an equivalence class in  $\mathcal{F}(R, S)/\simeq_{R,S}$ . Elements of  $[f]$  are called **realizers** for  $[f]$ .

**Definition 6.4.** We define the category  $\mathbf{AP}(D)$  of admissible partial equivalence relations over  $D$  as the full subcategory of  $\mathbf{PER}(D)$  on the admissible pers.

**Lemma 6.5.** *There is a faithful functor  $Classes: \mathbf{AP}(D) \rightarrow \mathbf{Set}$  mapping an admissible per to the set of equivalence classes and an equivalence class of realizers to the map of equivalence classes they induce.*

*Proof.* This is well-defined since two realizers are equivalent precisely when they define the same map of equivalence classes.  $\square$

**Theorem 6.6.** *The category  $\mathbf{AP}(D)$  is a sub-cartesian closed category of  $\mathbf{PER}(D)$ .*

*Proof.* We recall the constructions. It is straightforward to verify that the resulting pers are admissible. The terminal object  $1$  is the admissible per defined by

$$d \perp d' \Leftrightarrow d = \perp_D = d'.$$

The binary product of  $R$  and  $S$  is

$$\begin{array}{c} \langle d_1, d_2 \rangle R \times S \langle d'_1, d'_2 \rangle \\ \Downarrow \\ d_1 R d'_1 \quad \wedge \quad d_2 S d'_2 \end{array}$$

This is an exhaustive description, understood that only pairs are related in the product. The exponential of  $R$  and  $S$ ,  $S^R$ , is given by

$$d S^R d' \Leftrightarrow \Phi(d) \simeq_{R,S} \Phi(d').$$

$\square$

**Definition 6.7.** The category  $\mathbf{AP}(D)_\perp$  of admissible pers and strict continuous functions is the full-objects subcategory of  $\mathbf{AP}(D)$  with morphisms  $[f]: R \rightarrow S$  equivalence classes in  $\mathcal{F}(R, S)_\perp/\simeq_{R,S}$ .

Note that in  $\mathbf{AP}(D)_\perp$ , morphisms are required to have a *strict* continuous realizer.

**Theorem 6.8.**  $\mathbf{AP}(D)_\perp$  is a cartesian sub-category of  $\mathbf{AP}(D)$ .

*Proof.* Obvious using that  $\pi$ ,  $\pi'$ , and  $\langle \cdot, \cdot \rangle$  are strict.  $\square$

**Theorem 6.9.** *The category  $\mathbf{AP}(D)_\perp$  is symmetric monoidal closed.*

*Proof.* The tensor of  $R$  and  $S$  is

$$\begin{array}{c} \langle d_1, d_2 \rangle R \otimes S \langle d'_1, d'_2 \rangle \\ \Downarrow \\ \langle d_1, d_2 \rangle R \times S \langle d'_1, d'_2 \rangle \\ \vee \\ \left( \begin{array}{c} d_1 R d_1 \quad \wedge \quad d_2 S d_2 \quad \wedge \quad d'_1 R d'_1 \quad \wedge \quad d'_2 S d'_2 \quad \wedge \\ (d_1 R \perp_D \quad \vee \quad d_2 S \perp_D) \quad \wedge \quad (d'_1 R \perp_D \quad \vee \quad d'_2 S \perp_D) \end{array} \right) \end{array}$$

This complicated looking definition is most easily understood through the functor  $Classes$ : The equivalence classes of the tensor product are those of the product with the modification that all pairs where one of the coordinates are related to  $\perp_D$  has been gathered in one big equivalence class. It can thus be seen as a quotient of the product.

The unit of the tensor  $I$  is defined by

$$d I d' \Leftrightarrow d = d' = \perp_D \vee d = d' = \langle 1, \perp_D \rangle.$$

This definition is not taken out of the blue.  $I$  is actually in the image of a lifting functor to be defined later. Notice the “if construct” on  $I$ , which will be available on all lifted relations:

$$\begin{aligned} d I d &\Rightarrow d = \perp_D \vee d = \langle 1, \perp_D \rangle \\ &\Rightarrow \pi(d) = \perp_D \vee \pi(d) = 1 \\ &\Rightarrow \Phi(\pi(d)) = \perp_{[D \rightarrow D]} \vee \Phi(\pi(d)) = id_{[D \rightarrow D]} \end{aligned}$$

Thus for  $d I d$ ,  $\Phi(\pi(d))(d')$  can be read as “if  $d \neq \perp_D$  then  $d'$  else  $\perp_D$ ”. We will use this to construct realizers.

The exponential of  $R$  and  $S$ ,  $R \multimap S$ , is given by

$$d R \multimap S d' \Leftrightarrow d S^R d' \wedge (d'' R \perp_D \Rightarrow \Phi(d)(d'') S \perp_D S \Phi(d')(d''))$$

The proof consist of a series of straightforward verifications. □

For later use we shall mention how regular subobjects look in this category. We use  $A \multimap R$  to express that  $A$  is a regular subobject of  $R$ , if  $R$  is an admissible per.

**Lemma 6.10.**

$$A \multimap R \Leftrightarrow Classes(A) \subseteq Classes(R) \wedge A \in obj(\mathbf{AP}(D)_\perp)$$

*Proof.* In  $\mathbf{PER}(D)$  there is a standard way of constructing an equalizer out of a subset of the equivalence classes. This also works here, and the image of an equalizer is easily seen to be admissible. Thus all regular subobjects have a representative, which is tracked by the identity on  $D$ . □

We also need to know the following fact about admissible pers

**Lemma 6.11.** *If  $I$  is an arbitrary set, and for all  $i \in I$ ,  $R_i$  is an admissible per over  $D$  then*

$$\bigcap_{i \in I} R_i$$

*is an admissible per over  $D$ .*

*Proof.* We intersect relations, which may be seen as sets of pairs. Thus we have the following equation

$$d \bigcap_{i \in I} R_i d' \Leftrightarrow \forall i \in I. d R_i d'$$

which makes the statement obvious, as all  $R_i$  are admissible. □

## 6.1 The connection to CUPERs

In [1] Amadio and Curien show how complete uniform pers over a universal domain allows on to solve domain equations on the per level. As we claim, the same is true for admissible pers, a comparison is natural.

There are, however, some technical issues which makes this a little difficult.

Cupers are defined over a universal solution to the domain equation

$$D = T(D) = (D \multimap D) + (D \times D)$$

In the category  $CPO^{ip}$  of pointed directed complete partial orders and injection-projection pairs. It is known that  $D$  is then the colimit of the  $\omega^{op}$ -chain

$$\perp = 0 \xrightarrow{!} T0 \xrightarrow{T^1} T^2 0 \xrightarrow{T^2 1} \dots$$

Denoting  $T^n 0$  by  $D_n$  we have by the cocone property for each  $n$  the diagram:

$$\begin{array}{ccc} & j_n & \\ & \swarrow \triangleleft & \\ D_n & & D \\ & \searrow i_n & \end{array}$$

Defining  $p_n = i_n \circ j_n : D \rightarrow D$  we can define a cuper on  $D$  as a relation  $R \subseteq D \times D$  such that

- If  $A \subseteq R$  and  $A \subseteq_{dir} D \times D$  then  $\bigsqcup A \in R$ .
- If  $d R e$  then for all  $n$ ,  $p_n(d) R p_n(e)$ .

So apart from using directed-complete rather than chain-complete cpos and living on a universal domain solving a slightly different domain equation, cupers live on a domain with a known structure, and this structure appears in their definition.

Thus the only reasonable way to compare the two notions is to consider a suitably adapted notion of admissible pers over the above described  $D$ . We then find, that the cupers form a proper subset of the admissible pers.

It is noticeable, that cupers facilitate an ordering of the equivalence classes and thus allows one to solve recursive domain equations, while admissible pers achieve this by modeling polymorphic  $\lambda$ -calculus and calling upon parametricity<sup>2</sup>. Hence the two approaches are somewhat different.

## 6.2 Lifting

We now define a notion of lifting, to establish an adjunction between  $\mathbf{AP}(D)$  and  $\mathbf{AP}(D)_\perp$ . Our notion of lifting is essentially the one in [13], specialized to the partial combinatory algebra defined by  $D$ ,  $\Phi$  and  $\Psi$ .

Let  $PD$  denote the power set of  $D$ . Define the map  $L'_0 : PD \rightarrow PD$  by

$$L'_0(A) = \{d \in D \mid \pi(\Phi(d)(1)) = 1 \wedge \pi'(\Phi(d)(1)) \in A\},$$

for  $A \subseteq D$ . And then the map  $L_0 : \mathbf{AP}(D)_0 \rightarrow (\mathbf{AP}(D)_\perp)_0$  by

$$Classes(L_0(R)) = \{L'_0(K) \mid K \in D/R\} \cup \{\{\perp_D\}\}.$$

<sup>2</sup>And calling upon parametricity is, as far as we know, only possible after the deployment of a parametric completion process.

Notice the “if construct” available on a liftet relation: If  $R$  is an admissible per then

$$\begin{aligned} d L_0(R) d &\Rightarrow d = \perp_D \quad \vee \quad \pi(\Phi(d)(1)) = 1 \\ &\Rightarrow \pi(\Phi(d)(1)) = \perp_D \quad \vee \quad \pi(\Phi(d)(1)) = 1 \\ &\Rightarrow \Phi(\pi(\Phi(d)(1))) = \perp_{[D \rightarrow D]} \quad \vee \quad \Phi(\pi(\Phi(d)(1))) = id_{[D \rightarrow D]} \end{aligned}$$

Thus  $\Phi(\pi(\Phi(d)(1)))(d')$  can be read “if  $d \notin \perp_{L_0(R)}$  then  $d'$  else  $\perp_D$ ”, where  $\perp_D$  of course represents  $\perp_S$  for any admissible per  $S$ .

We also have a “lift” and an “unlift”: If  $d \in A$  then  $\Psi(\lambda d' \in D.\langle 1, d \rangle) \in L_0(A)$  and if  $d \in L_0(A)$  then  $\pi'(\Phi(d)(1)) \in A$ . This is convenient for constructing realizers.

Similarly define, for admissible pers  $R$  and  $S$ , the map  $L'_1: \mathcal{F}(R, S) \rightarrow \mathcal{F}(L_0(R), L_0(S))_\perp$  by

$$L'_1(f) = \lambda d \in D. \Phi(\pi(\Phi(d)(1)))(\Psi(\lambda d' \in D.\langle 1, f(\pi'(\Phi(d)(1))) \rangle))$$

which reads “if  $d \notin \perp_{L_0(R)}$  then lift( $f$ (unlift  $d$ )) else  $\perp_D$ ”.

And then the map  $L_1: (\mathbf{AP}(D))_1 \rightarrow (\mathbf{AP}(D)_\perp)_1$  by

$$L_1(f) = [L'_1(t_f)]_{\simeq_{L_0(R), L_0(S)}}$$

for  $f: R \rightarrow S$ . A tedious, but straightforward, verification shows that the definitions of  $L'_0$ ,  $L_0$ ,  $L'_1$  and  $L_1$  all make sense, and that  $L = (L_0, L_1): \mathbf{AP}(D) \rightarrow \mathbf{AP}(D)_\perp$  defines a functor. There is an obvious forgetful functor  $U: \mathbf{AP}(D)_\perp \rightarrow \mathbf{AP}(D)$ .

**Theorem 6.12.** *There is a monoidal adjunction  $L \dashv U$ .*

*Proof.* One first shows that  $L$  is left adjoint to  $U$  in the ordinary sense. The unit of the adjunction is given by  $(\eta_R: R \rightarrow UL(R))_{R \in \mathbf{AP}D_0}$ , all tracked by

$$t_\eta = \lambda d \in D. \Psi(\lambda d' \in D.\langle 1, d \rangle).$$

For  $f: R \rightarrow U(S)$  in  $\mathbf{AP}(D)_\perp$ , the required unique  $h: L(R) \rightarrow S$  in  $\mathbf{AP}(D)_\perp$ , such that  $U(h) \circ \eta_R = f$ , is given by the realizer

$$t_h = \lambda d \in D. \Phi(\pi(\Phi(d)(1)))(t_f(\pi'(\Phi(d)(1)))),$$

where  $t_f$  is a realizer for  $f$ .

To show that the adjunction is monoidal it suffices by to show that the left adjoint  $L$  is a strong symmetric monoidal functor (see [17] for an explanation). To this end, we must exhibit an isomorphism  $m_I: I \rightarrow L(1)$  and a natural isomorphism  $m_{R,S}: L(R) \otimes L(S) \rightarrow L(R \times S)$ . This is mostly straightforward; we just include the definition of  $m_{R,S}: L(R) \otimes L(S) \rightarrow L(R \times S)$ : it is the morphism tracked by the realizer

$$\begin{aligned} &\lambda d \in D. \\ &\Phi(\pi(\Phi(\pi(d)(1))))( \\ &\quad \Phi(\pi(\Phi(\pi'(d)(1))))( \\ &\quad \quad \Psi(\lambda d' \in D.\langle 1, \langle \pi'(\Phi(\pi(d)(1))), \pi'(\Phi(\pi'(d)(1))) \rangle \rangle)) \\ &\quad ) \\ & \end{aligned}$$

which reads

“if  $\pi(d) \neq \perp$  then  
 if  $\pi'(d) \neq \perp$  then  
 lift of  $\langle \text{unlift}(\pi(d)), \text{unlift}(\pi'(d)) \rangle$   
 else  $\perp_D$   
 else  $\perp_D$ ”.

Following a similar chain of thought, the inverse is tracked by

$\lambda d \in D.$   
 $\Phi(\pi(\Phi(d)(1)))($   
 $\langle \Psi(\lambda d' \in D. \langle 1, \pi(\pi'(\Phi(d)(1))) \rangle), \Psi(\lambda d' \in D. \langle 1, \pi'(\pi'(\Phi(d)(1))) \rangle) \rangle$   
 $\left. \right)$

which reads

“if  $d \neq \perp$  then  
 $\langle \text{lift of } \pi(\text{unlift}(d)), \text{lift of } \pi'(\text{unlift}(d)) \rangle$   
 else  $\perp_D$ ”.

□

### 6.3 Going fibred

In order to model polymorphism, we do a fibred version of the adjunction presented in the last subsection, thus arriving at the PILL-model

$$\begin{array}{ccc}
 \mathbf{UFam}(\mathbf{AP}(D)_\perp) & \begin{array}{c} \xleftarrow{L} \\ \perp \\ \xrightarrow{U} \end{array} & \mathbf{UFam}(\mathbf{AP}(D)) \\
 \searrow q & & \swarrow p \\
 & \mathbf{Set}. & 
 \end{array} \tag{10}$$

Define the contravariant functor  $P : \mathbf{Set}^{\text{op}} \rightarrow \mathbf{Cat}$  by mapping set  $I$  to the category  $P(I)$  with

objects:  $(R_i)_{i \in I}$  where for all  $i \in I$ ,  $R_i$  is an object of  $\mathbf{AP}(D)$ .

morphisms:  $(\alpha_i)_{i \in I} : (R_i)_{i \in I} \rightarrow (S_i)_{i \in I}$ , where, for all  $i \in I$ ,  $\alpha_i \in \mathbf{AP}(D)(R_i, S_i)$  and  $\exists \alpha \in [D \rightarrow D]. \forall i \in I. \alpha_i = [\alpha]_{\simeq_{R_i, S_i}}$ .

For a function  $f : I \rightarrow J$ , the reindexing functor  $P(f)$  is simply given by composition with  $f$ .

Define the contravariant functor  $Q : \mathbf{Set}^{\text{op}} \rightarrow \mathbf{Cat}$  given by mapping set  $I$  to the category  $Q(I)$  with

objects:  $(R_i)_{i \in I}$  where for all  $i \in I$ ,  $R_i$  is an object of  $\mathbf{AP}(D)_\perp$ .

morphisms:  $(\alpha_i)_{i \in I} : (R_i)_{i \in I} \rightarrow (S_i)_{i \in I}$  where for all  $i \in I$ ,  $\alpha_i \in \mathbf{AP}(D)_\perp(R_i, S_i)$  and  $\exists \alpha \in [D \rightarrow D]. \forall i \in I. \alpha_i = [\alpha]_{\simeq_{R_i, S_i}}$ .

For a function  $f: I \rightarrow J$ , the reindexing functor  $Q(f)$  is again simply given by composition with  $f$ .

That we have two contravariant functors is obvious. The Grothendieck construction then gives us two split fibrations,  $p: \mathbf{UFam}(\mathbf{AP}(D)) \rightarrow \mathbf{Set}$  and  $q: \mathbf{UFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{Set}$ . The functors  $L$  and  $U$  easily lift to fibred functors between these two fibrations (we abuse notation and also denote the fibred functors by  $L$  and  $U$ ). Explicitly, on objects  $L(I, (R_i)_{i \in I}) = (I, (L(R_i))_{i \in I})$  and on vertical morphisms  $L(I, (f_i)_{i \in I}) = (I, (L(f_i))_{i \in I})$ . Likewise for  $U$ . These are not recursive definitions, they simply look so because of the reuse of letters.

**Theorem 6.13.**  *$L$  and  $U$  are split fibred functors and  $L \dashv U$  is a split fibred strong monoidal adjunction*

*Proof.* It is obvious that  $L$  and  $U$  are split fibred functors; the second part follows immediately from Theorem 6.12.  $\square$

## 6.4 A domain-theoretic model of PILL

To show that (10) is a model of PILL it remains to be shown that  $q$  has a generic object and simple products.

**Lemma 6.14.** *The set  $\Omega = \text{Obj}(\mathbf{AP}(D)_\perp) = \text{Obj}(\mathbf{AP}(D))$  is a split generic object of the fibration  $q$ . The fibration  $q$  has simple split  $\Omega$ -products satisfying the Beck-Chevalley condition.*

*Proof.* The first part is obvious. For the second part, one uses the usual definition for uniform families of ordinary pers and verifies that it restricts to admissible pers: We recall from [12] that given any projection  $\pi_A: A \times \Omega \rightarrow A$  in  $\mathbf{Set}$ , the right adjoint  $\forall_A$  to  $\pi_A^*$  is given on objects by intersection:

$$\forall_A((R_{(a,\omega)})_{(a,\omega) \in A \times \Omega}) = \left( \bigcap_{\omega \in \Omega} R_{(a,\omega)} \right)_{a \in A}.$$

By lemma 6.11 the resulting per is admissible.  $\square$

**Theorem 6.15.** *The diagram (10) constitutes a model of  $PILL_Y$ .*

*Proof.* Given the preceding results it only remains to verify that (1) the structure in the diagram models the polymorphic fixed point combinator and that (2)  $\mathbf{UFam}(\mathbf{AP}(D))$  is equivalent to the category of products of free coalgebras of  $\mathbf{UFam}(\mathbf{AP}(D))_\perp$ .

For (1), the required follows, as expected, because the pers are strict and complete. In more detail, the reasoning is as follows: It is well-known that there is a Scott-continuous function  $y \in [[D \rightarrow D] \rightarrow D]$  giving fixed-points through iterated application at  $\perp_D$ . Since realizers are Scott-continuous functions in  $[D \rightarrow D]$ , every realizer  $\alpha$  has a fixed-point  $y(\alpha)$  in  $D$  given by  $\bigsqcup_n (\alpha^n)(\perp_D)$ . If for some admissible per  $R$ ,  $\alpha \in \mathcal{F}(R, R)$ , then, since  $R$  is strict,  $\alpha$  respects  $R$ , and  $R$  is chain-complete,  $y(\alpha) R y(\alpha)$ . Thus the equivalence class of  $y(\alpha)$  exists and is a fixed-point of the morphism represented by  $\alpha$ . This is applicable both in  $\mathbf{AP}(D)$  and  $\mathbf{AP}(D)_\perp$ , but is not so interesting on strict morphisms. It is, however, in  $\mathbf{AP}(D)_\perp$  that we model the calculus, and thus here we want a fixed-point combinator — albeit only for some morphisms, namely those of type  $!R \multimap R = R \rightarrow R$ , corresponding to morphisms of  $\mathbf{AP}(D)$ . Intuitively, we wish to take such a morphism, transpose it, grab the fixed-point in  $\mathbf{AP}(D)$  and call the whole process a morphism in  $\mathbf{AP}(D)_\perp$ . This is possible, since transposition cascades to the level of realizers. The function that transposes a morphism and returns the fixed-point of the result is continuous. The fixed-point function is in  $\mathcal{F}(R \rightarrow R, R)$ , for any  $R$ , and thus its code is a member of  $\forall \alpha \text{Type} . (\alpha \rightarrow \alpha) \multimap \alpha$ . Precomposing with

a uniform realizer for  $\epsilon$  before taking the code, one easily obtains the polymorphic fixed-point combinator  $Y: \forall \alpha \text{Type}. (\alpha \rightarrow \alpha) \rightarrow \alpha$ . Writing this out, one arrives at

$$\Psi(\lambda d \in D. y(\lambda d' \in D. \Phi(\Phi(\pi(\Phi(d)(1)))(\pi'(\Phi(d)(1))))(1, d'))))$$

For (2), observe that by [17, Proposition 1.21] applied to Theorem 6.8 it suffices to show that  $\mathbf{UFam}(\mathbf{AP}(D))$  is equivalent to the coKleisli category of the adjunction  $L \dashv U$ , but this follows from the fact that  $U$  is a forgetful functor.  $\square$

## 6.5 A parametric domain-theoretic model of PILL

In this section, we introduce a parametric version of the thus far constructed model. It is essentially obtained through a parametric completion process such as the one described in [5] for internal  $\lambda_2$  models (as mentioned in the Introduction, we will generalize that completion process to produce parametric LAPL-structures in [16]).

We will arrive at the diagram

$$\begin{array}{ccc} & L & \\ & \longleftarrow & \\ \mathbf{PFam}(\mathbf{AP}(D)_\perp) & \xleftrightarrow[\quad U \quad]{\quad \perp \quad} & \mathbf{PFam}(\mathbf{AP}(D)) \\ & \searrow & \swarrow \\ & \mathbf{PAP}(D) & \end{array} \quad (11)$$

Our construction is based on reflexive graphs and since our strategy is to obtain relational parametricity for admissible relations (to also model the fixed point combinator in the parametric model), we consider the set  $\text{RefGrph}$  of diagrams

$$A \mapsto R \times S,$$

where  $A$  is a regular subobject of  $R \times S$  in  $\mathbf{AP}(D)_\perp$ . (It is crucial that subobject is in the category with *strict* maps — it means that  $A$  will relate the equivalence class of  $\perp$  in  $R$  to the equivalence class of  $\perp$  in  $S$ .)

Identifying  $\text{RefGrph}^n$  with  $n$ , we define the base category  $\mathbf{PAP}(D)$  by

**Objects:**  $n \in N$  — objects are natural numbers.

**Morphisms:**  $f: n \rightarrow m$  is an  $m$ -tuple,  $(f_1, \dots, f_m)$ , where each  $f_i$  is a pair  $(f_i^p, f_i^r)$  satisfying

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

$$f_i^r \in \Pi_{\vec{R}, \vec{S} \in (\text{Obj}(\mathbf{AP}(D)_\perp))^n} \left( \Pi_{j \in \{1, \dots, n\}} \text{RegSub}(R_j \times S_j) \rightarrow \text{RegSub}(f_i^p(\vec{R}) \times f_i^p(\vec{S})) \right)$$

satisfying

$$\forall \vec{R} \in (\text{Obj}(\mathbf{AP}(D)_\perp))^n. f_i^r(\vec{R}, \vec{R})(Eq_{R_j}^{\vec{R}}) = Eq_{f_i^p(\vec{R})},$$

where the regular subobjects are to be calculated in  $\mathbf{AP}(D)_\perp$ .

We now describe  $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{PAP}(D)$  and  $\mathbf{PFam}(\mathbf{AP}(D)) \rightarrow \mathbf{PAP}(D)$ . As objects they basically contain an (indexed) per and an (indexed) relational interpretation of this per. As morphisms they

have uniformly tracked morphisms that respect admissible relations. We wish to model admissible relations as regular subobjects in  $\mathbf{AP}(D)_\perp$ , so we introduce the notation  $A \mapsto R$  for  $A \in \text{Obj}(\text{RegSub}_{\mathbf{AP}(D)_\perp}(R))$ . We plan to use the Grothendieck construction, and so define indexed categories:  $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$  is defined with

**Objects:**  $f: n \rightarrow 1$  is a morphism in  $\mathbf{PAP}(D)$  from  $n$  to 1.

**Morphisms:**  $\alpha: f \rightarrow g$  is a uniformly tracked family of morphisms  $(\alpha_{\vec{R}})_{\vec{R} \in (\text{Obj}(\mathbf{AP}(D)_\perp))^n}$  of  $\mathbf{AP}(D)_\perp$  such that

$$\alpha_{\vec{R}}: f^p(\vec{R}) \rightarrow g^p(\vec{R}).$$

That  $\alpha$  is uniformly tracked means that there is a strict continuous function  $t_\alpha \in [D \rightarrow D]$  such that

$$\forall \vec{R} \in (\text{Obj}(\mathbf{AP}(D)_\perp))^n. \alpha_{\vec{R}} =_{f^p(\vec{R})} [t_\alpha]_{g^p(\vec{R})}.$$

Furthermore this  $\alpha$  should respect relations:

$$\forall \vec{A} \mapsto \vec{R} \times \vec{S}. \langle a, b \rangle f^r(\vec{R}, \vec{S}, \vec{A}) \langle a, b \rangle \Rightarrow \langle t_\alpha(a), t_\alpha(b) \rangle g^r(\vec{R}, \vec{S}, \vec{A}) \langle t_\alpha(a), t_\alpha(b) \rangle.$$

Quite similarly  $(\mathbf{PFam}(\mathbf{AP}(D)))_n$  is defined as the category with

**Objects:**  $f: n \rightarrow 1$  is a morphism in  $\mathbf{PAP}(D)$  from some  $n$  to 1.

**Morphisms:**  $\alpha: f \rightarrow g$  is a uniformly tracked family of morphisms  $(\alpha_{\vec{R}})_{\vec{R} \in (\text{Obj}(\mathbf{AP}(D)_\perp))^n}$  of  $\mathbf{AP}(D)$  such that

$$\alpha_{\vec{R}}: U(f^p(\vec{R})) \rightarrow U(g^p(\vec{R}))$$

where  $U: \mathbf{AP}(D)_\perp \rightarrow \mathbf{AP}(D)$  is the forgetful functor. That we now ask for morphisms of  $\mathbf{AP}(D)$  removes the demand, that the uniform tracker be strict. Again this  $\alpha$  should respect relations:

$$\forall \vec{A} \mapsto \vec{R} \times \vec{S}. \langle a, b \rangle f^p(\vec{R}, \vec{S}, \vec{A}) \langle a, b \rangle \Rightarrow \langle t_\alpha(a), t_\alpha(b) \rangle g^p(\vec{R}, \vec{S}, \vec{A}) \langle t_\alpha(a), t_\alpha(b) \rangle$$

Here  $A$  is still a regular subobject in  $\mathbf{AP}(D)_\perp$ .

Note that the only difference between the two definitions is the choice of category in which the  $\alpha_{\vec{R}}$  are required to be morphisms.

**Definition 6.16.** Define  $\mathbb{L}: \mathbf{PFam}(\mathbf{AP}(D)) \rightarrow \mathbf{PFam}(\mathbf{AP}(D)_\perp)$  on

**objects** by

$$\mathbb{L}((f^p, f^r)) = (F^p, F^r)$$

where

$$F^p(\vec{R}) = L(f^p(\vec{R}))$$

and

$$F^r((\vec{R}, \vec{S}, \vec{A})) = L(f^r(\vec{R}, \vec{S}, \vec{A}))$$

**morphisms** by

$$\mathbb{L}(\alpha: (f^p, f^r) \rightarrow (g^p, g^r))(R) = L(\alpha(R))$$

Define  $\mathbb{U}: \mathbf{PFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{PFam}(\mathbf{AP}(D))$  in a similar way using  $U$  instead of  $L$ .

**Lemma 6.17.** *If  $A \mapsto R \times S$ , then  $L(A) \mapsto L(R) \times L(S)$ .*

**Lemma 6.18.**  $\mathbb{L}: \mathbf{PFam}(\mathbf{AP}(D)) \rightarrow \mathbf{PFam}(\mathbf{AP}(D)_\perp)$  and  $\mathbb{U}: \mathbf{PFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{PFam}(\mathbf{AP}(D))$  are both functors, and  $\mathbb{L} \dashv \mathbb{U}$

*Proof.* Easy given lemma 6.17 and the fact that for all admissible pers  $R$ ,  $L(Eq_R) = Eq_{L(R)}$ . Lemma 6.17 ensures that the realizer  $t_\eta$  for the unit of  $L \dashv U$  also defines a natural transformation  $id \Rightarrow \mathbb{U}\mathbb{L}$  with the required universal property.  $\square$

By an easy extension of Theorem 6.6, we have:

**Theorem 6.19.**  $\mathbf{PFam}(\mathbf{AP}(D))$  is fibred cartesian closed.

*Proof.* It turns out to be easy, since the product of two regular subobjects turns out to be a regular subobject of the product, and the exponent of two regular subobjects turns out to be a regular subobject of the exponent. Since the adjunction works on the level of realizers and realizers are uniform, the adjunction holds.  $\square$

**Theorem 6.20.**  $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$  is fibred cartesian and fibred symmetric monoidal closed.

*Proof.* We just present the SMCC structure: The tensor product of  $(f^p, f^r)$  and  $(g^p, g^r)$  in the fibre

$$(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n,$$

is denoted by  $(f^p, f^r) \otimes (g^p, g^r)$  and defined by

$$(f^p, f^r) \otimes (g^p, g^r) = (f^p \otimes g^p, f^r \otimes g^r),$$

where

$$(f^p \otimes g^p)(\vec{R}) = f^p(\vec{R}) \otimes g^p(\vec{R})$$

and  $(f^r \otimes g^r)(\vec{R}, \vec{S})(\vec{A})$  is defined as the image of the map  $f^r(\vec{R}, \vec{S})(\vec{A}) \otimes g^r(\vec{R}, \vec{S})(\vec{A}) \rightarrow f^p(\vec{R}) \otimes g^p(\vec{R}) \times f^p(\vec{S}) \otimes g^p(\vec{S})$  tracked by

$$t_t = \lambda d \in D. \langle \langle \pi \pi d, \pi \pi' d \rangle, \langle \pi' \pi d, \pi' \pi' d \rangle \rangle$$

which on pairs of pairs have the following behavior:

$$\langle \langle r_f, s_f \rangle, \langle r_g, s_g \rangle \rangle \mapsto \langle \langle r_f, r_g \rangle, \langle s_f, s_g \rangle \rangle.$$

The unit  $pI$  of the tensor is given by the object  $(\vec{R} \mapsto I, (\vec{R}, \vec{S}) \mapsto Eq_I)$ .

The exponential of  $(f^p, f^r)$  and  $(g^p, g^r)$  in  $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ , is  $(f^p, f^r) \multimap (g^p, g^r)$  defined by

$$(f^p, f^r) \multimap (g^p, g^r) = (f^p \multimap g^p, f^r \multimap g^r)$$

where

$$(f^p \multimap g^p)(\vec{R}) = f^p(\vec{R}) \multimap g^p(\vec{R})$$

and  $(f^r \multimap g^r)(\vec{R}, \vec{S})(\vec{A})$  is defined by

$$\begin{aligned} & \langle d_r, d_s \rangle (f^r \multimap g^r)(\vec{R}, \vec{S})(\vec{A}) \langle d'_r, d'_s \rangle \\ \Downarrow & \\ & \langle r, s \rangle f^r(\vec{R}, \vec{S})(\vec{A}) \langle r', s' \rangle \Rightarrow \langle \Phi(d_r)(r), \Phi(d_s)(s) \rangle g^r(\vec{R}, \vec{S})(\vec{A}) \langle \Phi(d'_r)(r'), \Phi(d'_s)(s') \rangle \\ & \quad \wedge \\ & \langle d_r, d_s \rangle (f^p \multimap g^p)(\vec{R}) \times (f^p \multimap g^p)(\vec{S}) \langle d'_r, d'_s \rangle \end{aligned}$$

This is an exhaustive description, in the sense that only pairs are ever related.

To verify the adjunction  $(-)\otimes(f^p, f^r) \dashv (f^p, f^r) \multimap (-)$ , we use that we know that it holds in the first component and then check that the bijection can be restricted to realizers that define morphisms in the second component; the latter is a direct consequence of the way the relational interpretations of  $\otimes$  and  $\multimap$  are defined.  $\square$

**Lemma 6.21.**  $\mathbb{L} \dashv \mathbb{U}$  is a fibred symmetric monoidal adjunction.

*Proof.* This proceeds much as in the unfibred case. We show that  $\mathbb{L}$  is a fibred strong symmetric monoidal functor. We must provide a morphism  $m_I$  and a natural transformation  $m$ , but we can simply use the same realizers as before, since everything has been defined coordinatewise and these realizers are independent of the specific pers, and hence are uniform realizers.  $\square$

**Lemma 6.22.**  $\Omega = 1$  is a split generic object of  $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \rightarrow \mathbf{PAP}(D)$ .

*Proof.* Obvious.  $\square$

**Lemma 6.23.** If  $(f^p, f^r)$  is an object of  $\mathbf{PFam}(\mathbf{AP}(D)_\perp)_{n+1}$ , then  $(\bigcap f^p, \bigcap f^r)$ , where

$$\bigcap_{R \in \text{Obj}(\mathbf{AP}(D)_\perp)} f^p(R_1, \dots, R_n, R)(x, y) \wedge \forall R, S, A \mapsto R \times S. \langle x, y \rangle f^r(Eq_{R_1}, \dots, Eq_{R_n}, A) \langle x, y \rangle$$

and

$$\langle x, y \rangle (\bigcap f^r)(A_1 \mapsto R_1 \times S_1, \dots, A_n \mapsto R_n \times S_n) \langle x', y' \rangle \iff \forall R, S, A \mapsto R \times S. \langle x, y \rangle f^p(A_1, \dots, A_n, A) \langle x', y' \rangle \wedge (\bigcap f^p)(\vec{R})(x, x') \wedge (\bigcap f^p)(\vec{S})(y, y')$$

is an object of  $\mathbf{PFam}(\mathbf{AP}(D)_\perp)_n$ .

**Lemma 6.24.**  $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$  has simple  $\Omega$ -products.

*Proof.* The construction is as in [12, Section 8.4]. Given a projection  $\pi: n + m \rightarrow n$ , we must define a right adjoint to  $\pi^*$ . This is done by extending the construction of the previous lemma in an obvious way to a functor.  $\square$

**Proposition 6.25.** The diagram (11) constitutes a  $\text{PILL}_Y$  model.

*Proof.* It only remains to verify that the structure models the fixed point combinator. Here we simply use the  $Y$  from Theorem 6.15, which works since relations are strict and chain complete.  $\square$

We now proceed to show that this new  $\text{PILL}_Y$  model can be extended to an LAPL-structure. For this we need just two more fibrations,  $q: \mathbf{Fam}(\mathbf{Set}) \rightarrow \mathbf{PAP}(D)$  and  $r: \mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) \rightarrow \mathbf{Fam}(\mathbf{Set})$ . The fibre of  $\mathbf{Fam}(\mathbf{Set})$  over  $n$  has as

**Objects** maps  $f: \text{obj}(\mathbf{AP}(D))^n \rightarrow \mathbf{Set}$ .

**Morphisms**  $t: f \rightarrow g$  is a family

$$(t_{\vec{R}}: f(\vec{R}) \rightarrow g(\vec{R}))_{\vec{R} \in \text{obj}(\mathbf{AP}(D))^n}$$

and reindexing is given by composition. The fibre of  $\mathbf{Fam}(\mathbf{Sub}(\mathbf{Set}))$  over an object  $f : \mathit{obj}(\mathbf{AP}(D))^n \rightarrow \mathbf{Set}$  is a preorder with

**Objects** maps  $g : \mathit{obj}(\mathbf{AP}(D))^n \rightarrow \mathbf{Set}$ , such that

$$\forall \vec{R} \in \mathit{obj}(\mathbf{AP}(D))^n. g(\vec{R}) \subseteq f(\vec{R}).$$

**Morphisms** There is a morphism  $g \rightarrow g'$  if

$$\forall \vec{R} \in \mathit{obj}(\mathbf{AP}(D))^n. g(\vec{R}) \subseteq g'(\vec{R}).$$

Here reindexing is with respect to morphisms in  $\mathbf{PAP}(D)$  is given by composition, whereas reindexing with respect to morphisms in  $\mathbf{Fam}(\mathbf{Set})$  is given by inverse image.

**Lemma 6.26.**  *$q$  is a fibration with fibred products, and  $(r, q)$  is an indexed first-order logic fibration with simple  $\Omega$ -products and -coproducts.*

*Proof.*

$$\begin{array}{c} \mathbf{Sub}(\mathbf{Set}) \\ \downarrow \\ \mathbf{Set} \end{array}$$

is a first-order logic fibration with generic object and all simple products and coproducts. By Lemma A.8 in [5] we can construct the pullback

$$\begin{array}{ccc} \mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) & \longrightarrow & \mathbf{Sub}(\mathbf{Set}) \\ \downarrow & & \downarrow \\ \mathbf{Set} & \xrightarrow{\text{dom}} & \mathbf{Set} \end{array}$$

obtaining that  $\mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) \rightarrow \mathbf{Set} \xrightarrow{\text{cod}} \mathbf{Set}$  is a composable fibration with the desired qualities. Yet this is not quite the right fibration. Fortunately we have

$$\begin{array}{ccc} \mathbf{Fam}(\mathbf{Set}) & \xrightarrow{\simeq} & \mathbf{Set} \rightarrow \\ & \searrow & \swarrow \text{cod} \\ & \mathbf{Set} & \end{array}$$

by the isomorphism mapping  $(U_x)_{x \in X}$  to  $\coprod_{x \in X} U_x \rightarrow X$ . And now

$$\begin{array}{ccccc} \mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) & \longrightarrow & \mathbf{Fam}(\mathbf{Sub}(\mathbf{Set})) & & \\ \downarrow \lrcorner & & \downarrow & & \downarrow \\ \mathbf{Fam}(\mathbf{Set}) & \longrightarrow & \mathbf{Fam}(\mathbf{Set}) & \xrightarrow{\simeq} & \mathbf{Set} \rightarrow \\ \downarrow \lrcorner & & \searrow & & \downarrow \text{cod} \\ \mathbf{PAP}(D) & \longrightarrow & \mathbf{Set} & & \mathbf{Set} \end{array}$$

is a pullback. The bottom half is a pullback by definition, the map  $\mathbf{PAP}(D) \rightarrow \mathbf{Set}$  operates as follows

$$n \mapsto \mathit{obj}(\mathbf{AP}(D))^n \quad (\vec{f}^p, \vec{f}^r) : n \rightarrow m \mapsto \vec{f}^p : \mathit{obj}(\mathbf{AP}(D))^n \rightarrow \mathit{obj}(\mathbf{AP}(D))^m$$

And the top one easily is a pullback as well. As  $\rightarrow$  preserves products, the leftmost composable fibration have the desired qualities.  $\square$

We can then define the functor  $I: \mathbf{PFam}(\mathbf{AP}(D)) \rightarrow \mathbf{Fam}(\mathbf{Set})$  to be the fibred version of *Classes*.

**Lemma 6.27.** *I is a faithful and product-preserving map of fibrations.*

It is now time to define the contravariant map of fibrations

$$\begin{array}{ccc} \mathbf{PFam}(\mathbf{AP}(D)_\perp)^2 & \xrightarrow{U} & \mathbf{Fam}(\mathbf{Set}) \\ & \searrow & \swarrow \\ & \mathbf{PAP}(D) & \end{array}$$

This is defined at index  $n$  on

**Objects** by  $U(f, g) = \vec{R} \mapsto P(I(f^p(\vec{R})) \times I(g^p(\vec{R})))$ , where  $P(-)$  denotes powerset,

**Morphisms** by  $U(\alpha : f \rightarrow f', \beta : g \rightarrow g') = \vec{R} \mapsto$

$$A \subseteq I(f'^p(\vec{R})) \times I(g'^p(\vec{R})) \mapsto \{ (x, y) \in I(f^p(\vec{R})) \times I(g^p(\vec{R})) \mid (I(\alpha)(x), I(\beta)(y)) \in A \}$$

**Lemma 6.28.** *U is a contravariant map of fibrations.*

*Proof.*  $U$  can be equivalently defined as

$$(\sigma, \tau) \mapsto 2^{I(\sigma) \times I(\tau)},$$

which makes the statement clear. □

We can then define a family of bijections  $(\chi_n)_{n \in \text{Obj}(\mathbf{PAP}(D))}$  such that for all  $f, g \in (\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$  and  $M \in (\mathbf{Fam}(\mathbf{Set}))_n$

$$\chi_n : \mathbf{Fam}(\mathbf{Set})(M, U_n(f, g))_n \rightarrow \text{Obj}(\mathbf{Fam}(\text{Sub}(\mathbf{Set}))_{M \times I_n(\mathbf{U}_n(f) \times \mathbf{U}_n(g))})$$

by

$$\chi_n(h) = \{ (m, (a, b)) \mid (a, b) \in h(m) \}$$

**Lemma 6.29.**  $\chi$  is a bijection, which is natural in the domain variable, is natural in  $f, g$ , and which commutes with reindexing functors.

We have now proved:

**Proposition 6.30.** *The diagram*

$$\begin{array}{ccc} & & \mathbf{Fam}(\text{Sub}(\mathbf{Set})) \\ & & \downarrow \\ \mathbf{PFam}(\mathbf{AP}(D)_\perp) & \begin{array}{c} \xleftarrow{L} \\ \perp \\ \xrightarrow{U} \end{array} & \mathbf{PFam}(\mathbf{AP}(D))^{\subset} \xrightarrow{\quad} \mathbf{Fam}(\mathbf{Set}) \\ & \searrow & \swarrow \\ & \mathbf{PAP}(D) & \end{array} \quad (12)$$

constitutes a pre-LAPL structure.

Now we define a subfunctor  $V$  of  $U$  on

**Objects** by  $V(f, g) = \vec{R} \mapsto \{ I(A) \mid A \mapsto_{\mathbf{AP}(D)_\perp} (f^p(\vec{R}) \times g^p(\vec{R})) \}$ ,

One can now show that  $V$  is closed under all the constructions performable on admissible relations and that it contains all graph relations.

**Lemma 6.31.** *The structure in diagram (12) and  $V$  model admissible relations.*

*Proof.* We refer to figure 4 and provide only a part of a formula to hint at which construction we are debating:

$eq_\sigma$ : Equality on a type  $\sigma$  is modeled as the diagonal subobject of  $\llbracket \sigma \rrbracket \times \llbracket \sigma \rrbracket$ . This corresponds to an admissible relation because it is isomorphic to  $\llbracket \sigma \rrbracket$  by the continuous functions  $\lambda d \in D. \langle d, d \rangle$  and  $\pi$ .

$\rho(t\ x, u\ y)$ : Reindexing an admissible relation by a strict continuous function (i.e.  $(t, u)$ ) is bound to give an admissible relation. We consider chain-completeness: Given two index-wise related chains in  $(t, u)^{-1}(\rho)$ ,  $(t, u)$  taken on these gives us two index-wise related chains in  $\rho$ . Since  $\rho$  is chain-complete their limits are related in  $\rho$ , and since  $(t, u)$  is continuous the limits of the original chains is in the inverse image of the limit in  $\rho$ .

$\rho(x, y) \wedge \rho'(x, y)$ : Conjunction is modeled by intersection, under which admissible relations by lemma 6.11 are stable.

$(x: \tau, y: \sigma). \rho(y, x)$ : swapping the abscissa and ordinal axis does not break admissibility.

$!\rho$ : This is simply the usual lift of relations.

$\top$ : This is all classes. This is admissible.

$\phi \supset \rho(x, y)$ : If  $\phi$  does not hold we get all classes. If  $\phi$  does hold we get  $\rho$  which is admissible.

Quantifications: All quantifications are modeled through intersections and are thus taken care of by lemma 6.11. □

Having come so far, we move on to describe **LinAdmRelations** and **AdmRelCtx** from Section 4. Recall that **AdmRelCtx** is defined as the pullback

$$\begin{array}{ccc} \mathbf{AdmRelCtx} & \longrightarrow & \mathbf{Fam}(\mathbf{Set}) \\ \langle \partial_0, \partial_1 \rangle \downarrow & & \downarrow \\ \mathbf{PAP}(D) \times \mathbf{PAP}(D) & \xrightarrow{\times} & \mathbf{PAP}(D) \end{array}$$

which means that **AdmRelCtx** has as

**Objects** triples  $(n, m, \Phi)$  where  $\Phi: \mathit{obj}(\mathbf{AP}(D))^{n+m} \rightarrow \mathbf{Set}$ , assigns a set to a vector of admissible pers.

**Morphisms** triples  $(f, g, \rho): (n, m, \Phi) \rightarrow (n', m', \Phi')$  where  $f: n \rightarrow n'$  and  $g: m \rightarrow m'$  are morphisms in  $\mathbf{PAP}(D)$  and  $\rho$  is an indexed family of maps

$$\rho = (\rho_{\vec{R}, \vec{S}}: \Phi(\vec{R}, \vec{S}) \rightarrow \Phi'(f^p(\vec{R}), g^p(\vec{S})))_{\vec{R} \in \mathit{obj}(\mathbf{AP}(D))^n, \vec{S} \in \mathit{obj}(\mathbf{AP}(D))^m}$$

where  $\Phi$  and  $\Phi'$  are evaluated on the combined lists of admissible pers.

In this concrete case **LinAdmRelations** can be described as follows: Given an object  $(n, m, \Phi)$  over  $(n, m)$ , the fibre of **LinAdmRelations** over  $(n, m, \Phi)$  has as

**Objects** triples  $(\phi, f, g)$  such that  $f$  and  $g$  are objects of  $\mathbf{PFam}(\mathbf{AP}(D)_\perp)$  over  $n$  and  $m$  respectively and  $\phi$  is an indexed family of maps

$$\phi = (\phi_{\vec{R}, \vec{S}}: \Phi(\vec{R}, \vec{S}) \rightarrow \{ A \mid A \mapsto f^p(\vec{R}) \times g^p(\vec{S}) \})_{\vec{R} \in \text{obj}(\mathbf{AP}(D))^n, \vec{S} \in \text{obj}(\mathbf{AP}(D))^m}$$

**Morphisms** A morphism  $(\phi, f, g) \rightarrow (\psi, f', g')$  is a pair of morphisms

$$(t: f \rightarrow f', u: g \rightarrow g')$$

in  $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$  and  $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_m$ , respectively, such that

$$\begin{aligned} \forall \vec{R} \in \text{obj}(\mathbf{AP}(D))^n, \vec{S} \in \text{obj}(\mathbf{AP}(D))^m. \forall z \in \Phi(\vec{R}, \vec{S}). \\ \langle x, y \rangle \phi(z) \langle x, y \rangle \Rightarrow \langle t(x), u(y) \rangle \psi(z) \langle t(x), u(y) \rangle \end{aligned}$$

Note that we now have two obvious projections  $\partial_0$  and  $\partial_1$ .

Finally we can define the required functor  $J$ .

$$\left( \begin{array}{c} \mathbf{PFam}(\mathbf{AP}(D)_\perp) \\ \downarrow \\ \mathbf{PAP}(D) \end{array} \right) \longrightarrow \left( \begin{array}{c} \mathbf{LinAdmRelations} \\ \downarrow \\ \mathbf{AdmRelCtx} \end{array} \right)$$

For the base categories,  $J$  is defined on

**Objects** by  $n \mapsto (n, n, (\prod_i \{ A \mid A \mapsto R_i \times S_i \})_{\vec{R}, \vec{S} \in \mathbf{AP}(D)^n})$

**Morphisms** by  $f \mapsto (f, f, \prod_i f_i^r)$

and for the total categories,  $J$  is defined on

**Objects** by  $(f^p, f^r) \mapsto (f^r, f, f)$

**Morphisms** by  $\alpha \mapsto (\alpha, \alpha)$ .

This definition on morphisms is legal because  $\alpha$  preserves relations.

In order to show that  $J$  preserves tensor products, we need the following lemma

**Lemma 6.32.** *The tensor product in **LinAdmRelations**  $\rightarrow$  **AdmRelCtx** can be described as*

$$(\rho, f, g) \otimes (\rho', f', g') = (\rho \otimes \rho', f \otimes f', g \otimes g')$$

where  $\rho \otimes \rho'$  is calculated pointwise, i.e for  $z \in \phi(\vec{R}, \vec{S})$

$$(\rho \otimes \rho')_{\vec{R}, \vec{S}}(z) = \rho(z) \otimes \rho'(z)$$

*Proof.* We argue that this construction defines a left adjoint to  $\multimap$ . The standard curry-uncurry-adjunction holds, on the level of realizers even, which is not hard to show.  $\square$

**Lemma 6.33.** *J is a map of linear  $\lambda_2$ -fibrations.*

*Proof.* We must show that  $J$  preserves  $\multimap$ ,  $\otimes$ ,  $\prod$ ,  $I$  and  $!$ .

The constructions in the two categories are virtually identical except for  $\otimes$  until application of lemma 6.32.

To check the case of  $!$  we consider the logical expression for  $!\rho: \text{AdmRel}(\sigma, \tau)$ :

$$(x :!\sigma, y :!\tau).x \downarrow \iff y \downarrow \quad \wedge \quad x \downarrow \supset \rho(\epsilon x, \epsilon y)$$

The expression  $x \downarrow$  equates  $x \neq [\perp]$ . Hence  $x \downarrow \iff y \downarrow$  express the fact that no lifted class is related to  $[\perp]$  in  $!\rho$ .

Further since  $\epsilon$  provides us with unlifted versions of its argument,  $x \downarrow \supset \rho(\epsilon x, \epsilon y)$  states that lifted classes are related in  $!\rho$  only if their unlifted versions are related in  $\rho$ .

This is an exact description of the lifting performed by the functor  $L$ . □

It is easy to see that  $\partial_0 J = id$  and  $\partial_1 J = id$ .

**Theorem 6.34.** *The diagram in (12) constitutes a parametric LAPL-structure.*

*Proof.* By the preceding results it is clear that it is an LAPL-structure; it only remains to show that it is a parametric such. Extensionality holds since the logic is essentially given by regular subobjects, which means that we have very strong equality [12], and thus also extensionality. The parametricity schema is easily verified to hold. □

**Example 6.35.** To ease notation in this example we shall write  $(x, y) \in A$  for  $\langle x, y \rangle A \langle x, y \rangle$  for regular subobjects  $A \mapsto R \times S$ , as we do in LAPL. We will also leave  $\Psi, \Phi$  implicit, and simply write  $f x$  for  $\Phi(f)(x)$ .

We consider the type  $\text{Nat} = \llbracket \prod \alpha. (\alpha \multimap \alpha) \rightarrow \alpha \multimap \alpha \rrbracket$ . By definition

$$d(\text{Nat}^p)d'$$

iff for all  $R, S$  pers and all regular subobjects  $A \mapsto R \times S$ ,  $(f, g) \in (A \multimap A)$  and  $(x, y) \in A$

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

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

**Lemma 6.36.** *Suppose  $\underline{n} \leq \underline{m}$ . Then  $n = m$ .*

*Proof.* Consider the two functions  $f, g: D \rightarrow D$  given by  $f(d) = \langle d, \iota \rangle$ , where  $\iota$  is the code of the identity function, and  $g$  being the first projection. Both are continuous and since  $g \circ f = id$   $f$  is injective. Define the sequence of elements  $x_n = f^n(\perp)$ . This sequence is strictly increasing.

Now, if  $\underline{n} \leq \underline{m}$  then

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

so  $n \leq m$ . Further,

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

so  $m = n$ . □

**Lemma 6.37.** *The per*

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

*is a admissible.*

*Proof.* Direct consequence of the lemma above. □

**Proposition 6.38.** *Suppose  $d(\text{Nat}^p)d$ . Then either  $d = \perp$  or  $d = \underline{n}$ .*

*Proof.* Consider the discrete admissible per  $D$ :

$$\{\{d\} \mid d \in D\}$$

Then given  $f, x$  consider the regular subobject  $A \mapsto \text{Nat} \times D$  given by

$$(\perp, \perp) \in A, \quad \forall n. (\underline{n}, f^n(x)) \in A.$$

$A$  is admissible, simply because it contains no interesting increasing chains. Clearly  $(\text{succ}, f) \in A \multimap A$ , so

$$(d \text{ succ } 0, d f x) \in A,$$

i.e., if  $d \text{ succ } 0 = \perp$ , then  $d f x = \perp$  for all  $f, x$  and so  $d = \perp$ , and if  $d \text{ succ } 0 = \underline{n}$  for some  $n$ , then  $d f x = f^n(x)$ , for all  $f, x$ , so  $d = \underline{n}$ . As we have seen, there are no other possibilities for  $d \text{ succ } 0$ . □

**Proposition 6.39.** *Suppose  $d(\text{Nat}^p)d'$ , then  $d = d'$ .*

*Proof.* Analyzing the above proof we see that

$$d = d \text{ succ } 0$$

By considering the regular subobject  $A \mapsto \text{Nat} \times \text{Nat}$  given by

$$(\perp, \perp) \in A, \quad \forall n. (\underline{n}, \underline{n}) \in A$$

we conclude

$$d \text{ succ } 0 = d' \text{ succ } 0.$$

□

## Acknowledgments

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

## References

- [1] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1998. 6.1
- [2] A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, Edinburgh University, 1997. 2.1, 3.8, 4, 4.2
- [3] P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical report, University of Cambridge, 1995. 1.1
- [4] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of Lily, a polymorphic linear lambda calculus with recursion. In *Fourth International Workshop on Higher Order Operational Techniques in Semantics, Montréal*, volume 41 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 2000. 1
- [5] L. Birkedal and R. Møgelberg. Categorical models for Abadi-Plotkin’s Logic for parametricity. *Mathematical Structures in Computer Science*, 2005. To Appear (Accepted for publication). 1, 3.2, 4, 1, 4.1, 6.5, 6.5
- [6] M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1996. 1
- [7] P.J. Freyd. Algebraically complete categories. In A. Carboni, M. C. Pedicchio, and G. Rosolini, editors, *Category Theory. Proceedings, Como 1990*, volume 1488 of *Lecture Notes in Mathematics*, pages 95–104. Springer-Verlag, 1990. 3.12, 3.12.3
- [8] P.J. Freyd. Recursive types reduced to inductive types. In *Proceedings of the fifth IEEE Conference on Logic in Computer Science*, pages 498–507, 1990. 3.12, 3.12.3
- [9] P.J. Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P.T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1991. 3.12, 3.12.3
- [10] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures de l’arithmétique d’ordre supérieur*. Thèse d’Etat, Université Paris VII, 1972. 1
- [11] H. Huwig and A. Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73:101–112, 1990. 1
- [12] B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., 1999. 2.1, 4, 6.4, 6.5, 6.5
- [13] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Math. Struct. in Comp. Science*, 11, 1996. 6, 6.2
- [14] M. Maietti, P. Maneggia, and E. Ritter. Relating categorical semantics for intuitionistic linear logic. *Applied Categorical Structures*, 2004. To Appear. 1.1
- [15] Maria E Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. Technical Report CSR-01-7, University of Birmingham, School of Computer Science, August 2001. 4

- [16] R. E. Møgelberg. Parametric completion for models of polymorphic intuitionistic / linear lambda calculus. Technical Report TR-2005-60, IT University of Copenhagen, February 2005. 1, 3.3, 6.5
- [17] R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005. 1, 4, 4, 6.2, 6.4
- [18] R. E. Møgelberg, L. Birkedal, and G. Rosolini. Synthetic domain theory and models of linear Abadi & Plotkin logic. Technical Report TR-2005-59, IT University of Copenhagen, February 2005. 1
- [19] R.L. Petersen and J. Thamsborg. Polymorphism and linearity all in one pill. Student Project, 2003. 4
- [20] B.C. Pierce. *Types and Programming Languages*. MIT Press, 2002. 1
- [21] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000. 1
- [22] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1, 3.12, 3.12.3, 6
- [23] Gordon Plotkin and Martín Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, volume 664 of *Lecture Notes in Comput. Sci.*, pages 361–375. Springer, Berlin, 1993. 1, 2, 2.2.3, 2.2.6, 3.8
- [24] J.C. Reynolds. Towards a theory of type structure. In *Colloquium sur La Programmation*, volume 19 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1974. 1
- [25] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983. 1
- [26] J.C. Reynolds. Private communication, June 2000. 1
- [27] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. 1
- [28] Izumi Takeuti. An axiomatic system of parametricity. *Fund. Inform.*, 33(4):397–432, 1998. Typed lambda-calculi and applications (Nancy, 1997). 2.2
- [29] P. Wadler. The Girard-Reynolds isomorphism (second edition). Manuscript, March 2004. 2.2

*The previous technical report was written at a certain level of detail fittings its size and audience. The following text (which should soon be reworked to a technical report) deals with the first part in a little more details, more suited for students or non-experts.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>The Setup</b>	<b>2</b>
2.1	Basic tools . . . . .	2
2.2	Justification . . . . .	3
2.3	The Setup . . . . .	5
2.3.1	$\text{AdmPer}(D)$ . . . . .	5
2.3.2	$\text{AdmPer}(D)_\perp$ . . . . .	10
<b>3</b>	<b>Lifting</b>	<b>17</b>
<b>4</b>	<b>Going fibred</b>	<b>23</b>
<b>5</b>	<b>PILL</b>	<b>25</b>

# 1 Introduction

In these notes, a concrete instance of a split PILL model almost in the sense of [2] is constructed. The reason it is not entirely a split PILL model is, that the definition presented in [2] has been deemed wrong and revised by the authors. Incidentally, though, the model constructed here is a split PILL model in the new, improved sense.

Only one correction has to be made: In [2] Definition 18 condition 5, the adjunction  $U_I \dashv G_I$  is no longer required to be strong, but merely symmetric monoidal.  $U_I$  will then automatically be a strong symmetric monoidal functor.  $G_I$ , however, may not be strong, as is the case in the standard example, where  $U_I$  is a lifting functor and  $G_I$  is forgetful.

A lifting-forgetting adjunction is also the basis of the model constructed herein.

The categories corresponding to  $\mathbb{A}$  and  $\mathbb{D}$  of [2] Definition 18, are categories of pers over a reflexive cpo. This cpo is assumed  $\omega$ -chain complete.

The lifting functor is then an encoded lifting. The construction is a direct translation of that in [7, p.10].

## 2 The Setup

The usual setup is a reflexive  $\omega$ -chain complete cpo.  $D$  will denote one such. This section argues that a number of functions can be assumed strict.

First we need some basic tools:

### 2.1 Basic tools

**Lemma 1** *Let  $D_0$  be a subset of  $D$  closed under supremum of chains, then*

$$\overline{D_0} = \{d \in D \mid \exists d' \in D_0. d \leq d'\}$$

*is also closed under supremum of chains*

**Proof:** For every element  $d$  of  $\overline{D_0}$ , there is an element  $d_0$  of  $D_0$  such that  $d \leq d_0$ . Let  $(x_n)_{n \in \mathbb{N}}$  be a chain in  $\overline{D_0}$ . We can construct a chain  $(y_n)_{n \in \mathbb{N}}$  in  $D_0$  such that for all  $n \in \mathbb{N}$   $x_n \leq y_n$ . Thus  $D_0$  contains an upper bound of  $(x_n)_{n \in \mathbb{N}}$ . By definition  $\overline{D_0}$  then contains the least upper bound of  $(x_n)_{n \in \mathbb{N}}$ .  $\square$

**Lemma 2** *Let  $D$  and  $E$  be pointed chain-complete partial orders, and  $f: D \rightarrow E$  continuous. Let  $D_0$  be a downwards closed subset of  $D$ , which is also closed under supremum of chains. Then the function  $\bar{f}$  defined by*

$$\bar{f}(d) = \begin{cases} \perp_E & d \in D_0 \\ f(d) & d \notin D_0 \end{cases}$$

*is strict and continuous.*

**Proof:** Clearly  $\bar{f}$  is strict and monotone, as  $a \in D_0 \wedge b \notin D_0 \Rightarrow b \not\leq a$ . Given a chain  $(x_i)_{i \in I}$  in  $D$ , either

$$\bigsqcup_{i \in I} x_i \in D_0 \Rightarrow \forall i \in I. x_i \in D_0$$

in which case  $\bar{f}(\bigsqcup_{i \in I} x_i) = \perp_E = \bigsqcup_{i \in I} \bar{f}(x_i)$ , or

$$\bigsqcup_{i \in I} x_i \notin D_0 \Rightarrow \exists i_0 \in I. \forall i \in I. i_0 \leq i \Rightarrow x_i \notin D_0$$

in which case  $\bar{f}(\bigsqcup_{i \in I} x_i) = f(\bigsqcup_{i \in I} x_i) = \bigsqcup_{i \in I} \bar{f}(x_i)$ .  $\square$

We will primarily use this lemma in the simpler special case:

**Lemma 3** *Let  $D$  and  $E$  be pointed chain-complete partial orders, and  $f: D \rightarrow E$  continuous. Then the function  $\bar{f}$  defined by*

$$\bar{f}(d) = \begin{cases} \perp_E & d = \perp_D \\ f(d) & d \neq \perp_D \end{cases}$$

*is strict and continuous.*

**Proof:** Put  $D_0 = \{\perp_D\}$ .  $\square$

We call this, the *strictification* of  $f$ .

**Lemma 4** *Let  $D$  be a chain-complete partial order, and  $(d_n)_{n \in \mathbb{N}}$  a chain in  $D$ . Let  $(d_{n_r})_{r \in \mathbb{N}}$  be a subchain of  $(d_n)_{n \in \mathbb{N}}$  (meaning  $(n_r)_{r \in \mathbb{N}}$  is strictly increasing). Then*

$$\bigsqcup_{r \in \mathbb{N}} d_{n_r} = \bigsqcup_{n \in \mathbb{N}} d_n$$

**Proof:** For all  $n_0 \in \mathbb{N}$  there is a  $r_0 \in \mathbb{N}$  so that  $n_{r_0} > n_0$ . Thus  $d_{n_{r_0}} \geq d_{n_0}$  and since  $\bigsqcup_{r \in \mathbb{N}} d_{n_r} \geq d_{n_{r_0}}$ ,  $\bigsqcup_{r \in \mathbb{N}} d_{n_r} \geq d_{n_0}$ . So  $\bigsqcup_{r \in \mathbb{N}} d_{n_r}$  is an upper bound of  $(d_n)_{n \in \mathbb{N}}$ . If  $d$  is an upper bound of  $(d_n)_{n \in \mathbb{N}}$ ,  $d$  is an upper bound of  $(d_{n_r})_{r \in \mathbb{N}}$ , and so  $\bigsqcup_{r \in \mathbb{N}} d_{n_r} \leq d$ . So  $\bigsqcup_{r \in \mathbb{N}} d_{n_r}$  is the least upper bound of  $(d_n)_{n \in \mathbb{N}}$ .  $\square$

Now the following lemmas will justify, that we sometimes assume certain maps strict.

## 2.2 Justification

**Lemma 5** *Let  $D$  and  $E$  be pointed chain-complete partial orders such that we have*

$$f: D \rightarrow E \quad \text{and} \quad g: E \rightarrow D$$

*both continuous, satisfying*

$$f \circ g = id_E,$$

*then*

$$\bar{f} \circ \bar{g} = id_E$$

**Proof:** We simply evaluate the composite function in all  $e \in E$ . There are three cases:

$e \neq \perp_E$  and  $g(e) \neq \perp_D$

$$(\bar{f} \circ \bar{g})(e) = \bar{f}(\bar{g}(e)) = \bar{f}(g(e)) = f(g(e)) = (f \circ g)(e) = e$$

$e \neq \perp_E$  and  $g(e) = \perp_D$  This is not possible. Since  $g$  is monotone, if  $g(e) = \perp_D$ , then also  $g(\perp_E) = \perp_D$ . But the equation  $f \circ g = id_E$  ensures, that  $g$  is injective.

$e = \perp_E$

$$(\bar{f} \circ \bar{g})(\perp_E) = \bar{f}(\bar{g}(\perp_E)) = \bar{f}(\perp_D) = \perp_E$$

□

The following two results are really just easy corollaries, but since they are what we are really after, we have promoted them to lemmas.

**Lemma 6** *Let  $D$  be a pointed chain-complete partial order such that we have*

$$\Phi: D \rightarrow [D \rightarrow D] \quad \text{and} \quad \Psi: [D \rightarrow D] \rightarrow D$$

*both continuous, satisfying*

$$\Phi \circ \Psi = id_{[D \rightarrow D]},$$

*then*

$$\bar{\Phi} \circ \bar{\Psi} = id_{[D \rightarrow D]}$$

**Proof:** The result readily follows from lemma 5. □

$\Phi$  and  $\Psi$  being strict is not usually a part of  $D$  being a reflexive cpo, but we need them to be later on.

**Lemma 7** *Let  $D$  be a pointed chain-complete partial order such that we have*

$$\Phi: D \rightarrow [D \rightarrow D] \quad \text{and} \quad \Psi: [D \rightarrow D] \rightarrow D$$

*both continuous, satisfying*

$$\Phi \circ \Psi = id_{[D \rightarrow D]},$$

*then there exists strict continuous functions*

$$\langle \cdot, \cdot \rangle: D \times D \rightarrow D, \quad \pi: D \rightarrow D \quad \text{and} \quad \pi': D \rightarrow D$$

*such that for all  $d, d' \in D$ :*

$$\pi \langle d, d' \rangle = d \quad \text{and} \quad \pi' \langle d, d' \rangle = d'$$

**Proof:** As in [1] we have  $\langle \cdot, \cdot \rangle$ ,  $\pi$  and  $\pi'$  continuous, satisfying the equations. Lemma 5 then applies. □

## 2.3 The Setup

Let  $D$  be a pointed chain-complete partial order such that we have

$$\Phi: D \rightarrow [D \rightarrow D] \quad \text{and} \quad \Psi: [D \rightarrow D] \rightarrow D$$

both strict and continuous, satisfying

$$\Phi \circ \Psi = id_{[D \rightarrow D]}$$

**Definition 8** Let  $i$  denote  $\Psi(id_{[D \rightarrow D]})$

Notice that  $\Phi(i) = id_{[D \rightarrow D]}$ .

**Definition 9** An admissible partial equivalence relation on  $D$  is a partial equivalence relation  $R$  on  $D$  satisfying

strict  $\perp_D R \perp_D$

chain complete For  $(x_i)_{i \in I}$  and  $(y_i)_{i \in I}$  chains in  $D$ :

$$(\forall i \in I. x_i R y_i) \Rightarrow \bigsqcup_{i \in I} x_i R \bigsqcup_{i \in I} y_i$$

**Definition 10** For  $R$  and  $S$  admissible pers on  $D$ , define the set of functions admissible from  $R$  to  $S$  as

$$\mathcal{F}(R, S) = \{f \in [D \rightarrow D] \mid x R y \Rightarrow f(x) S f(y)\}$$

and the set of strict functions admissible from  $R$  to  $S$  as

$$\mathcal{F}(R, S)_\perp = \{f \in \mathcal{F}(R, S) \mid f(\perp_D) = \perp_D\}$$

Note  $\mathcal{F}(R, S)_\perp \subseteq \mathcal{F}(R, S)$ .

**Definition 11** For  $R$  and  $S$  admissible pers on  $D$ , define on  $\mathcal{F}(R, S)$  or  $\mathcal{F}(R, S)_\perp$  the equivalence relation  $R \sim_S$  by

$$f R \sim_S g \Leftrightarrow \forall d \in D. d R d \Rightarrow f(d) S g(d)$$

### 2.3.1 AdmPer( $D$ )

Consider the category  $\text{AdmPer}(D)$  of admissible partial equivalence relations on  $D$  and continuous functions:

Objects: are admissible pers.

Morphisms: a morphism  $[f]: R \rightarrow S$  is an equivalence class in  $\mathcal{F}(R, S) / R \sim_S$ . Elements of  $[f]$  are called *realizers* for  $[f]$ .

**Definition 12** For  $R$  and  $S$  admissible pers, define

$$R \times S = \{(\langle d_1, d_2 \rangle, \langle d'_1, d'_2 \rangle) \mid d_1 R d'_1 \wedge d_2 S d'_2\}$$

**Lemma 13** If  $R$  and  $S$  are admissible pers, then so is  $R \times S$ .

**Proof:** As pairing is injective,  $R \times S$  is obviously a per, but is it admissible? Let  $(x_i)_{i \in I}$  and  $(y_i)_{i \in I}$  be chains so that

$$\forall i \in I. x_i R \times S y_i.$$

Then

$$\forall i \in I. \exists x1_i, x2_i, y1_i, y2_i \in D. \\ x1_i R y1_i \wedge x2_i S y2_i \wedge x_i = \langle x1_i, x2_i \rangle \wedge y_i = \langle y1_i, y2_i \rangle,$$

since only pairs appear in the equivalence classes of  $R \times S$ . As  $\pi$  and  $\pi'$  are continuous,

$$(x1_i)_{i \in I} = \pi((x_i)_{i \in I}), \quad (x2_i)_{i \in I} = \pi'((x_i)_{i \in I}),$$

$$(y1_i)_{i \in I} = \pi((y_i)_{i \in I}) \quad \text{and} \quad (y2_i)_{i \in I} = \pi'((y_i)_{i \in I}).$$

are all chains; the former two related in  $R$  and the latter two related in  $S$ . Now several facts work our way:

$$\bigsqcup_{i \in I} x1_i R \bigsqcup_{i \in I} y1_i, \quad \bigsqcup_{i \in I} x2_i S \bigsqcup_{i \in I} y2_i,$$

$$\langle \bigsqcup_{i \in I} x1_i, \bigsqcup_{i \in I} x2_i \rangle = \bigsqcup_{i \in I} \langle x1_i, x2_i \rangle = \bigsqcup_{i \in I} x_i \quad \text{and} \quad \langle \bigsqcup_{i \in I} y1_i, \bigsqcup_{i \in I} y2_i \rangle = \bigsqcup_{i \in I} \langle y1_i, y2_i \rangle = \bigsqcup_{i \in I} y_i,$$

all of which immediately amounts to  $\bigsqcup_{i \in I} x_i R \times S \bigsqcup_{i \in I} y_i$ . That  $\perp_D R \times S \perp_D$  follows from pairing being strict.  $\square$

**Proposition 14**  $R \times S$  defines a cartesian product on  $\text{AdmPer}(D)$ .

**Proof:** Projections are easily defined as the equivalence classes  $p$  and  $p'$  of  $\pi$  and  $\pi'$ , as these obviously preserve the relevant equivalences. To see, that we have indeed a terminal cone, consider an object  $Q$  and morphisms  $f: Q \rightarrow R$  and  $g: Q \rightarrow S$ , with realizers  $t_f$  and  $t_g$ . Define the continuous function

$$(t_f, t_g) = \lambda d \in D. \langle t_f(d), t_g(d) \rangle$$

that readily preserves equivalence from  $Q$  to  $R \times S$ . This definition is independent of the choice of representatives; assume  $t'_f \sim t_f$  and  $t'_g \sim t_g$ , then for all  $d \in D$ :

$$d Q \Rightarrow t_f(d) R t'_f(d) \wedge t_g(d) S t'_g(d) \\ \Leftrightarrow \langle t_f(d), t_g(d) \rangle R \times S \langle t'_f(d), t'_g(d) \rangle$$

We have thus defined a morphism  $\langle f, g \rangle$  hither. Clearly  $p\langle f, g \rangle = f$  and  $p'\langle f, g \rangle = g$ .

Assume, conversely, that we have a morphism  $h: Q \rightarrow R \times S$ , so that  $ph = f$  and  $p'h = g$ . Then  $h$  has a realizer  $t_h$  satisfying

$$\forall d \in D. dQd \Rightarrow (\pi(t_h(d)) R t_f(d) \quad \wedge \quad \pi'(t_h(d)) S t_g(d)).$$

Now  $t_h$  preserving equivalence gives for all  $d \in D$

$$\begin{aligned} d Q d &\Rightarrow t_h(d) R \times S t_h(d) \\ &\Rightarrow \exists d_1, d_2 \in D. t_h(d) = \langle d_1, d_2 \rangle \\ &\Rightarrow t_h(d) R \times S \langle t_f(d), t_g(d) \rangle, \end{aligned}$$

rendering  $h$  unique.  $\square$

**Definition 15** Let  $\mathbb{1}$  denote the admissible per defined by

$$d \mathbb{1} d' \Leftrightarrow d = \perp_D = d'$$

**Lemma 16**  $\mathbb{1}$  is a terminal object of  $\text{AdmPer}(D)$ .

Proof: Given any admissible per  $R$ , a morphism  $R \rightarrow \mathbb{1}$  is defined by the realizer

$$\lambda d \in D. \perp_D$$

On the other hand, since there is only one equivalence class in  $\mathbb{1}$ , there can at most be one morphism  $R \rightarrow \mathbb{1}$ .  $\square$

**Lemma 17** For all admissible pers  $R$  and  $S$ ,  $S^R$  given by

$$d S^R d' \Leftrightarrow \Phi(d) R \sim_S \Phi(d')$$

is also an admissible per.

Proof: As  $R \sim_S$  is an equivalence relation,  $S^R$  is immediately a per. Let  $(x_i)_{i \in I}$  and  $(y_i)_{i \in I}$  be chains so that

$$\forall i \in I. x_i S^R y_i.$$

Then for all  $d \in D$ ,  $(\Phi(x_i)d)_{i \in I}$  and  $(\Phi(y_i)d)_{i \in I}$  are chains in  $D$ . Furthermore, as for all  $i \in I$  we have  $\Phi(x_i) \in \mathcal{F}(R, S)$  and  $\Phi(x_i) R \sim_S \Phi(y_i)$ , we get

$$\begin{aligned} d R d' &\Rightarrow \forall i \in I. \Phi(x_i)d S \Phi(x_i)d' \\ &\Rightarrow \bigsqcup_{i \in I} \Phi(x_i)d S \bigsqcup_{i \in I} \Phi(x_i)d' \\ &\Rightarrow \Phi(\bigsqcup_{i \in I} x_i)d S \Phi(\bigsqcup_{i \in I} x_i)d' \end{aligned}$$

and

$$\begin{aligned} d R d &\Rightarrow \forall i \in I. \Phi(x_i)d S \Phi(y_i)d \\ &\Rightarrow \bigsqcup_{i \in I} \Phi(x_i)d S \bigsqcup_{i \in I} \Phi(y_i)d \\ &\Rightarrow \Phi(\bigsqcup_{i \in I} x_i)d S \Phi(\bigsqcup_{i \in I} y_i)d \end{aligned}$$

Giving us<sup>1</sup>  $\Phi(\bigsqcup_{i \in I} x_i) \ R \sim_S \ \Phi(\bigsqcup_{i \in I} y_i)$ , yielding  $\bigsqcup_{i \in I} x_i \ S^R \ \bigsqcup_{i \in I} y_i$ .  
As  $\Phi$  is assumed strict,  $\Phi(\perp_D) = \lambda d \in D.\perp_D$ .  $\square$

**Lemma 18** *The map  $S \mapsto S^R$  extends to a functor.*

**Proof:** Given a morphism  $f: S \rightarrow U$ , define  $f^R$  as the morphism tracked by

$$\circ t_f = \lambda d \in D.\Psi(\lambda d' \in D.t_f(\Phi(d)(d'))),$$

where  $t_f$  is some realizer for  $f$ . The choice of this realizer is irrelevant; assume  $t_f, t'_f \in [f]$ :

$$\begin{aligned} d \ S^R \ d &\Rightarrow (d' \ R \ d' \Rightarrow \Phi(d)(d') \ S \ \Phi(d)(d')) \\ &\Rightarrow (d' \ R \ d' \Rightarrow t_f(\Phi(d)(d')) \ U \ t'_f(\Phi(d)(d'))) \\ &\Rightarrow \lambda d' \in D.t_f(\Phi(d)(d')) \ R \sim_U \ \lambda d' \in D.t'_f(\Phi(d)(d')) \\ &\Rightarrow \Psi(\lambda d' \in D.t_f(\Phi(d)(d'))) \ U^R \ \Psi(\lambda d' \in D.t'_f(\Phi(d)(d'))), \end{aligned}$$

since  $\Phi \circ \Psi = id_{[D \rightarrow D]}$ , and  $\circ t_f$  is in  $\mathcal{F}(S^R, U^R)$ :

$$\begin{aligned} d \ S^R \ d' &\Rightarrow (d'' \ R \ d'' \Rightarrow \Phi(d)(d'') \ S \ \Phi(d')(d'')) \\ &\Rightarrow (d'' \ R \ d'' \Rightarrow t_f(\Phi(d)(d'')) \ U \ t_f(\Phi(d')(d''))) \\ &\Rightarrow \lambda d'' \in D.t_f(\Phi(d)(d'')) \ R \sim_U \ \lambda d'' \in D.t_f(\Phi(d')(d'')) \\ &\Rightarrow \Psi(\lambda d'' \in D.t_f(\Phi(d)(d''))) \ U^R \ \Psi(\lambda d'' \in D.t_f(\Phi(d')(d''))). \end{aligned}$$

The operation is well-behaved on identities, consider  $id_S$  tracked by  $id_D$ :

$$\circ id_D = \lambda d \in D.\Psi(\lambda d' \in D.id_D(\Phi(d)(d'))) = \lambda d \in D.\Psi(\lambda d' \in D.\Phi(d)(d'))$$

$$\begin{aligned} d \ S^R \ d &\Rightarrow (d' \ R \ d' \Rightarrow \Phi(d)(d') \ S \ \Phi(d)(d')) \\ &\Rightarrow (d' \ R \ d' \Rightarrow \Phi(\Psi(\lambda d'' \in D.\Phi(d)(d'')))(d') \ S \ \Phi(d)(d')) \\ &\Rightarrow \Psi(\lambda d'' \in D.\Phi(d)(d'')) \ S^R \ d, \end{aligned}$$

so  $\lambda d \in D.\Psi(\lambda d' \in D.\Phi(d)(d')) \ S^R \sim_{S^R} id_D$ , and  $id_D$  tracks  $id_{S^R}$ .

And it also works for composition; given realizers  $t_f \in \mathcal{F}(S, U)$  and  $t_g \in \mathcal{F}(U, V)$ , we have for  $d \in D$ :

$$\begin{aligned} (\circ t_g) \circ (\circ t_f)(d) &= \circ t_g(\circ t_f d) \\ &= \Psi(\lambda d' \in D.t_g(\Phi(\Psi(\lambda d'' \in D.t_f(\Phi(d)(d'')))(d')))(d')) \\ &= \Psi(\lambda d' \in D.t_g((\lambda d'' \in D.t_f(\Phi(d)(d'')))(d')))(d')) \\ &= \Psi(\lambda d' \in D.t_g(t_f(\Phi(d)(d')))) \\ &= \Psi(\lambda d' \in D.(t_g \circ t_f)(\Phi(d)(d'))) \\ &= \circ(t_g \circ t_f)(d) \end{aligned}$$

$\square$

**Proposition 19**  $- \times R \dashv (-)^R$

<sup>1</sup>Supposing, that we have done the first calculation for  $(y_i)_{i \in I}$  as well. . .

**Proof:** For objects  $S$  and  $T$ , we consider  $\text{AdmPer}(D)(T \times R, S)$  and  $\text{AdmPer}(D)(T, S^R)$ . A natural bijection  $\bar{\cdot}$  between these is given by

$$t_f \in \mathcal{F}(T \times R, S) \quad \mapsto \quad \bar{t}_f = \lambda d \in D. \Psi(\lambda d' \in D.t_f(\langle d, d' \rangle))$$

which is in  $\mathcal{F}(T, S^R)$

$$\begin{aligned} d_1 T d'_1 &\Rightarrow (d_2 R d'_2 \Rightarrow \langle d_1, d_2 \rangle T \times R \langle d'_1, d'_2 \rangle) \\ &\Rightarrow (d_2 R d'_2 \Rightarrow t_f(\langle d_1, d_2 \rangle) S t_f(\langle d'_1, d'_2 \rangle)) \\ &\Rightarrow \Psi(\lambda d' \in D.t_f(\langle d_1, d' \rangle)) S^R \Psi(\lambda d' \in D.t_f(\langle d'_1, d' \rangle)) \end{aligned}$$

and whose equivalence class under  $T \sim_{S^R}$  is independent of choice of realizers; for  $t_f T \times R \sim_S t'_f$ :

$$\begin{aligned} d T d &\Rightarrow (d_2 R d'_2 \Rightarrow \langle d, d_2 \rangle T \times R \langle d, d'_2 \rangle) \\ &\Rightarrow (d_2 R d'_2 \Rightarrow t_f(\langle d, d_2 \rangle) S t'_f(\langle d, d'_2 \rangle)) \\ &\Rightarrow \Psi(\lambda d' \in D.t_f(\langle d, d' \rangle)) S^R \Psi(\lambda d' \in D.t'_f(\langle d, d' \rangle)) \end{aligned}$$

so that it is well-defined. That it is a bijection is obvious, once one have found the inverse :

$$t_g \in \mathcal{F}(T, S^R) \quad \mapsto \quad \underline{t}_g = \lambda d \in D. \Phi(t_g(\pi(d)))(\pi'(d))$$

which is in  $\mathcal{F}(T \times R, S)$

$$\begin{aligned} \langle d_1, d_2 \rangle T \times R \langle d'_1, d'_2 \rangle &\Rightarrow d_1 T d'_1 \wedge d_2 R d'_2 \\ &\Rightarrow t_g(d_1) S^R t_g(d'_1) \wedge d_2 R d'_2 \\ &\Rightarrow \Phi(t_g(d_1))(d_2) R \Phi(t_g(d'_1))(d'_2) \end{aligned}$$

and whose equivalence class under  $T \times R \sim_S$  is independent of choice of realizers; for  $t_g T \sim_{S^R} t'_g$ :

$$\begin{aligned} \langle d_1, d_2 \rangle T \times R \langle d_1, d_2 \rangle &\Rightarrow d_1 T d_1 \wedge d_2 R d_2 \\ &\Rightarrow t_g(d_1) S^R t'_g(d_1) \wedge d_2 R d_2 \\ &\Rightarrow \Phi(t_g(d_1))(d_2) R \Phi(t'_g(d_1))(d'_2) \end{aligned}$$

so that it is well-defined. Now composing gives

$$\begin{aligned} t_f \in \mathcal{F}(T \times R, S) &\quad \mapsto \quad \lambda d \in D. \Psi(\lambda d' \in D.t_f(\langle d, d' \rangle)) \\ &\quad \mapsto \quad \lambda d'' \in D. \Phi(\Psi(\lambda d' \in D.t_f(\langle \pi(d''), d' \rangle)))(\pi'(d'')) \\ &\quad = \quad \lambda d'' \in D. (\lambda d' \in D.t_f(\langle \pi(d''), d' \rangle))(\pi'(d'')) \\ &\quad = \quad \lambda d'' \in D. (t_f(\langle \pi(d''), \pi'(d'') \rangle)) \\ T \times R \sim_S t_f &\end{aligned}$$

since they are equal on pairs. Composing the other way gives

$$\begin{aligned} t_g \in \mathcal{F}(T, S^R) &\quad \mapsto \quad \lambda d \in D. \Phi(t_g(\pi(d)))(\pi'(d)) \\ &\quad \mapsto \quad \lambda d' \in D. \Psi(\lambda d'' \in D. \Phi(t_g(d''))(d'')) \\ T \sim_{S^R} t_g &\end{aligned}$$

since

$$\begin{aligned}
d' T d' &\Rightarrow t_g(d') S^R t_g(d') \\
&\Rightarrow (d'' R d'' \Rightarrow) \\
&\Rightarrow \Psi(\lambda d' \in D.t_f(\langle d, d' \rangle)) S^R \Psi(\lambda d' \in D.t'_f(\langle d, d' \rangle))
\end{aligned}$$

To see, that the bijection is natural, consider  $t_f: T' \rightarrow T$ ,  $t_\alpha: T \times R \rightarrow S$  and  $t_g: S \rightarrow S'$ . We would then like

$$t_g \circ t_\alpha \circ (t_f \times id): T' \times R \rightarrow S' \quad \mapsto \quad \circ t_g \circ \bar{t}_\alpha \circ t_f: T' \rightarrow S'^R$$

where  $t_f \times id = \lambda d \in D.\langle t_f(\pi(d)), \pi'(d) \rangle$ . Thus we calculate:

$$\begin{aligned}
t_g \circ t_\alpha \circ (t_f \times id) &\mapsto \lambda d \in D.\Psi(\lambda d' \in D.(t_g \circ t_\alpha \circ (t_f \times id))(\langle d, d' \rangle)) \\
&= \lambda d \in D.\Psi(\lambda d' \in D.t_g(t_\alpha(\langle t_f(d), d' \rangle)))
\end{aligned}$$

and

$$\begin{aligned}
\circ t_g \circ \bar{t}_\alpha \circ t_f &= \lambda d \in D.\Psi(\lambda d' \in D.t_g(\Phi(\bar{t}_\alpha(t_f(d)))(d'))) \\
&= \lambda d \in D.\Psi(\lambda d' \in D.t_g(\Phi(\Psi(\lambda d'' \in D.t_\alpha(\langle t_f(d), d'' \rangle)))(d'))) \\
&= \lambda d \in D.\Psi(\lambda d' \in D.t_g(t_\alpha(\langle t_f(d), d' \rangle)))
\end{aligned}$$

□

**Theorem 20**  $\text{AdmPer}(D)$  is cartesian closed.

Proof: Immediate from Proposition 19. □

### 2.3.2 $\text{AdmPer}(D)_\perp$

Consider then the category  $\text{AdmPer}(D)_\perp$  of admissible pers and strict continuous functions:

Objects: same as for  $\text{AdmPer}(D)$ , admissible pers.

Morphisms: a morphism  $[f]: R \rightarrow S$  is an equivalence class in  $\mathcal{F}(R, S)_\perp / R \sim S$ .

**Theorem 21**  $\text{AdmPer}(D)_\perp$  is cartesian.

Proof: The objects are the same as those in  $\text{AdmPer}(D)$ , so for objects  $R$  and  $S$ ,  $R \times S$  is defined. We may assume  $\pi$  and  $\pi'$  strict, so their equivalence classes are genuine morphisms, and since we may assume  $\langle \cdot, \cdot \rangle$  strict,  $\lambda d \in D.\langle f(d), g(d) \rangle$  becomes strict, whenever  $f$  and  $g$  are strict. □

We define the tensor product as a quotient of the cartesian product. This looks somewhat awkward when written out, but we need a definition to calculate with...

**Definition 22** For  $R$  and  $S$  admissible pers, define

$$\begin{array}{c} \langle d_1, d_2 \rangle R \otimes S \langle d'_1, d'_2 \rangle \\ \Downarrow \\ \langle d_1, d_2 \rangle R \times S \langle d'_1, d'_2 \rangle \\ \vee \\ \left( \begin{array}{c} d_1 R d_1 \wedge d_2 S d_2 \wedge d'_1 R d'_1 \wedge d'_2 S d'_2 \wedge \\ (d_1 R \perp_D \vee d_2 S \perp_D) \wedge (d'_1 R \perp_D \vee d'_2 S \perp_D) \end{array} \right) \end{array}$$

**Lemma 23** If  $R$  and  $S$  are admissible pers, then so is  $R \otimes S$ .

*Proof:* Notice first that this is really just  $R \times S$ , with some of the equivalence classes merged, namely all those containing  $\perp_D$  in one of the coordinates. Thus, if  $R \times S$  is a per, then so is  $R \otimes S$ .

As for chain-completeness, let  $(\langle d_n, d'_n \rangle)_{n \in \mathbb{N}}$  and  $(\langle e_n, e'_n \rangle)_{n \in \mathbb{N}}$  be chains in  $D$  such that  $\forall n \in \mathbb{N}. \langle d_n, d'_n \rangle R \otimes S \langle e_n, e'_n \rangle$ . Then for all  $n \in \mathbb{N}$

$$\begin{array}{l} \text{either I) } d_n R e_n \wedge d'_n S e'_n \\ \text{or II) } (d_n R \perp_D \vee d'_n S \perp_D) \wedge (e_n R \perp_D \vee e'_n S \perp_D). \end{array}$$

So either I) holds from some step  $N$  or II) holds infinitely often.

In the first case  $\bigsqcup_{n \in \mathbb{N}} d_n R \bigsqcup_{n \in \mathbb{N}} e_n$  and  $\bigsqcup_{n \in \mathbb{N}} d'_n S \bigsqcup_{n \in \mathbb{N}} e'_n$  and since pairing is continuous we are done.

In the second case either  $d_n R \perp_D$  or  $d_n S \perp_D$  happens infinitely often. Assume it is  $d_n R \perp_D$ . Then  $(d_n)_{n \in \mathbb{N}}$  has a subchain  $(d_{n_r})_{r \in \mathbb{N}}$  such that  $\forall r \in \mathbb{N}. d_{n_r} R \perp_D$ . Thus  $\bigsqcup_{r \in \mathbb{N}} d_{n_r} R \perp_D$  and thus  $\bigsqcup_{n \in \mathbb{N}} d_n R \perp_D$ . Since all this also happens for  $(e_n)_{n \in \mathbb{N}}$  and  $(e'_n)_{n \in \mathbb{N}}$ , we have

$$\left( \bigsqcup_{n \in \mathbb{N}} d_n R \perp_D \wedge \bigsqcup_{n \in \mathbb{N}} d'_n S \perp_D \right) \vee \left( \bigsqcup_{n \in \mathbb{N}} e_n R \perp_D \wedge \bigsqcup_{n \in \mathbb{N}} e'_n S \perp_D \right)$$

giving  $\langle \bigsqcup_{n \in \mathbb{N}} d_n, \bigsqcup_{n \in \mathbb{N}} d'_n \rangle R \otimes S \langle \bigsqcup_{n \in \mathbb{N}} e_n, \bigsqcup_{n \in \mathbb{N}} e'_n \rangle$ . By continuity of pairing we are done.  $\square$

We now wish to show, that we actually have a functor.

**Lemma 24** The map  $(R, S) \mapsto R \otimes S$  extends to a functor.

*Proof:* The idea is to reuse the definition from cartesian products. Given morphisms  $f: R \rightarrow R'$  and  $g: S \rightarrow S'$  with strict realizers  $t_f$  and  $t_g$ , define  $f \otimes g: R \otimes S \rightarrow R' \otimes S'$  as the morphism tracked by

$$\lambda d \in D. \langle t_f(\pi(d)), t_g(\pi'(d)) \rangle$$

which is strict since pairing and projections are strict. Since it is “strict coordinate-wise” it respects equivalence.

This realizer is independent of the choices of  $t_f$  and  $t_g$ : Consider  $t'_f \sim_{R'} t_f$  and  $t'_g \sim_{S'} t_g$ . Then

$$\begin{aligned} \langle d, d' \rangle R \otimes S \langle d, d' \rangle &\Rightarrow d R d \wedge d' S d' \\ &\Rightarrow t_f(d) R' t'_f(d) \wedge t_g(d') S' t'_g(d') \\ &\Rightarrow \langle t_f(d), t_g(d') \rangle R' \otimes S' \langle t'_f(d), t'_g(d') \rangle \end{aligned}$$

so the morphism is well-defined. Identities are easily preserved, and composition also.

□

**Definition 25** Define the strongly admissible per  $I$  by

$$d I d' \Leftrightarrow d = d' = \perp_D \vee d = d' = \langle i, \perp_D \rangle$$

This definition is not taken out of the blue.  $I$  is actually in the image of a lifting functor to be defined later. Notice the “if construct” on  $I$ , which will be available on all lifted relations:

$$\begin{aligned} d I d &\Rightarrow d = \perp_D \vee d = \langle i, \perp_D \rangle \\ &\Rightarrow \pi(d) = \perp_D \vee \pi(d) = i \\ &\Rightarrow \Phi(\pi(d)) = \perp_{[D \rightarrow D]} \vee \Phi(\pi(d)) = id_{[D \rightarrow D]} \end{aligned}$$

Thus for  $d I d$ ,  $\Phi(\pi(d))(d')$  can be read as “if  $d \neq \perp_D$  then  $d'$  else  $\perp_D$ ”. We will use this to construct realizers.

**Lemma 26** The morphism  $\alpha_{R,S,T}: R \otimes (S \otimes T) \rightarrow (R \otimes S) \otimes T$  tracked by

$$\lambda d \in D. \langle \langle \pi(d), \pi(\pi'(d)) \rangle, \pi'(\pi'(d)) \rangle$$

defines a natural isomorphism.

**Proof:** Firstly, the realizer is strict, since pairing and projection is. As it is strict “coordinate-wise”, it preserves equivalence. The inverse map is tracked by

$$\lambda d \in D. \langle \pi(\pi(d)), \langle \pi'(\pi(d)), \pi'(d) \rangle \rangle$$

which is equally preserving equivalence, and obviously an inverse:

$$\langle d_1, \langle d_2, d_3 \rangle \rangle \mapsto \langle \langle d_1, d_2 \rangle, d_3 \rangle \mapsto \langle d_1, \langle d_2, d_3 \rangle \rangle$$

Naturality translates to the commutativity of

$$\begin{array}{ccccc} R & S & T & R \otimes (S \otimes T) & \xrightarrow{\alpha_{R,S,T}} & (R \otimes S) \otimes T \\ f \downarrow & g \downarrow & h \downarrow & f \otimes (g \otimes h) \downarrow & & \downarrow (f \otimes g) \otimes h \\ R' & S' & T' & R' \otimes (S' \otimes T') & \xrightarrow{\alpha_{R',S',T'}} & (R' \otimes S') \otimes T' \end{array}$$

which is clear, once one traces the values around both edges:

$$\langle d_1, \langle d_2, d_3 \rangle \rangle \mapsto \langle \langle f(d_1), g(d_2) \rangle, h(d_3) \rangle$$

□

**Lemma 27** *The morphism  $\rho_R: R \otimes I \rightarrow R$  tracked by*

$$\lambda d \in D. \Phi(\pi(\pi'(d)))(\pi(d))$$

*defines a natural isomorphism.*

**Proof:** The realizer reads “if the second component is not  $\perp_D$  return the first component, otherwise return  $\perp_D$ .” Thus the realizer preserves equivalence almost by definition. The inverse map is easily defined by

$$\lambda d \in D. \langle d, \langle i, \perp_D \rangle \rangle$$

Naturality is easiest shown for the inverse. Naturality of the inverse means that

$$\begin{array}{ccc} R & \xrightarrow{\rho_R} & R \otimes I \\ f \downarrow & & \downarrow f \otimes id_I \\ S & \xrightarrow{\rho_S} & S \otimes I \end{array}$$

commutes, which is seen by tracing values:

$$\begin{array}{ccc} d \vdash & \longrightarrow & \langle d, \langle i, \perp_D \rangle \rangle \\ \downarrow & & \downarrow \\ f(d) \vdash & \longrightarrow & \langle f(d), \langle i, \perp_D \rangle \rangle \end{array}$$

□

**Lemma 28** *The morphism  $\lambda_R: I \otimes R \rightarrow R$  tracked by*

$$\lambda d \in D. \Phi(\pi(\pi(d)))(\pi'(d))$$

*defines a natural isomorphism.*

**Proof:** See the proof of lemma 27. . . □

**Lemma 29** *The morphism  $\gamma_{R,S}: R \otimes S \rightarrow S \otimes R$  tracked by*

$$\lambda d \in D. \langle \pi'(d), \pi(d) \rangle$$

*defines a natural isomorphism.*

**Proof:** The proof is similar to those of the previous lemmas, the inverse realizer is easily defined, and preservation of equivalence is obvious. Naturality is clear when tracing values. □

**Lemma 30** *For all objects  $R, S, T$  and  $U$  the following holds:*

1.  $\alpha_{R \otimes S, T, U} \circ \alpha_{R, S, T \otimes U} = \alpha_{R, S, T} \otimes id_U \circ \alpha_{R, S \otimes T, U} \circ id_R \otimes \alpha_{S, T, U}$
2.  $id_R \otimes \lambda_S = \rho_R \otimes id_S \circ \alpha_{R, I, S}$
3.  $\gamma_{S, R} \circ \gamma_{R, S} = id$
4.  $\gamma_{R, I} \circ \lambda_R = \rho_R$
5.  $id_R \otimes \gamma_{S, T} \circ \alpha_{R, S, T} \circ \gamma_{R, T} \otimes id_S = \alpha_{R, S, T} \circ \gamma_{R \otimes S, T} \circ \alpha_{T, R, S}$

**Proof:** Tracing values on both sides of the equations, we get

1. rather quickly, that both sides simply rearranges the pairing:

$$\langle d_1, \langle d_2, \langle d_3, d_4 \rangle \rangle \rangle \mapsto \langle \langle \langle d_1, d_2 \rangle, d_3 \rangle, d_4 \rangle$$

2. two cases defined by the second coordinate. First case: left hand side

$$\langle d_1, \langle \perp_D, d_2 \rangle \rangle \mapsto \langle d_1, \perp_D \rangle$$

and right hand side

$$\langle d_1, \langle \perp_D, d_2 \rangle \rangle \mapsto \langle \langle d_1, \perp_D \rangle, d_2 \rangle \mapsto \langle \perp_D, d_2 \rangle$$

and second case: left hand side

$$\langle d_1, \langle \langle i, \perp_D \rangle, d_2 \rangle \rangle \mapsto \langle d_1, d_2 \rangle$$

and right hand side

$$\langle d_1, \langle \langle i, \perp_D \rangle, d_2 \rangle \rangle \mapsto \langle \langle d_1, \langle i, \perp_D \rangle \rangle, d_2 \rangle \mapsto \langle d_1, d_2 \rangle$$

when  $d_1 \notin [\perp_D]_R$  and  $d_2 \notin [\perp_D]_S$ .

3. no problems

4. right hand side

$$\langle d, \perp_D \rangle \mapsto \langle \perp_D, d \rangle \mapsto \perp_D$$

and left hand side

$$\langle d, \perp_D \rangle \mapsto \perp_D$$

when  $d \in [\perp_D]_R$ , and right hand side

$$\langle d, \langle i, \perp_D \rangle \rangle \mapsto \langle \langle i, \perp_D \rangle, d \rangle \mapsto d$$

and left hand side

$$\langle d, \langle i, \perp_D \rangle \rangle \mapsto d$$

when  $d \notin [\perp_D]_R$ .

5. the warm feeling, that we know how to rearrange pairs. . .

$$\langle d_1, \langle d_2, d_3 \rangle \rangle \mapsto \langle \langle d_3, d_1 \rangle, d_2 \rangle$$

□

**Proposition 31**  $\text{AdmPer}(D)_\perp$  is symmetric monoidal

*Proof:* It is monoidal by lemmas 26, 27, 28 and 30, and then symmetric monoidal by lemmas 29 and 30. □

We wish to define a closure operation, and use the idea from the cartesian closed category  $\text{AdmPer}(\cdot)(D)$ . In our current category, however, we only allow strict morphisms and we still want exponentials to be internal hom sets. So we only relate elements that code strict morphisms, i.e. that codes strictifiable realizers.

**Lemma 32** For all admissible pers  $R$  and  $S$ ,  $R \multimap S$  given by

$$d R \multimap S d' \Leftrightarrow d S^R d' \wedge (d'' R \perp_D \Rightarrow \Phi(d)(d'') S \perp_D S \Phi(d')(d''))$$

is also an admissible per.

*Proof:* As  $S^R$  is an equivalence relation,  $R \multimap S$  is readily a per. Let  $(x_n)_{n \in \mathbb{N}}$  and  $(y_n)_{n \in \mathbb{N}}$  be chains so that

$$\forall n \in \mathbb{N}. x_n R \multimap S y_n.$$

Then  $\bigsqcup_{n \in \mathbb{N}} x_i S^R \bigsqcup_{n \in \mathbb{N}} y_i$  as  $\forall n \in \mathbb{N}. x_n S^R y_n$ . We just have to show, that

$$d R \perp_D \Rightarrow \Phi\left(\bigsqcup_{n \in \mathbb{N}} x_n\right)(d) S \perp_D S \Phi\left(\bigsqcup_{n \in \mathbb{N}} y_n\right)(d)$$

But

$$\begin{aligned} d R \perp_D &\Rightarrow \forall n \in \mathbb{N}. \Phi(x_n)(d) S \perp_D S \Phi(y_n)(d) \\ &\Rightarrow \bigsqcup_{n \in \mathbb{N}} \Phi(x_n) S \perp_D S \bigsqcup_{n \in \mathbb{N}} \Phi(y_n) \\ &\Rightarrow \Phi\left(\bigsqcup_{n \in \mathbb{N}} x_n\right)(d) S \perp_D S \Phi\left(\bigsqcup_{n \in \mathbb{N}} y_n\right)(d). \end{aligned}$$

□

**Lemma 33** For all objects  $R$  of  $\text{AdmPer}(D)_\perp$ , the map  $R \multimap -$  extends to a functor  $\text{AdmPer}(D)_\perp \rightarrow \text{AdmPer}(D)_\perp$ .

*Proof:* For  $f: S \rightarrow U$  with strict realizer  $t_f$  define  $R \multimap f$  as the morphism tracked by formerly introduced  $ot_f$ .

We know, that  $ot_f \in \mathcal{F}(S^R, U^R)$ , and if for some  $d, d' R \perp_D \Rightarrow \Phi(d)(d') S \perp_D$ , then

$$\begin{aligned} d' R \perp_D &\Rightarrow \Phi(d)(d') S \perp_D \\ &\Rightarrow t_f(\Phi(d)(d')) U \perp_D \\ &\Rightarrow \lambda d'' \in D. t_f(\Phi(d)(d''))(d') U \perp_D \\ &\Rightarrow \Phi(ot_f(d))(d') U \perp_D \end{aligned}$$

so  $ot_f \in \mathcal{F}(R \multimap S, R \multimap U)$ , and since  $t_f$  is strict, so is  $ot_f$

$$ot_f(\perp_D) = \Psi(\lambda d' \in D. t_f(\Phi(\perp_D)(d'))) = \Psi(\lambda d' \in D. t_f(\perp_D)) = \Psi(\lambda d' \in D. \perp_D) = \perp_D,$$

so  $ot_f \in \mathcal{F}(R \multimap S, R \multimap U)_\perp$ . The morphism is independent of the choice of realizer; if  $t_f \sim_U t'_f$ :

$$\begin{aligned} d R \multimap S d &\Rightarrow d S^R d \wedge (d' R \perp_D \Rightarrow \Phi(d)(d') S \perp_D) \\ &\Rightarrow ot_f(d) U^R \circ t'_f(d) \wedge (d' R \perp_D \Rightarrow \Phi(ot_f(d))(d') U \perp_D U \Phi(ot'_f(d))(d')) \\ &\Rightarrow ot_f(d) R \multimap S \circ t'_f(d) \end{aligned}$$

And the operation is still functorial; for identities, we know

$$\begin{aligned} d R \multimap S d &\Rightarrow \circ id_D(d) S^R d \wedge (d' R \perp_D \Rightarrow \Phi(d)(d') S \perp_D) \\ &\Rightarrow \circ id_D(d) S^R d \wedge (d' R \perp_D \Rightarrow \Phi(\circ id_D(d))(d') = \Phi(d)(d') S \perp_D) \\ &\Rightarrow \circ id_D(d) R \multimap S d, \end{aligned}$$

and composition already works on the level of realizers.  $\square$

Now we aim to show, that  $\text{AdmPer}(D)_\perp$  is SMCC.

**Proposition 34** *For all objects  $R$ ,  $- \otimes R \dashv R \multimap -$ .*

*Proof:* For objects  $S$  and  $T$ , we consider  $\text{AdmPer}(D)_\perp(T \otimes R, S)$  and  $\text{AdmPer}(D)_\perp(T, R \multimap S)$ . A natural isomorphism  $\bar{\tau}$  between these is given by considering

$$\tau_f \in \mathcal{F}(T \otimes R, S)_\perp \mapsto \bar{\tau}_f = \lambda d \in D. \Psi(\lambda d' \in D. \tau_f(\langle d, d' \rangle))$$

which sends  $\tau_f$  to a strictifiable realizer

$$\begin{aligned} d T \perp_D &\Rightarrow (d' R d' \Rightarrow \tau_f(\langle d, d' \rangle) S \perp_D) \\ &\Rightarrow \Psi(\lambda d' \in D. \tau_f(\langle d, d' \rangle)) S^R \Psi(\lambda d' \in D. \perp_D) \\ &\Rightarrow \bar{\tau}_f(d) S^R \perp_D \end{aligned}$$

that produces codes of strictifiable functions

$$\begin{aligned} d T d &\Rightarrow (d' R \perp_D \Rightarrow \tau_f(\langle d, d' \rangle) S \perp_D) \\ &\Rightarrow (d' R \perp_D \Rightarrow \lambda d'' \in D. \tau_f(\langle d, d'' \rangle)(d') S \perp_D) \\ &\Rightarrow (d' R \perp_D \Rightarrow \Phi(\bar{\tau}_f(d))(d') S \perp_D) \end{aligned}$$

and whose strictification thus is in  $\mathcal{F}(T, R \multimap S)_\perp$

$$\begin{aligned} d_1 T d'_1 &\Rightarrow (d_2 R d'_2 \Rightarrow \langle d_1, d_2 \rangle T \times R \langle d'_1, d'_2 \rangle) \\ &\Rightarrow (d_2 R d'_2 \Rightarrow \tau_f(\langle d_1, d_2 \rangle) S \tau_f(\langle d'_1, d'_2 \rangle)) \\ &\Rightarrow \Psi(\lambda d' \in D. \tau_f(\langle d_1, d' \rangle)) S^R \Psi(\lambda d' \in D. \tau_f(\langle d'_1, d' \rangle)) \end{aligned}$$

and whose equivalence class under  $T \sim_{S^R}$  and thus under  $T \sim_{R \multimap S}$  we know is independent of the choice of realizers, so that it is well-defined. That it is a isomorphism is obvious, once one have found the inverse :

$$t_g \in \mathcal{F}(T, S^R) \mapsto \underline{t}_g = \lambda d \in D. \Phi(t_g(\pi(d)))(\pi'(d))$$

which is in  $\mathcal{F}(T \times R, S)$ , and also in  $\mathcal{F}(T \otimes R, S)_\perp$  as it is strict

$$\underline{t}_g(\perp_D) = \Phi(t_g(\pi(\perp_D)))(\pi'(\perp_D)) = \Phi(t_g(\perp_D))(\perp_D) = \Phi(\perp_D)(\perp_D) = \perp_D.$$

and whose equivalence class under  $T \times R \sim_S$  and thus under  $T \otimes R \sim_S$  is independent of choice of realizers, so that it is well-defined.

We know, that the two operations form a bijection, and that this bijection is natural even if non-strict functions were allowed.  $\square$

**Theorem 35**  $\text{AdmPer}(D)_\perp$  is symmetric monoidal closed

Proof: This is precisely the content of lemma 34.  $\square$

### 3 Lifting

Define the map  $L_0: \mathcal{P}(D) \rightarrow \mathcal{P}(D)$  by

$$L_0(A) = \{d \in D \mid \pi(\Phi(d)(i)) = i \wedge \pi'(\Phi(d)(i)) \in A\}.$$

for  $A \subseteq D$ . And then the map  $\mathcal{L}_0: \text{AdmPer}(D)_0 \rightarrow \text{AdmPer}(D)_{\perp 0}$  by

$$\mathcal{L}_0(R) = \{L(K) \mid K \in D/R\} \cup \{\{\perp_D\}\}.$$

Similarly define, for strongly admissible pers  $R$  and  $S$ , the map  $L_1: \mathcal{F}(R, S) \rightarrow \mathcal{F}(\mathcal{L}_0(R), \mathcal{L}_0(S))_\perp$  by

$$L_1(f) = \lambda d \in D. \Phi(\pi(\Phi(d)(i))) (\Psi(\lambda d' \in D. (i, f(\pi'(\Phi(d)(i)))))$$

And then the map  $\mathcal{L}_1: \text{AdmPer}(D)_1 \rightarrow \text{AdmPer}(D)_{\perp 1}$  by

$$\mathcal{L}_1(f) = [L_1(t_f)]_{\mathcal{L}_0(R) \sim \mathcal{L}_0(S)}$$

for  $f: R \rightarrow S$ .

**Lemma 36** The definitions of  $L_0$ ,  $\mathcal{L}_0$ ,  $L_1$  and  $\mathcal{L}_1$  make sense.

Proof: They do make sense in the listed order.

$L_0$ : If  $A$  is a subset of  $D$  then so is  $L_0(A)$ , quite trivially.

$\mathcal{L}_0$ : We first need to show that  $\mathcal{L}_0(R)$  is a per, i.e. that all the alleged equivalence classes are disjoint: First notice, that  $\pi(\Phi(\perp_D)(i)) = \perp_D \neq i$ , so  $\{\{\perp_D\}\}$  is disjoint from all the other subsets of  $D$ . Now assume  $d \in D$  and  $K, K' \in D/R$  so that  $d \in L_0(K)$  and  $d \in L_0(K')$ . Then  $\pi'(\Phi(d)(i)) \in K$  and  $\pi'(\Phi(d)(i)) \in K'$ , so that  $K = K'$ .

$d \in \mathcal{L}_0(R)$   $d, d' \in \mathcal{L}_0(R)$   $\perp_D$  and  $d \leq d'$  implies  $d = d' = \perp_D$  and thus  $d \in \mathcal{L}_0(R)$   $\perp_D$ .

Let  $(x_i)_{i \in I}$  and  $(y_i)_{i \in I}$  be chains in  $D$  such that

$$\forall i \in I. x_i \mathcal{L}(R) y_i$$

Then

$$(\pi'(\Phi(x_i)(i)))_{i \in I} \quad \text{and} \quad (\pi'(\Phi(y_i)(i)))_{i \in I}$$

are chains in  $R$ , and by continuity of  $\Phi$ , application and  $\pi'$

$$\bigsqcup_{i \in I} \pi'(\Phi(x_i)(i)) = \pi'(\Phi(\bigsqcup_{i \in I} x_i)(i)) \quad \text{and} \quad \bigsqcup_{i \in I} \pi'(\Phi(y_i)(i)) = \pi'(\Phi(\bigsqcup_{i \in I} y_i)(i))$$

As  $R$  is chain complete, there is then a  $K \in D/R$ , containing both  $\pi'(\Phi(\bigsqcup_{i \in I} x_i)(i))$  and  $\pi'(\Phi(\bigsqcup_{i \in I} y_i)(i))$ . Similarly, by continuity of  $\Phi$ , application and  $\pi$ ,  $\pi(\Phi(\bigsqcup_{i \in I} x_i)(i)) = \pi(\Phi(\bigsqcup_{i \in I} y_i)(i)) = i$ , and so  $\mathcal{L}_0(K)$  contains both  $\bigsqcup_{i \in I} x_i$  and  $\bigsqcup_{i \in I} y_i$ .

$L_1$ : Observe that for all  $d \in D$

$$\pi(\Phi(\Psi(\lambda d' \in D.\langle i, d \rangle)(i))) = i$$

and

$$\pi'(\Phi(\Psi(\lambda d' \in D.\langle i, d \rangle)(i))) = d$$

so that if  $d \in A \subseteq D$  then  $\Psi(\lambda d' \in D.\langle i, d \rangle) \in L_0(A)$ . With this in mind, we can argue that if  $f \in \mathcal{F}(R, S)$  then

$$\begin{aligned} \exists K \in D/R. d \in K \wedge d' \in K &\Rightarrow \exists K' \in D/S. f(d) \in K' \wedge f(d') \in K' \\ &\Rightarrow \exists K' \in D/S. \Psi(\lambda d'' \in D.\langle i, f(d) \rangle) \in L_0(K') \quad \wedge \\ &\quad \Psi(\lambda d'' \in D.\langle i, f(d') \rangle) \in L_0(K') \\ &\Rightarrow \Psi(\lambda d'' \in D.\langle i, f(d) \rangle) \mathcal{L}_0(S) \Psi(\lambda d'' \in D.\langle i, f(d') \rangle) \end{aligned}$$

and thus

$$\begin{aligned} d \mathcal{L}_0(R) d' &\Rightarrow \exists K \in D/R. d \in L_0(K) \wedge d' \in L_0(K) \quad \vee \\ &\quad d = d' = \perp_D \\ &\Rightarrow \exists K \in D/R. \pi'(\Phi(d)(i)) \in K \wedge \pi'(\Phi(d')(i)) \in K \quad \vee \\ &\quad d = d' = \perp_D \\ &\Rightarrow L_1(f)(d) \mathcal{L}_0(S) L_1(f)(d') \quad \vee \\ &\quad L_1(f)(d) = L_1(f)(d') = \perp_D \\ &\Rightarrow L_1(f)(d) \mathcal{L}_0(S) L_1(f)(d') \end{aligned}$$

so that  $L_1(f) \in \mathcal{F}(\mathcal{L}_0(R), \mathcal{L}_0(S))$ . And

$$L_1(f)(\perp_D) = \Phi(\pi(\Phi(\perp_D)(i)))(\Psi(\lambda d' \in D.\langle i, f(\perp_D) \rangle)) = \Phi(\perp_D)(\Psi(\lambda d' \in D.\langle i, f(\perp_D) \rangle)) = \perp_D$$

so that  $L_1(f) \in \mathcal{F}(\mathcal{L}_0(R), \mathcal{L}_0(S))_{\perp}$ .

$\mathcal{L}_1$ : Consider  $f \in \text{AdmPer}(D)(R, S)$ . Taking two realizers  $t_f$  and  $t'_f$  for  $f$ , we find firstly that

$$\begin{aligned} \exists K \in D/R. d \in K &\Rightarrow \exists K' \in D/S. t_f(d) \in K' \wedge t'_f(d) \in K' \\ &\Rightarrow \exists K' \in D/S. \Psi(\lambda d' \in D.\langle i, t_f(d) \rangle) \in L_0(K') \quad \wedge \\ &\quad \Psi(\lambda d' \in D.\langle i, t'_f(d) \rangle) \in L_0(K') \\ &\Rightarrow \Psi(\lambda d' \in D.\langle i, t_f(d) \rangle) \mathcal{L}_0(S) \Psi(\lambda d' \in D.\langle i, t'_f(d) \rangle) \end{aligned}$$

and thus

$$\begin{aligned}
d \mathcal{L}_0(R) d &\Rightarrow \exists K \in D/R. d \in K \quad \vee \\
&\quad d = \perp_D \\
&\Rightarrow \exists K \in D/R. \pi'(\Phi(d)(i)) \in K \quad \vee \\
&\quad d = \perp_D \\
&\Rightarrow L_1(t_f)(d) \mathcal{L}_0(S) L_1(t'_f)(d) \quad \vee \\
&\quad L_1(t_f)(d) = L_1(t'_f)(d) = \perp_D \\
&\Rightarrow L_1(t_f)(d) \mathcal{L}_0(S) L_1(t'_f)(d)
\end{aligned}$$

so that  $L_1(t_f) \mathcal{L}_0(R) \sim_{\mathcal{L}_0(S)} \mathcal{L}_1(t'_f)$ .

□

Notice the “if construct” available on a lifted relation: If  $R$  is an admissible per then

$$\begin{aligned}
d \mathcal{L}(R) d &\Rightarrow d = \perp_D \quad \vee \quad \pi(\Phi(d)(i)) = i \\
&\Rightarrow \pi(\Phi(d)(i)) = \perp_D \quad \vee \quad \pi(\Phi(d)(i)) = i \\
&\Rightarrow \Phi(\pi(\Phi(d)(i))) = \perp_{[D \rightarrow D]} \quad \vee \quad \Phi(\pi(\Phi(d)(i))) = id_{[D \rightarrow D]}
\end{aligned}$$

Thus  $\Phi(\pi(\Phi(d)(i)))(d')$  can be read “if  $d \notin \perp_{\mathcal{L}(R)}$  then  $d'$  else  $\perp_D$ ”, where  $\perp_D$  of course lies in the bottom class of any admissible per.

We also has a “lift” and an “unlift”: If  $d \in A$  then  $\Psi(\lambda d' \in D. \langle i, d \rangle) \in \mathcal{L}(A)$  and if  $d \in \mathcal{L}(A)$  then  $\pi'(\Phi(d)(i)) \in A$ .

This is convenient for constructing realizers. We here used  $\mathcal{L}$  without a subscript because

**Lemma 37**  $(\mathcal{L}_0, \mathcal{L}_1) = \mathcal{L}: \text{AdmPer}(D) \rightarrow \text{AdmPer}(D)_\perp$  is a functor

**Proof:** We only need to show proper behavior with respect to identities and composition. Identities are tracked by  $id_D$ , and

$$L_1(id_D) = \lambda d \in D. \Phi(\pi(\Phi(d)(i)))(\Psi(\lambda d' \in D. \langle i, \pi'(\Phi(d)(i)) \rangle))$$

Assuming  $d \mathcal{L}_0(R) d$  we either have  $d = \perp_D$ , in which case  $L_1(id_D)$  being strict ensures  $L_1(id_D)(d) \mathcal{L}_0(R) d$ , or there is a  $K \in D/R$ , with  $d \in L_0(K)$ . For this case, we calculate

$$\begin{aligned}
d \in L_0(K) &\Rightarrow \pi'(\Phi(d)(i)) \in K \wedge \pi(\Phi(d)(i)) = i \\
&\Rightarrow \pi'(\Phi(d)(i)) \in K \wedge \\
&\quad L_1(id_D)(d) = \Phi(i)(\Psi(\lambda d' \in D. \langle i, \pi'(\Phi(d)(i)) \rangle)) \\
&\Rightarrow \pi'(\Phi(d)(i)) \in K \wedge \\
&\quad L_1(id_D)(d) = \Psi(\lambda d' \in D. \langle i, \pi'(\Phi(d)(i)) \rangle) \\
&\Rightarrow \pi'(\Phi(d)(i)) \in K \wedge \\
&\quad \pi(\Phi(L_1(id_D)(d))(i)) = i \wedge \\
&\quad \pi'(\Phi(L_1(id_D)(d))(i)) = \pi'(\Phi(d)(i)) \\
&\Rightarrow \pi(\Phi(L_1(id_D)(d))(i)) = i \wedge \\
&\quad \pi'(\Phi(L_1(id_D)(d))(i)) \in K \\
&\Rightarrow L_1(id_D)(d) \in L_0(K)
\end{aligned}$$

so  $L_1(id_D) \sim_{L_0(K)} id_D$ .

To address composition, consider two composable morphisms  $f: S \rightarrow T$  and  $g: R \rightarrow S$ , with realizers  $t_f$  and  $t_g$ :

$$\begin{aligned}
& K \in D/R \wedge d \in L_0(K) \quad \Rightarrow \\
L_1(t_f)(L_1(t_g)(d)) &= L_1(t_f)(\Phi(\pi(\Phi(d)(i)))(\Psi(\lambda d' \in D.\langle i, t_g(\pi'(\Phi(d)(i))) \rangle))) \\
&= L_1(t_f)(\Psi(\lambda d' \in D.\langle i, t_g(\pi'(\Phi(d)(i))) \rangle)) \\
&= \Phi(i)(\Psi(\lambda d' \in D.\langle i, t_f(t_g(\pi'(\Phi(d)(i))) \rangle))) \\
&= \Psi(\lambda d' \in D.\langle i, t_f \circ t_g(\pi'(\Phi(d)(i))) \rangle) \\
&\in L_0(t_f \circ t_g(K))
\end{aligned}$$

and of course

$$K \in D/R \wedge d \in L_0(K) \Rightarrow L_1(t_f)(L_1(t_g)(d)) \in L_0(t_f \circ t_g(K))$$

so that the two maps are equivalent.  $\square$

There is an obvious forgetful functor  $\mathcal{U}: \mathbf{AdmPer}(D)_\perp \rightarrow \mathbf{AdmPer}(D)$

**Lemma 38** *The morphisms  $(\eta_R: R \rightarrow \mathcal{U}\mathcal{L}(R))_{R \in \mathbf{AdmPer}D_0}$  all tracked by*

$$t_\eta = \lambda d \in D. \Psi(\lambda d' \in D. \langle i, d \rangle)$$

*form a natural transformation  $\eta: id_{\mathbf{AdmPer}(D)} \Rightarrow \mathcal{U}\mathcal{L}$ .*

**Proof:** First we must show, that for all  $R$ ,  $t_\eta \in \mathcal{F}(R, \mathcal{U}\mathcal{L}(R))$ : If  $d \in A \subseteq D$ , then  $t_\eta(d) = \Psi(\lambda d' \in D. \langle i, d \rangle)$ , and

$$\pi(\Phi(t_\eta(d))(i)) = \pi((\lambda d' \in D. \langle i, d \rangle)(i)) = \pi(\langle i, d \rangle) = i$$

$$\pi'(\Phi(t_\eta(d))(i)) = \pi'((\lambda d' \in D. \langle i, d \rangle)(i)) = \pi'(\langle i, d \rangle) = d \in A$$

thus  $t_\eta(d) \in L(A)$ . We then calculate

$$\begin{aligned}
d R d' &\Rightarrow \exists K \in D/R. d \in K \wedge d' \in K \\
&\Rightarrow \exists K \in D/R. t_\eta(d) \in L(K) \wedge t_\eta(d') \in L(K) \\
&\Rightarrow d \mathcal{L}(R) d'.
\end{aligned}$$

For  $\eta$  to be a natural transformation, we need

$$\begin{array}{ccc}
R & R & \xrightarrow{\eta_R} \mathcal{U}\mathcal{L}(R) \\
f \downarrow & f \downarrow & \downarrow \mathcal{U}\mathcal{L}(f) \\
S & S & \xrightarrow{\eta_S} \mathcal{U}\mathcal{L}(S)
\end{array}$$

to commute for all  $f: R \rightarrow S$ . Assuming a realizer  $t_f$  of  $f$ , we calculate

$$\begin{aligned}
L(t_f)(t_\eta(d)) &= \Phi(\pi(\Phi(t_\eta(d))(i)))(\Psi(\lambda d' \in D. \langle i, t_f(\pi'(\Phi(t_\eta(d))(i))) \rangle))) \\
&= \Phi(i)(\Psi(\lambda d' \in D. \langle i, t_f(d) \rangle)) \\
&= \Psi(\lambda d' \in D. \langle i, t_f(d) \rangle) \\
&= t_\eta(t_f(d))
\end{aligned}$$

$\square$

**Proposition 39**  $\mathcal{L} \dashv \mathcal{U}$

**Proof:** According to [6, p.83] we need only to define the unit  $\eta: id_{\text{AdmPer}(D)} \Rightarrow \mathcal{U}\mathcal{L}$ , and find for each  $f: R \rightarrow \mathcal{U}(S)$  in  $\text{AdmPer}(D)_\perp$  a unique  $h: \mathcal{L}(R) \rightarrow S$  in  $\text{AdmPer}(D)_\perp$ , such that  $\mathcal{U}(h) \circ \eta_R = f$ :

$$\begin{array}{ccc}
 R & \xrightarrow{\eta_R} & \mathcal{U}\mathcal{L}(R) & & \mathcal{L}(R) \\
 & \searrow f & \downarrow \mathcal{U}(h) & & \downarrow h \\
 & & \mathcal{U}(S) & & S
 \end{array}$$

We have  $\eta$  by lemma 38, if we can produce a unique  $h$ . Let  $f: R \rightarrow \mathcal{U}(S)$  be given. Let  $t_f$  be a realizer for  $f$ . Define  $h$  through

$$t_h = \lambda d \in D. \Phi(\pi(\Phi(d)(i)))(t_f(\pi'(\Phi(d)(i))))$$

which is strict

$$t_h(\perp_D) = \Phi(\pi(\Phi(\perp_D)(i)))(t_f(\pi'(\Phi(\perp_D)(i)))) = \Phi(\perp_D)(t_f(\pi'(\Phi(\perp_D)(i)))) = \perp_D$$

and in  $\mathcal{F}(\mathcal{L}(R), S)_\perp$

$$\begin{aligned}
 d \mathcal{L}(R) d' &\Rightarrow (\exists K \in D/R. d \in L(K) \wedge d' \in L(K)) \vee \\
 &\quad d = d' = \perp_D \\
 &\Rightarrow (\pi(\Phi(d)(i)) = \pi(\Phi(d')(i)) = i \wedge \\
 &\quad \exists K \in D/R. \pi'(\Phi(d)(i)) \in K \wedge \pi'(\Phi(d')(i)) \in K) \vee \\
 &\quad d = d' = \perp_D \\
 &\Rightarrow (\Phi(\pi(\Phi(d)(i))) = \Phi(\pi(\Phi(d')(i))) = id_D \wedge \\
 &\quad \exists K \in D/\mathcal{U}(S). t_f(\pi'(\Phi(d)(i))) \in K \wedge t_f(\pi'(\Phi(d')(i))) \in K) \vee \\
 &\quad \Phi(\pi(\Phi(d)(i))) = \Phi(\pi(\Phi(d')(i))) = \lambda d \in D. \perp_D \\
 &\Rightarrow (\exists K \in D/\mathcal{U}(S). t_h(d) \in K \wedge t_h(d') \in K) \vee \\
 &\quad t_h(d) = t_h(d') = \perp_D \\
 &\Rightarrow t_h(d) \mathcal{U}(S) t_h(d')
 \end{aligned}$$

and independent of the choice of realizer for  $f$ : Let  $t_f \mathcal{L}(R) \sim_S t'_f$  and build  $t'_h$  from  $t'_f$  as  $t_h$  was build from  $t_f$ , then

$$\begin{aligned}
 d \mathcal{L}(R) d &\Rightarrow \exists K \in D/R. d \in L(K) \vee d = \perp_D \\
 &\Rightarrow (\pi(\Phi(d)(i)) = i \wedge \exists K \in D/R. \pi'(\Phi(d)(i)) \in K) \vee d = \perp_D \\
 &\Rightarrow (\Phi(\pi(\Phi(d)(i))) = id_D \wedge \\
 &\quad \exists K \in D/\mathcal{U}(S). t_f(\pi'(\Phi(d)(i))) \in K \wedge t'_f(\pi'(\Phi(d)(i))) \in K) \vee \\
 &\quad \Phi(\pi(\Phi(d)(i))) = \lambda d \in D. \perp_D \\
 &\Rightarrow (\exists K \in D/\mathcal{U}(S). t_h(d) \in K \wedge t'_h(d) \in K) \vee \\
 &\quad t_h(d) = t'_h(d) = \perp_D \\
 &\Rightarrow t_h(d) \mathcal{U}(S) t'_h(d)
 \end{aligned}$$

Now  $\mathcal{U}(h) \circ \eta_R = f$ :

$$t_h(t_\eta(d)) = \Phi(\pi(\Phi(t_\eta(d)(i)))(t_f(\pi'(\Phi(t_\eta(d)(i))))) = \Phi(i)(t_f(d)) = t_f(d)$$

and  $h$  is unique with this property: First observe that  $\eta_R$  almost is surjective, in the sense that

$$\forall K \in D/R. \exists K' \in D/R. \eta_R(K') = L(K)$$

since  $\eta_R(K) = L(K)$ .

Now assume  $h': \mathcal{L}(R) \rightarrow S$  so that  $\mathcal{U}(h') \circ \eta_R = f$ . Then

$$\begin{aligned} d \mathcal{L}(R) d &\Rightarrow \exists K \in D/R. d \in L(K) \quad \vee \quad d = \perp_D \\ &\Rightarrow \exists K \in D/R. d \in \eta_R(K) \quad \vee \quad d = \perp_D \\ &\Rightarrow \mathcal{U}(h')([d]_{\mathcal{L}(R)}) = f([d]_{\mathcal{L}(R)}) \quad \vee \quad \mathcal{U}(h')([d]_{\mathcal{L}(R)}) = \perp_D \\ &\Rightarrow \mathcal{U}(h')([d]_{\mathcal{L}(R)}) = \mathcal{U}(h)([d]_{\mathcal{L}(R)}) \end{aligned}$$

□

**Theorem 40**  $\mathcal{L} \dashv \mathcal{U}$  is a monoidal adjunction

*Proof:* According to [8, p.6] it is enough to show, that  $\mathcal{L} \dashv \mathcal{U}$ , and that  $\mathcal{L}$  is a strong symmetric monoidal functor. By proposition 39 we have  $\mathcal{L} \dashv \mathcal{U}$ , and we now show, that  $\mathcal{L}$  is a strong symmetric monoidal functor.

We must provide an isomorphism  $m_i: I \rightarrow \mathcal{L}(\mathbb{1})$  and a natural isomorphism  $m_{R,S}: \mathcal{L}(R) \otimes \mathcal{L}(S) \rightarrow \mathcal{L}(R \times S)$ .

$m_i: I \rightarrow \mathcal{L}(\mathbb{1})$ : Recall, that  $\mathbb{1}$  is  $\{\{\perp_D\}\}$ , and thus  $\mathcal{L}(\mathbb{1})$  is

$$\{\{\perp_D\}, \{d \in D \mid \pi(\Phi(d)(i)) = i \wedge \pi'(\Phi(d)(i)) = \perp_D\}\}$$

whereas  $I$  is

$$\{\{\perp_D\}, \{i, \perp_D\}\}$$

It is obvious, what the (strict) isomorphism must do, all we need is a realizer such as

$$\lambda d \in D. \Psi(\lambda d' \in D. \langle d \rangle)$$

and a realizer for the inverse, such as

$$\lambda d \in D. \langle \pi(\Phi(d)(i)), \pi'(\Phi(d)(i)) \rangle$$

$m_{R,S}: \mathcal{L}(R) \otimes \mathcal{L}(S) \rightarrow \mathcal{L}(R \times S)$ : Again it is obvious what the morphism should do, but we must define the realizers and check naturality.

For all strongly admissible pers  $R$  and  $S$ , let  $m_{R,S}$  be the morphism tracked by

$$\lambda d \in D. \Phi(\pi(\Phi(\pi(d)(i)))\Phi(\pi(\Phi(\pi'(d)(i))))(\Psi(\lambda d' \in D. \langle i, \langle \pi'(\Phi(\pi(d)(i))), \pi'(\Phi(\pi'(d)(i)) \rangle \rangle \rangle)))$$

which reads “if  $\pi(d) \neq \perp$  then if  $\pi'(d) \neq \perp$  then lift of  $\langle \text{unlift}(\pi(d)), \text{unlift}(\pi'(d)) \rangle$ ”. Thus the idea is the following:

$$\begin{array}{ccccc}
 & & L_0(U) \xrightarrow{\text{unlift}} U & & \\
 & \nearrow \pi & & \searrow & \\
 \langle L_0(U), L_0(V) \rangle & & & & \langle U, V \rangle \xrightarrow{\text{lift}} L_0(\langle U, V \rangle) \\
 & \searrow \pi' & & \nearrow & \\
 & & L_0(V) \xrightarrow{\text{unlift}} V & & 
 \end{array}$$

Following a similar chain of thought, the inverse is tracked by

$$\lambda d \in D. \Phi(\pi(\Phi(d)(i))) (\langle \Psi(\lambda d' \in D. \langle i, \pi(\pi'(\Phi(d)(i)))) \rangle, \Psi(\lambda d' \in D. \langle i, \pi'(\pi'(\Phi(d)(i)))) \rangle)$$

Here the idea is

$$\begin{array}{ccccc}
 & & U \xrightarrow{\text{lift}} L_0(U) & & \\
 & \nearrow \pi & & \searrow & \\
 L_0(\langle U, V \rangle) \xrightarrow{\text{unlift}} \langle U, V \rangle & & & & \langle L_0(U), L_0(V) \rangle \\
 & \searrow \pi' & & \nearrow & \\
 & & V \xrightarrow{\text{lift}} L_0(V) & & 
 \end{array}$$

Once we view this isomorphism as a switch between two isomorphic encodings, and realize, that the two functors  $\mathcal{L}(R) \otimes \mathcal{L}(S)$  and  $\mathcal{L}(R \times S)$  simply perform these encodings and otherwise preserve the operation of a given morphism, naturality becomes obvious.

□

## 4 Going fibred

As i [1, 3.3] define split indexed categories, and apply the Grothendieck construction:

The contravariant functor  $P : SET \rightarrow Cat$  gives on

sets: For  $I$  set,  $P(I)$  is the category with

objects:  $(R_i)_{i \in I}$  where for all  $i \in I$ ,  $R_i$  is an object of  $\text{AdmPer}(D)$ .

morphisms:  $(\alpha_i)_{i \in I} : (R_i)_{i \in I} \rightarrow (S_i)_{i \in I}$  where for all  $i \in I$ ,  $\alpha_i \in \text{AdmPer}(D)(R_i, S_i)$   
and  $\exists \alpha \in [D \rightarrow D]. \forall i \in I. \alpha_i = [\alpha]_{R_i \sim S_i}$ .

functions: For  $f: I \rightarrow J$  a function,  $P(f)$  is the functor from  $P(J)$  to  $P(I)$  with values on

$$\begin{aligned} \text{objects: } P(f)((R_j)_{j \in J}) &= (R_{f(i)})_{i \in I} \\ \text{morphisms: } P(f)((\alpha_j)_{j \in J}) &= (\alpha_{f(i)})_{i \in I} \end{aligned}$$

The contravariant functor  $Q: SET \rightarrow Cat$  gives on

sets: For  $I$  set,  $Q(I)$  is the category with

$$\begin{aligned} \text{objects: } (R_i)_{i \in I} \text{ where for all } i \in I, R_i \text{ is an object of } \text{AdmPer}(D)_\perp. \\ \text{morphisms: } (\alpha_i)_{i \in I}: (R_i)_{i \in I} \rightarrow (S_i)_{i \in I} \text{ where for all } i \in I, \alpha_i \in \text{AdmPer}(D)_\perp(R_i, S_i) \\ \text{and } \exists \alpha \in [D \rightarrow D]. \forall i \in I. \alpha_i = [\alpha]_{R_i \sim S_i}. \end{aligned}$$

functions: For  $f: I \rightarrow J$  a function,  $Q(f)$  is the functor from  $Q(J)$  to  $P(I)$  with values on

$$\begin{aligned} \text{objects: } Q(f)((R_j)_{j \in J}) &= (R_{f(i)})_{i \in I} \\ \text{morphisms: } Q(f)((\alpha_j)_{j \in J}) &= (\alpha_{f(i)})_{i \in I} \end{aligned}$$

That we have two contravariant functors is obvious. The Grothendieck construction then gives us two split fibrations,  $UFam(\text{AdmPer}(D))$  and  $UFam(\text{AdmPer}(D)_\perp)$ . We now wish to view  $\mathcal{L}$  and  $\mathcal{U}$  as split fibred functors forming a split fibred adjunction. We thus define two new functors and wildly abuse notation, by naming them  $\mathcal{L}$  and  $\mathcal{U}$ :

The functor  $\mathcal{L}: UFam(\text{AdmPer}(D)) \rightarrow UFam(\text{AdmPer}(D)_\perp)$  is defined on

$$\begin{aligned} \text{objects: } \mathcal{L}(I, (R_i)_{i \in I}) &= (I, (\mathcal{L}(R_i))_{i \in I}) \\ \text{morphisms: } \mathcal{L}(I, (f_i)_{i \in I}) &= (I, (\mathcal{L}(f_i))_{i \in I}) \end{aligned}$$

The functor  $\mathcal{U}: UFam(\text{AdmPer}(D)_\perp) \rightarrow UFam(\text{AdmPer}(D))$  is defined on

$$\begin{aligned} \text{objects: } \mathcal{U}(I, (R_i)_{i \in I}) &= (I, (\mathcal{U}(R_i))_{i \in I}) \\ \text{morphisms: } \mathcal{U}(I, (f_i)_{i \in I}) &= (I, (\mathcal{U}(f_i))_{i \in I}) \end{aligned}$$

These are not recursive definitions, they simply look so due to name clashes.

**Theorem 41**  $\mathcal{L}$  and  $\mathcal{U}$  are split fibred functors

**Proof:** Clearly fibers are sent to fibers by both functors. To see that they commute with reindexing we calculate: Take  $f: I \rightarrow J$  in  $Set$ . Then for  $(R_j)_{j \in J}$  in  $UFam(\text{AdmPer}(D))$

$$\mathcal{L}(f^*((R_j)_{j \in J})) = \mathcal{L}((R_{f(i)})_{i \in I}) = (\mathcal{L}(R_{f(i)}))_{i \in I} = f^*((\mathcal{L}(R_j))_{j \in J}) = f^*(\mathcal{L}((R_j)_{j \in J}))$$

and for  $(S_j)_{j \in J}$  in  $UFam(\text{AdmPer}(D)_\perp)$

$$\mathcal{U}(f^*((S_j)_{j \in J})) = \mathcal{U}((S_{f(i)})_{i \in I}) = (\mathcal{U}(S_{f(i)}))_{i \in I} = f^*((\mathcal{U}(S_j))_{j \in J}) = f^*(\mathcal{U}((S_j)_{j \in J}))$$

where hopefully the name clashes are not too confusing.  $\square$

**Theorem 42**  $\mathcal{L} \dashv \mathcal{U}$  is a split fibred strong monoidal adjunction

Proof: Immediate from theorems 40 and 41  $\square$

## 5 PILL

We now have the picture, we are looking for:

$$\begin{array}{ccc}
 UFam(\text{AdmPer}(D)_\perp) & \begin{array}{c} \xleftarrow{\mathcal{L}} \\ \perp \\ \xrightarrow{\mathcal{U}} \end{array} & UFam(\text{AdmPer}(D)) \\
 \searrow q & & \swarrow p \\
 & SET & 
 \end{array}$$

What remains to check is essentially the parts, that make  $p$  a  $\lambda$ -2 fibration.

**Lemma 43** The set  $\Omega = \text{Obj}(\text{AdmPer}(D)_\perp) = \text{Obj}(\text{AdmPer}(D))$  is a split generic object of the fibration  $q$ .

Proof: This is obvious. Taking a morphism in the base category from any object  $A$  to  $\Omega$ , we simply get the object  $(R_a)_{a \in A}$  in the fiber  $UFam(\text{AdmPer}(D)_\perp)_A$ . The two sets  $\text{Obj}(UFam(\text{AdmPer}(D)_\perp)_A)$  and  $\text{Set}(A, \Omega)$  are not merely isomorphic, they are identical.  $\square$

**Lemma 44** If for all  $i \in I$ ,  $R_i$  is a strongly admissible per, then so is  $\bigcup_{i \in I} R_i$  defined by

$$d \bigcup_{i \in I} R_i d' \Leftrightarrow \forall i \in I. d R_i d'$$

Proof: That it is a per is obvious. Let  $(x_j)_{j \in J}$  and  $(y_j)_{j \in J}$  be chains in  $D$ , such that  $\forall j \in J. x_j \bigcup_{i \in I} R_i y_j$ . Then

$$\begin{aligned}
 \forall j \in J. x_j \bigcup_{i \in I} R_i y_j &\Rightarrow \forall i \in I. \forall j \in J. x_j R_i y_j \\
 &\Rightarrow \forall i \in I. \bigsqcup_{j \in J} x_j R_i \bigsqcup_{j \in J} y_j \\
 &\Rightarrow \bigsqcup_{j \in J} x_j \bigcup_{i \in I} R_i \bigsqcup_{j \in J} y_j
 \end{aligned}$$

And  $\forall i \in I. \perp_D R_i \perp_D \Rightarrow \perp_D \bigcup_{i \in I} R_i \perp_D$ .

Now take  $d \bigcup_{i \in I} R_i d$  and  $d' \bigcup_{i \in I} R_i \perp_D$  such that  $d \leq d'$ . Then

$$\begin{aligned}
 &d \bigcup_{i \in I} R_i d \wedge d' \bigcup_{i \in I} R_i \perp_D \wedge d \leq d' \\
 \Rightarrow &\forall i \in I. (d R_i d \wedge d' R_i \perp_D \wedge d \leq d') \\
 \Rightarrow &\forall i \in I. d R_i \perp_D \\
 \Rightarrow &d \bigcup_{i \in I} R_i \perp_D
 \end{aligned}$$

□

**Lemma 45** *If for all  $i \in I$ ,  $R_i$  and  $S_i$  are strongly admissible pers and  $\alpha \in [D \rightarrow D]$ , then*

$$\forall i \in I. \alpha \in \mathcal{F}(R_i, S_i) \quad \Rightarrow \quad \alpha \in \mathcal{F}\left(\bigcup_{i \in I} R_i, \bigcup_{i \in I} S_i\right)$$

**Proof:** We simply calculate:

$$\begin{aligned} d \bigcup_{i \in I} R_i d' &\Rightarrow \forall i \in I. d R_i d' \\ &\Rightarrow \forall i \in I. \alpha(d) S_i \alpha(d') \\ &\Rightarrow \alpha(d) \bigcup_{i \in I} S_i \alpha(d') \end{aligned}$$

□

**Proposition 46** *The fibration  $q$  has simple split  $\Omega$ -products satisfying the Beck-Chevalley condition.*

**Proof:** Given any projection  $\pi_A: A \times \Omega \rightarrow A$  in *Set*, we must exhibit a right adjoint  $\forall_A$  to  $\pi_A^*$  and then show that the collection of these satisfy the Beck-Chevalley condition.

### Simple $\Omega$ -products

Let  $\pi_A: A \times \Omega \rightarrow A$  in *Set* be given. Construct the functor

$$\forall_A: UFam(\text{AdmPer}(D)_\perp)_{A \times \Omega} \rightarrow UFam(\text{AdmPer}(D)_\perp)_A$$

as follows, on

Objects: Form the intersection:

$$\forall_A((R_{(a,\omega)})_{(a,\omega) \in A \times \Omega}) = \left( \bigcup_{\omega \in \Omega} R_{(a,\omega)} \right)_{a \in A}$$

Morphisms: Simply take the uniform realizer:

$$\forall_A([\alpha]_{R_{(a,\omega)} \sim S_{(a,\omega)}})_{(a,\omega) \in A \times \Omega} = ([\alpha]_{\bigcup_{\omega \in \Omega} R_{(a,\omega)} \sim \bigcup_{\omega \in \Omega} S_{(a,\omega)}})_{a \in A}$$

This definition makes sense due to lemmas 44 and 45. Surely this map preserves identities and composition, so it is a functor. To see, that it is right adjoint to  $\pi_A^*$ , we consider for objects  $(R_a)_{a \in A}$  and  $(S_{(a,\omega)})_{(a,\omega) \in A \times \Omega}$  the hom-sets

$$UFam(\text{AdmPer}(D)_\perp)_{A \times \Omega}(\pi_A^*((R_a)_{a \in A}), (S_{(a,\omega)})_{(a,\omega) \in A \times \Omega})$$

and

$$UFam(\text{AdmPer}(D)_\perp)_A((R_a)_{a \in A}, \forall_A((S_{(a,\omega)})_{(a,\omega) \in A \times \Omega})),$$

i.e. morphism of types

$$(R_a)_{(a,\omega) \in A \times \Omega} \rightarrow (S_{(a,\omega)})_{(a,\omega) \in A \times \Omega}$$

and

$$(R_a)_{a \in A} \rightarrow \left( \bigcup_{\omega \in \Omega} S_{(a,\omega)} \right)_{a \in A} .$$

A natural bijection  $b$  is now evident from the fact, that for  $f \in [D \rightarrow D]$ :

$$\begin{aligned} \forall (a, \omega) \in A \times \Omega. \quad f \in \mathcal{F}(R_a, S_{(a,\omega)}) \\ \Downarrow \\ \forall a \in A. \quad f \in \mathcal{F}(R_a, \bigcup_{\omega \in \Omega} S_{(a,\omega)}) \end{aligned}$$

which we hasten to prove:

$\Downarrow$ : We have for all  $a \in A$ :

$$\begin{aligned} d R_a d' &\Rightarrow \forall \omega \in \Omega. f(d) S_{a,\omega} f(d') \\ &\Rightarrow f(d) \bigcup_{a \in A} S_{(a,\omega)} f(d') \end{aligned}$$

$\Uparrow$ : We have for all  $(a, \omega) \in A \times \Omega$ :

$$\begin{aligned} d R_a d' &\Rightarrow f(d) \bigcup_{\omega \in \Omega} S_{(a,\omega)} f(d') \\ &\Rightarrow \forall \omega \in \Omega. f(d) S_{(a,\omega)} f(d') \\ &\Rightarrow f(d) S_{(a,\omega)} f(d') \end{aligned}$$

### Beck-Chevalley

Given  $u: A \rightarrow B$  we must verify, that

$$u^* \circ \forall_B = \forall_A \circ (u \times id_\Omega)^*$$

So once again we calculate... on objects we get

$$(u^* \circ \forall_B)((R_{(b,\omega)})_{(b,\omega) \in B \times \Omega}) = u^* \left( \left( \bigcup_{\omega \in \Omega} R_{(b,\omega)} \right)_{b \in B} \right) = \left( \bigcup_{\omega \in \Omega} R_{(u(a),\omega)} \right)_{a \in A}$$

and

$$(\forall_A \circ (u \times id_\Omega)^*)((R_{(b,\omega)})_{(b,\omega) \in B \times \Omega}) = \forall_A \left( (R_{(u(a),\omega)})_{(a,\omega) \in A \times \Omega} \right) = \left( \bigcup_{\omega \in \Omega} R_{(u(a),\omega)} \right)_{a \in A}$$

... wow, and on morphisms:

$$\begin{aligned} &(u^* \circ \forall_B)(([\alpha]_{R_{(b,\omega)} \sim S_{(b,\omega)}})_{(b,\omega) \in B \times \Omega}) \\ &= u^* \left( ([\alpha]_{\bigcup_{\omega \in \Omega} R_{(b,\omega)} \sim \bigcup_{\omega \in \Omega} S_{(b,\omega)}})_{b \in B} \right) \\ &= ([\alpha]_{\bigcup_{\omega \in \Omega} R_{(u(a),\omega)} \sim \bigcup_{\omega \in \Omega} S_{(u(a),\omega)}})_{a \in A} \end{aligned}$$

and

$$\begin{aligned}
& (\forall_A \circ (u \times id_\Omega)^*)(([\alpha]_{R_{(b,\omega)} \sim S_{(b,\omega)}})_{(b,\omega) \in B \times \Omega}) \\
&= \forall_A(([\alpha]_{R_{(u(a),\omega)} \sim S_{(u(a),\omega)}})_{(u(a),\omega) \in A \times \Omega}) \\
&= ([\alpha]_{\cup_{\omega \in \Omega} R_{(u(a),\omega)} \sim \cup_{\omega \in \Omega} S_{(u(a),\omega)}})_{a \in A}
\end{aligned}$$

... Hallelujah! So we just need the canonical natural transformation to be the identity. It just so happens, that the canonical natural transformation being the identity is equivalent to the following equation among the counits of the adjunctions:

$$(u \times id_\Omega)^*(\epsilon^{\pi_B^* \dashv \forall B}) = \epsilon^{\pi_A^* \dashv \forall A}(u \times id_\Omega)^*$$

The counits, however, are easy to calculate. Remembering that transposition is merely reinterpretation of realizers, we see, that both counits are tracked by  $id_D$ . As  $(u \times id_\Omega)^*$  does not change the underlying realizer, both sides of the equation are tracked by  $id_D$ . Having the same domain and codomain<sup>2</sup> the morphisms are identical.  $\square$

**Theorem 47** *The considered setup constitutes a split PILL model.*

**Proof:** Immediate from the preceding propositions.  $\square$

## References

- [1] Rasmus Lerchedahl Petersen & Jacob Junker Thamsborg  
“Constructing a Parametric Model of the Polymorphic  $\lambda$ -Calculus”
- [2] Rasmus Lerchedahl Petersen & Jacob Junker Thamsborg  
“Polymorphism and Linearity all in one PILL”
- [3] B. Jacobs  
“Categorical Logic and Type Theory”
- [4] Rasmus Møgelberg & Lars Birkedal  
“On the definition of parametricity”
- [5] Rasmus Møgelberg, Lars Birkedal & Rasmus Lerchedahl Petersen  
“Parametric models of linear/intuitionistic polymorphism”
- [6] Saunders Mac Lane  
“Categories for the Working Mathematician”
- [7] John R. Longley and Alex K. Simpson  
“A Uniform Approach to Domain Theory in Realizability Models”
- [8] Masahito Hasegawa  
“Categorical Glueing and Logical Predicates for Models of Linear Logic”

---

<sup>2</sup>Otherwise the equation would not make sense...