Parametric Domain-theoretic models of Linear Abadi & Plotkin Logic

Lars Birkedal
Rasmus Ejlers Møgelberg
Rasmus Lerchedahl Petersen

IT University Technical Report Series

TR-2005-57

ISSN 1600–6100

February 2005

ISSN 1600–6100

ISBN 87-7949-086-7

Copies may be obtained by contacting:

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

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

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

1 Introduction **3**
1.1 Outline . . . 4

2 Linear Abadi-Plotkin Logic **5**
2.1 PILL$_Y$ . . . 5
2.1.1 Equality . . . 6
2.1.2 Ordinary lambda abstraction . . . 9
2.2 The logic . . . 9
2.2.1 Definable relations . . . 11
2.2.2 Constructions on definable relations . . . 11
2.2.3 Admissible relations . . . 13
2.2.4 Axioms and Rules . . . 13
2.2.5 Admissible relations preserved by structure maps . . . 17
2.2.6 Extensionality and Identity Extension Schemes . . . 18

3 Proofs in LAPL **20**
3.1 Logical Relations Lemma . . . 20
3.2 A category of linear functions . . . 23
3.3 Tensor types . . . 24
3.4 Unit object . . . 25
3.5 Natural Numbers . . . 26
3.5.1 Induction principle . . . 27
3.6 Types as functors . . . 27
3.7 Initial algebras . . . 29

3.8 Final Coalgebras . . . 31
3.9 Recursive type equations . . . 34
3.9.1 Parametrized initial algebras . . . 34
3.9.2 Dialgebras . . . 35
3.9.3 Compactness . . . 37

4 LAPL-structures **39**
4.1 Soundness . . . 49
4.2 Completeness . . . 51

5 Parametric LAPL-structures **54**
5.1 Solving recursive domain equations in parametric LAPL-structures . . . 55

6 Concrete Models **56**
6.1 The connection to CUPERs . . . 59
6.2 Lifting . . . 60
6.3 Going fibred . . . 62
6.4 A domain-theoretic model of PILL . . . 62
6.5 A parametric domain-theoretic model of PILL . . . 63

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 [23] (see also [18] for a recent textbook description). For real programming, one is of course not just interested in a strongly terminating calculus such as the second-order lambda calculus, but also in a language with full recursion. Thus in *loc. cit.* Reynolds also asked for a parametric *domain-theoretic* model of polymorphism. Informally, what is meant [24] by this is a model of an extension of the polymorphic lambda calculus [22, 7], with a polymorphic fixed-point operator $Y : \forall\alpha.\,(\alpha \to \alpha) \to \alpha$ such that

1. types are modeled as domains, the sublanguage without polymorphism is modeled in the standard way and $Y\sigma$ is the least fixed-point operator for the domain σ;
2. 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;
3. 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 [20] 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 [8]. 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 linear version of Abadi and Plotkin's logic for parametricity [21] with fixed points.

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

Recently, Pitts and coworkers [19, 4] 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 [25] have shown how to construct parametric domain-theoretic models using synthetic domain-theory in intuitionistic set-theory. Moreover, they have shown how to give a computationally adequate denotational semantics of Lily.

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

- We present a formalization of Linear Abadi-Plotkin Logic with fixed points (LAPL). The term language, called PILL_Y for polymorphic intuitionistic / linear logic, is a simple extension of Barber and Plotkin's calculus for dual intuitionistic / linear lambda calculus (DILL) with polymorphism and fixed points and the logic is an extension of Abadi-Plotkin's logic for parametricity with rules for forming admissible relations. The logic allows for intuitionistic reasoning over PILL$_Y$ terms; *i.e.*, the terms can be linear but the reasoning about terms is always done intuitionistically.

- We give detailed proofs in LAPL 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 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 show how to solve recursive domain equations in parametric LAPL-structures by a simple use of the internal language and the earlier proofs in LAPL.
- We present a detailed 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 [6, 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 also model 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 [16].

The work presented here builds upon our previous work on categorical models of Abadi-Plotkin's logic for parametricity [5], which includes detailed proofs of consequences of parametricity for polymorphic lamdba calculus and also includes a description of a parametric completion process that given an internal model of polymorphic lambda calculus produces a parametric model. It is not necessary to be familiar with the details of [5] to read the present paper (except for Appendix A of [5], which contains some definitions and theory concerning composable fibrations), but, for readers unfamiliar with parametricity, it may be helpful to start with [5], since the proofs of consequences of parametricity given here are slightly more sophisticated than the ones in [5] 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 [25] (this has been worked out at the time of writing [16]) and from Pitts and coworkers operational models [4] (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 [25] and [4] really do work out (these properties were not formally proved in *loc. cit.*). We will also extend the parametric completion process of [5] to produce a parametric LAPL-structure given a model of polymorphic intuitionistic / linear lambda calculus, see [14].

1.1 Outline

The remainder of this paper is organized as follows. In Section 2 we present LAPL, the logic for reasoning about parametricity over polymorphic intuitionistic / linear lambda calculus (PILL$_Y$). In Section 3 we give detailed proofs of many consequences of parametricity, including initial algebras and final coalgebras for definable functors and recursive types of mixed variance. In Section 4 we present our definition of an LAPL-structure, and we prove soundness and completeness with respect to LAPL in Sections 4.1 and 4.2, respectively. The definition of LAPL-structure builds upon fibred versions of models of intuitionistic / linear logic [3, 11]. In our presentation we assume that the reader is familiar with models of intuitionistic / linear logic.[1] 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 it easier to understand the model, we first present a model of PILL_Y (without parametricity) and then show how to make it into a parametric LAPL-structure. We also include an example of calculations in the concrete model.

2 Linear Abadi-Plotkin Logic

In this section we define a logic for reasoning about parametricity for Polymorphic Intuitionistic Linear Lambda calculus with fixed points (PILL_Y). The logic is based on Abadi and Plotkin's logic for parametricity [21] for the second-order lambda calculus and thus we refer to the logic as Linear Abadi-Plotkin Logic (LAPL).

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

2.1 PILL_Y

PILL_Y is essentially Barber and Plotkin's DILL [2] extended with polymorphism and a fixed point combinator.

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

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

where σ is built using the syntax

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

and all the free variables of sigma appear on the left hand side of the turnstile. The last construction binds α, so if we have a type

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

then we may form the type

$$\alpha_1 \colon \mathsf{Type}, \ldots, \alpha_{i-1}\mathsf{Type}, \alpha_{i+1}\mathsf{Type} \ldots \alpha_n \colon \mathsf{Type} \vdash \prod \alpha_i.\, \sigma \colon \mathsf{Type}.$$

We use σ, τ, ω, σ', τ'... to range over types. The list of α's is called the kind context, and is often denoted simply by Ξ or $\vec{\alpha}$. Since there is only one kind this annotation is often omitted.

The terms of PILL_Y are of the form:

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

[1]To aid readers unfamiliar with these matters, we have written a short technical note containing detailed definitions and propositions needed here [15].

where the σ_i, σ_i', and τ are well-formed types in the kind context Ξ. The list of x's is called the intuitionistic type context and is often denoted Γ, and the list of x''s is called the linear type context, often denoted Δ. No repetition of variable names is allowed in any of the contexts, but permutation akin to having an exchange rule is. Note, that due to the nature of the axioms of the to-be-introduced formation rules, weakening and contraction can be derived for all but the linear context.

The grammar for terms is:

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

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

The formation rules are given in Figure 1. $\Xi \mid \Gamma; \Delta$ is considered well-formed if for all types σ appearing in Γ and Δ, $\Xi \vdash \sigma\colon \mathsf{Type}$ is a well-formed type construction. Δ and Δ' are considered disjoint if the set of variables appearing in Δ is disjoint from the set of variables appearing in Δ'. We use $-$ to denote an empty context. As the types of variables in the let-constructions and function abstractions are often apparent from the context, these will just as often be omitted. What we have described above is called *pure* PILL$_Y$. In general we will consider PILL$_Y$ over polymorphic signatures [9, 8.1.1]. Informally, one may think of such a calculus as pure PILL$_Y$ with added type-constants and term-constants. For instance, one may have a constant type for integers or a constant type for lists $\alpha \vdash \mathit{lists}(\alpha)\colon \mathsf{Type}$. We will be particularly interested in the internal language of a PILL$_Y$ model (see Section 4), which in general will be a non-pure calculus.

We will also sometimes speak of the calculus PILL. This is PILL$_Y$ without the fixed point combinator Y.

2.1.1 Equality

The *external equality* relation on PILL$_Y$ terms is the least equivalence relation given by the rules in Figure 2. The definition makes use of the notion of a *context*, which, loosely speaking, is a term with exactly one hole in it. Formally contexts are defined using the grammar:

$$\begin{aligned} C[-] \quad ::= \quad & - \mid \text{let } \star \text{ be } C[-] \text{ in } t \mid \text{let } \star \text{ be } t \text{ in } C[-] \mid t \otimes C[-] \mid C[-] \otimes t \mid \\ & \text{let } x \otimes y \text{ be } C[-] \text{ in } t \mid \text{let } x \otimes y \text{ be } t \text{ in } C[-] \mid \lambda^\circ x\colon \sigma.\, C[-] \mid \\ & C[-]\, t \mid t\, C[-] \mid !C[-] \mid \text{let } !x \text{ be } C[-] \text{ in } t \mid \text{let } !x \text{ be } t \text{ in } C[-] \mid \\ & \Lambda\alpha\colon \mathsf{Type}.\, C[-] \mid C[-]\sigma \end{aligned}$$

A $\Xi \mid \Gamma; \Delta \vdash \sigma$ — $\Xi \mid \Gamma'; \Delta' \vdash \tau$ context is a context $C[-]$ such that for any well-formed term $\Xi \mid \Gamma; \Delta \vdash t\colon \sigma$, the term $\Xi \mid \Gamma'; \Delta' \vdash C[t]\colon \tau$ is well-formed. A context is **linear**, if it does not contain a subcontext of the form $!C[-]$.

We prove a couple of practical lemmas about external equality.

Lemma 2.1. *Suppose* $\Xi \mid \Gamma; \Delta \vdash f, g\colon\, !\sigma \multimap \tau$ *are terms such that*

$$\Xi \mid \Gamma, x\colon \sigma; \Delta \vdash f(!x) = g(!x).$$

Then $f = g$.

Proof. Using the rules for external equality, we conclude from the assumption that

$$\Xi \mid \Gamma; \Delta, y\colon\, !\sigma \vdash \text{let } !x \text{ be } y \text{ in } f(!x) = \text{let } !x \text{ be } y \text{ in } g(!x)$$

$$\frac{}{\Xi \mid \Gamma; - \vdash \star \colon I}$$

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\Xi \mid \Gamma; \Delta \vdash t \colon I \quad \Xi \mid \Gamma; \Delta' \vdash s \colon \sigma}{\Xi \mid \Gamma; \Delta, \Delta' \vdash \text{let } \star \text{ be } t \text{ in } s \colon \sigma}$$

Figure 1: Formation rules for terms

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash (\lambda^{\circ} x\colon \sigma.\, t)u = t[u/x]}\ \beta\text{-term}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash (\Lambda \alpha\colon \mathsf{Type}.\, t)\sigma = t[\sigma/\alpha]}\ \beta\text{-type}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \lambda^{\circ} x\colon \sigma.\, (tx) = t}\ \eta\text{-term}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \Lambda \alpha\colon \mathsf{Type}.\, (t\alpha) = t}\ \eta\text{-type}$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } \star \text{ in } t = t}\ \beta - \star$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } t \text{ in } \star = t}\ \eta - \star$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } s \otimes u \text{ in } t = t[s, u/x, y]}\ \beta - \otimes$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } t \text{ in } x \otimes y = t}\ \eta - \otimes$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x\colon \sigma \text{ be } !u \text{ in } t = t[u/x]}\ \beta - !$$

$$\frac{}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x\colon \sigma \text{ be } t \text{ in } !x = t}\ \eta - !$$

$$\frac{\Xi \mid \Gamma; \Delta \vdash t = s\colon \sigma \quad C[-] \text{ is a } \Xi \mid \Gamma; \Delta \vdash \sigma - \Xi \mid \Gamma'; \Delta' \vdash \tau \text{ context}}{\Xi \mid \Gamma'; \Delta' \vdash C[t] = C[s]\colon \tau}$$

$$\frac{C[-] \text{ is a linear context}}{\Xi \mid \Gamma; \Delta \vdash \text{let } \star \text{ be } t \text{ in } C[u] = C[\text{let } \star \text{ be } t \text{ in } u]}$$

$$\frac{C[-] \text{ is a linear context and does not bind } x, y \text{ or contain them free}}{\Xi \mid \Gamma; \Delta \vdash \text{let } x \otimes y \text{ be } t \text{ in } C[u] = C[\text{let } x \otimes y \text{ be } t \text{ in } u]}$$

$$\frac{C[-] \text{ is linear and does not bind } x \text{ or contain it free}}{\Xi \mid \Gamma; \Delta \vdash \text{let } !x \text{ be } t \text{ in } C[u] = C[\text{let } !x \text{ be } t \text{ in } u]}$$

$$\frac{\Xi \mid \Gamma; - \vdash f\colon !\sigma \multimap \sigma}{\Xi \mid \Gamma; - \vdash f\ !(Y\ \sigma\ (!f)) = Y\ \sigma\ (!f)}$$

Figure 2: Rules for external equality

$$\Xi\colon \mathsf{Ctx} \qquad \Xi \vdash \sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma; \Delta\colon \mathsf{Ctx}$$
$$\Xi \mid \Gamma \mid \Theta\colon \mathsf{Ctx} \qquad \Xi \mid \Gamma; \Delta \vdash t\colon \sigma \qquad \Xi \mid \Gamma; \Delta \vdash t = u$$
$$\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma, \tau) \qquad \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma, \tau)$$
$$\Xi \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop} \qquad \Xi \mid \Gamma \mid \Theta \mid \phi_1, \ldots, \phi_n \vdash \psi$$

Figure 3: Types of judgements

and further that
$$\Xi \mid \Gamma; \Delta, y\colon !\sigma \vdash f(\text{let } !x \text{ be } y \text{ in } !x) = g(\text{let } !x \text{ be } y \text{ in } !x).$$
Thus
$$\Xi \mid \Gamma; \Delta, y\colon !\sigma \vdash f(y) = g(y),$$
and hence $f = \lambda^\circ y\colon !\sigma.\, f(y) = \lambda^\circ y\colon !\sigma.\, g(y) = g$. $\square$

2.1.2 Ordinary lambda abstraction

We encode ordinary lambda abstraction in the usual way by defining
$$\sigma \to \tau = !\sigma \multimap \tau$$
and
$$\lambda x\colon \sigma.\, t = \lambda^\circ y\colon !\sigma.\, \text{let } !x \text{ be } y \text{ in } t$$
where y is a fresh variable. This gives us the rule
$$\frac{\Xi \mid \Gamma, x\colon \sigma; \Delta \vdash t\colon \tau}{\Xi \mid \Gamma; \Delta \vdash \lambda x\colon \sigma.\, t\colon \sigma \to \tau}$$
For evaluation we have the rule
$$\frac{\Xi \mid \Gamma; - \vdash t\colon \sigma \quad \Xi \mid \Gamma; \Delta \vdash f\colon \sigma \to \tau}{\Xi \mid \Gamma; \Delta \vdash f\, !t\colon \tau}$$
and the equality rules give
$$(\lambda x\colon \sigma.\, t)\, !s = t[s/x].$$
Note that using this notation the constant Y can obtain the more familiar looking type
$$Y : \Pi\alpha.\, (\alpha \to \alpha) \to \alpha$$

2.2 The logic

As mentioned, expressions of LAPL live in contexts of variables of PILL_Y and relations among types of PILL_Y. The contexts look like this:
$$\Xi \mid \Gamma \mid R_1\colon \mathsf{Rel}(\tau_1, \tau_1'), \ldots, R_n\colon \mathsf{Rel}(\tau_n, \tau_n'), S_1\colon \mathsf{AdmRel}(\omega_1, \omega_1'), \ldots, S_m\colon \mathsf{AdmRel}(\omega_m, \omega_m')$$
where $\Xi \mid \Gamma; -$ is a context of PILL_Y and the $\tau_i, \tau_i', \omega_i, \omega_i'$ are well-formed types in context Ξ, for all i. The list of R's and S's is called the relational context and is often denoted Θ. As for the other contexts we do not allow repetition, but permutation of variables. The R's and the S's are interchangeable.

The concept of admissible relations is taken from domain theory. Intuitively admissible relations relate $\perp$ to $\perp$ as expressed by axiom 2.18 described later.

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

Propositions in the logic are given by the syntax:

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

where ρ is a definable relation (to be defined below). The judgements of the logic are presented in figure 3. In the following we give formation rules for the above.

Remark 2.2. Our Linear Abadi & Plotkin logic is designed for reasoning about binary relational parametricity. For reasoning about other arities of parametricity, one can easily replace binary relations in the logic by relations of other arities. In the case of unary parametricity, for example, one would then have an interpretation of types as predicates. See also [26, 27]

We first have the formation rule for internal equality:

$$\frac{\Xi \mid \Gamma; - \vdash t\colon \sigma \qquad \Xi \mid \Gamma; - \vdash u\colon \sigma}{\Xi \mid \Gamma \mid \Theta \vdash t =_\sigma u\colon \mathsf{Prop}}$$

Notice here the notational difference between $t = u$ and $t =_\sigma u$. The former denotes *external* equality and the latter is a proposition in the logic. The rules for $\supset$, $\vee$ and $\wedge$ are the usual ones, where $\supset$ denotes implication. $\top$, $\perp$ are propositions in any context. We use $\supset\subset$ for biimplication.

We have the following formation rules for universal quantification:

$$\frac{\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall x\colon \sigma.\, \phi\colon \mathsf{Prop}}$$

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

$$\frac{\Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\sigma,\tau) \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall S\colon \mathsf{AdmRel}(\sigma,\tau).\, \phi\colon \mathsf{Prop}}$$

$$\frac{\Xi, \alpha \mid \Gamma \mid \Theta \vdash \phi\colon \mathsf{Prop}}{\Xi \mid \Gamma \mid \Theta \vdash \forall \alpha\colon \mathsf{Type}.\, \phi\colon \mathsf{Prop}}\ \Xi \mid \Gamma \mid \Theta \text{ is well-formed}$$

The side condition $\Xi \mid \Gamma \mid \Theta$ is well-formed means that all the types of variables in Γ and of relation variables in Θ are well-formed in Ξ (i.e., all the free type variables of the types occur in Ξ).

There are similar formation rules for the existential quantifier.

Before we give the formation rule for $\rho(t,u)$, we discuss definable relations.

2.2.1 Definable relations

Definable relations are given by the grammar:

$$\rho ::= R \mid (x\colon \sigma, y\colon \tau).\phi$$

Definable relations always have a domain and a codomain, just as terms always have types. The basic formation rules for definable relations are:

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

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

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

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

An example of a definable relation is the graph relation of a function:

$$\langle f\rangle = (x\colon \sigma, y\colon \tau).\, fx =_\tau y,$$

for $f\colon \sigma \multimap \tau$. The equality relation eq_σ is defined as the graph of the identity map.

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

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

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

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

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

2.2.2 Constructions on definable relations

In this subsection we present some constructions on definable relations, which will be used to give a relational interpretation of the types of PILL$_Y$.

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

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

defined by

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

If

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

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

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

as

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

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

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

The intuition here is that types are pointed, and $x \downarrow$ is thought of as $x \neq \bot$. Since we have also fixed points, we may think of types as domains.

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

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

Following the intuition of domains, ! is to be thought of as lifting, and ϵ the unit providing the unlifted version of an element. The intuitive reading of $!\rho$ is, that $\bot$ is related to $\bot$ (represented by the fact, that x is related to y if neither $x \downarrow$ nor $y \downarrow$) and that two !'ed elements are related if their un-!-ed versions are.

Next we define the tensor product of ρ and ρ'

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

for $\rho \colon \mathsf{Rel}(\sigma, \tau)$ and $\rho' \colon \mathsf{Rel}(\sigma', \tau')$. The basic requirement for this definition is that $\otimes$ should become a left adjoint to $\multimap$ in the category of relations $\mathbf{LinAdmRelations}$ to be defined 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

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

Then we define

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

or, if we write it out,

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

The reason for this at first sight fairly 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. We use the same trick to define a relation on I:

Following the same strategy as before, we define a relation $I_{Rel} \colon \mathsf{AdmRel}(I, I)$ using the map

$$f \colon I \multimap \prod \alpha.\, \alpha \multimap \alpha$$

defined as $\lambda^\circ x \colon I.\, \mathrm{let}\ \star\ \mathrm{be}\ x\ \mathrm{in}\ id$, where $id = \Lambda\alpha.\, \lambda^\circ x \colon \alpha.\, x$ and define

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

which, if we write it out, is

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

2.2.3 Admissible relations

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 4. In the last of these rules $\rho \equiv \rho'$ is a shorthand for $\forall x, y.\, \rho(x,y) \supset\subset \rho'(x,y)$.

Proposition 2.3. *The class of admissible relations contains all graphs and is closed under the constructions of Section 2.2.2.*

Proof. Graph relations are admissible since equality relations are and admissible relations are closed under reindexing. For the constructions of Section 2.2.2, we just give the proof of $\multimap$.

We must prove that for ρ, ρ' admissible relations $\rho \multimap \rho'$ is admissible

$$\dfrac{\dfrac{\dfrac{\Xi \mid \Gamma \mid \Theta \vdash \rho' \colon \mathsf{AdmRel}(\sigma', \tau')}{\Xi \mid \Gamma, x, y \mid \Theta \vdash (f,g).\, \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}(\sigma \multimap \sigma', \tau \multimap \tau')} \qquad \dfrac{\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{AdmRel}(\sigma, \tau)}{\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash \rho(x,y) \colon \mathsf{Prop}}}{\Xi \mid \Gamma, x \colon \sigma, y \colon \tau \mid \Theta \vdash (f \colon \sigma \multimap \sigma', g \colon \tau \multimap \tau').\, \rho(x,y) \supset \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}((\sigma \multimap \sigma'), (\tau \multimap \tau'))}}{\Xi \mid \Gamma \mid \Theta \vdash (f \colon \sigma \multimap \sigma', g \colon \tau \multimap \tau').\, \forall x \colon \sigma, y \colon \tau.\, \rho(x,y) \supset \rho'(f\,x, g\,y) \colon \mathsf{AdmRel}((\sigma \multimap \sigma'), (\tau \multimap \tau'))}$$

where in the top deduction on the left, we have reindexed ρ' along the evaluation maps

$$\lambda^\circ f \colon \sigma \multimap \sigma'.\, f\,x \qquad \lambda^\circ g \colon \tau \multimap \tau'.\, g\,y.$$

□

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

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

Observe that $\sigma[\vec{\rho}]$ is a syntactic construction and is not obtained by substitution as in [21]. Still the notation $\sigma[\rho_1/\alpha_1, \ldots, \rho_n/\alpha_n]$ might be more complete, but this quickly becomes overly verbose. In [21] $\sigma[\vec{\rho}]$ is to some extent defined inductively on the structure of σ, but in our case that is not enough, since we will need to form $\sigma[\vec{\rho}]$ for type constants (when using the internal language of a model of LAPL). We call $\sigma[\vec{\rho}]$ the *relational interpretation of the type σ*.

2.2.4 Axioms and Rules

The last judgement in figure 3 has not yet been mentioned. It says that in the given context, the formulas $\phi_1, \ldots, \phi_n$ collectively imply ψ. We will often write Φ for $\phi_1, \ldots, \phi_n$.

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 the axioms and rules specified below.

Rules for substitution:

Rule 2.4. $$\dfrac{\Xi \mid \Gamma, x \colon \sigma \mid \Theta \mid \top \vdash \phi \quad \Xi \mid \Gamma \vdash t \colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[t/x]}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 4: Rules for admissible relations

$$\textbf{Rule 2.5.}\ \frac{\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \mid \top \vdash \phi \quad \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[\rho/R]}$$

$$\textbf{Rule 2.6.}\ \frac{\Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\sigma,\tau) \mid \top \vdash \phi \quad \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{AdmRel}(\sigma,\tau)}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \phi[\rho/S]}$$

$$\textbf{Rule 2.7.}\ \frac{\Xi, \alpha \mid \Gamma \mid \Theta \mid \top \vdash \phi \quad \Xi \vdash \sigma\colon \mathsf{Type}}{\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \mid \top \vdash \phi[\sigma/\alpha]}$$

The *substitution* axiom:

$$\textbf{Axiom 2.8.}\ \forall \alpha, \beta\colon \mathsf{Type}.\forall x, x'\colon \alpha.\forall y, y'\colon \beta.\forall R\colon \mathsf{Rel}(\alpha,\beta.)R(x,y) \wedge \\ x =_\alpha x' \wedge y =_\beta y' \supset R(x', y')$$

Rules for $\forall$-quantification:

$$\textbf{Rule 2.9.}\ \begin{array}{c} \Xi, \alpha \mid \Gamma \mid \Theta \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall \alpha\colon \mathsf{Type}.\psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

$$\textbf{Rule 2.10.}\ \begin{array}{c} \Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall x\colon \sigma.\psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

$$\textbf{Rule 2.11.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau,\tau') \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall R\colon \mathsf{Rel}(\tau,\tau').\psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

$$\textbf{Rule 2.12.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\tau,\tau') \mid \Phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \Phi \vdash \forall S\colon \mathsf{AdmRel}(\tau,\tau').\psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \Phi$$

Rules for $\exists$-quantification:

$$\textbf{Rule 2.13.}\ \begin{array}{c} \Xi, \alpha \mid \Gamma \mid \Theta \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists \alpha\colon \mathsf{Type}.\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.14.}\ \begin{array}{c} \Xi \mid \Gamma, x\colon \sigma \mid \Theta \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists x\colon \sigma.\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.15.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau,\tau') \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists R\colon \mathsf{Rel}(\tau,\tau'.)\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

$$\textbf{Rule 2.16.}\ \begin{array}{c} \Xi \mid \Gamma \mid \Theta, S\colon \mathsf{AdmRel}(\tau,\tau') \mid \phi \vdash \psi \\ \hline\hline \Xi \mid \Gamma \mid \Theta \mid \exists S\colon \mathsf{AdmRel}(\tau,\tau').\phi \vdash \psi \end{array}\ \Xi \mid \Gamma \mid \Theta \vdash \psi$$

External equality implies internal equality:

$$\textbf{Rule 2.17.}\ \frac{\Xi \mid \Gamma; - \vdash t = u\colon \sigma}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash t =_\sigma u}$$

There are also 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 $x \neq \perp$.

Rule 2.18. $$\frac{\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(!\sigma, !\tau), \rho'\colon \mathsf{AdmRel}(!\sigma, !\tau) \quad x, y \notin \Gamma}{\begin{array}{l}\Xi \mid \Gamma \mid \Theta \mid \quad \forall x\colon \sigma, y\colon \tau.\, \rho(!x, !y) \supset \rho'(!x, !y) \vdash \\ \qquad\qquad \forall x\colon !\sigma, y\colon !\tau.\, x \downarrow\supset\subset y \downarrow\supset (\rho(x,y) \supset \rho'(x,y))\end{array}}$$

We have rules concerning the interpretation of types as relations:

Rule 2.19. $$\frac{\vec{\alpha} \vdash \alpha_i\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash \alpha_i[\vec{\rho}] \equiv \rho_i}$$

Rule 2.20. $$\frac{\vec{\alpha} \vdash \sigma \multimap \sigma'\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \multimap \sigma')[\vec{\rho}] \equiv (\sigma[\vec{\rho}] \multimap \sigma'[\vec{\rho}])}$$

Rule 2.21. $$\frac{\vec{\alpha} \vdash \sigma \otimes \sigma'\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (\sigma \otimes \sigma')[\vec{\rho}] \equiv (\sigma[\vec{\rho}] \otimes \sigma'[\vec{\rho}])}$$

Rule 2.22. $$\frac{\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash I[\vec{\rho}] \equiv I_{Rel}}$$

Rule 2.23. $$\frac{\vec{\alpha} \vdash \prod \beta.\, \sigma(\vec{\alpha}, \beta)\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \top \vdash (\prod \beta.\, \sigma(\vec{\alpha}, \beta))[\vec{\rho}] \equiv \forall(\beta, \beta', R\colon \mathsf{AdmRel}(\beta, \beta')).\, \sigma[\vec{\rho}, R])}$$

Rule 2.24. $$\frac{\vec{\alpha} \vdash !\sigma\colon \mathsf{Type} \qquad \Xi \mid \Gamma \mid \Theta \vdash \vec{\rho}\colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash (!\sigma)[\vec{\rho}] \equiv !(\sigma[\rho])}$$

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

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

Rule 2.25. $$\frac{\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi\colon \mathsf{Prop} \quad \Xi \mid \Gamma; - \vdash t\colon \sigma, u\colon \tau}{\Xi \mid \Gamma \mid \Theta \mid \top \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(t, u) \supset\subset \phi[t, u/x, y]}$$

Axiom 2.26. $$\frac{}{\Xi \mid \Gamma; - \mid \Theta \vdash Y(\prod \alpha.\, (\alpha \to \alpha) \to \alpha)Y}$$

Given a definable relation ρ we may construct a proposition $\rho(x, y)$. On the other hand, if ϕ is a proposition containing two free variables x and y, then we may construct the definable relation $(x, y).\, \phi$. The next lemma tells us that these constructions give a correspondence between definable relations and propositions, which is bijective up to provable equivalence in the logic.

Lemma 2.27. *Suppose ϕ is a proposition with at least two free variables $x\colon \sigma, y\colon \tau$. Then*

$$((x\colon \sigma, y\colon \tau).\, \phi)(x, y) \supset\subset \phi$$

Suppose $\rho\colon \mathsf{Rel}(\sigma, \tau)$ is a definable relation, then

$$\rho \equiv (x\colon \sigma, y\colon \tau).\, \rho(x, y).$$

Proof. The first biimplication follows trivially from Rule 2.25. For the second, we need to prove

$$\forall z\colon \sigma, w\colon \tau.\, \rho(z, w) \supset\subset ((x\colon \sigma, y\colon \tau).\, \rho(x, y))(z, w),$$

which is trivial by the same rule. □

The substitution axiom above implies the *replacement* rule:

Lemma 2.28.

$$\frac{\Xi \mid \Gamma \mid - \vdash t =_\sigma t' \qquad \Xi \mid \Gamma, x\colon \sigma; - \vdash u\colon \tau}{\Xi \mid \Gamma \mid - \vdash u[t/x] =_\tau u[t'/x]}$$

Proof. Consider the definable relation

$$\rho = (y\colon \sigma, z\colon \sigma).\, u[y/x] =_\tau u[z/x].$$

Clearly $\rho(t,t)$ holds, so by substitution $\rho(t,t')$ holds. ☐

Lemma 2.29.

$$\rho(x,y) \wedge \rho'(x',y') \supset \rho \otimes \rho'(x \otimes x', y \otimes y')$$

Proof. Suppose $\rho(x,y) \wedge \rho'(x',y')$ and that $(\rho \multimap \rho' \multimap R)(t,t')$. Then clearly $R(t\, x\, x', t'\, y\, y')$ and thus, since

$$\text{let } x \otimes x' \text{ be } x \otimes x' \text{ in } t\, x\, x' = t\, x\, x',$$

we conclude $\rho \otimes \rho'(x \otimes x', y \otimes y')$. ☐

Lemma 2.30. *For $x\colon \sigma$, $(!x)\downarrow$ always holds in the logic.*

Proof. Define $f\colon\ !\sigma \multimap I$ as $\lambda^\circ x\colon !\sigma$. let $!y$ be x in $\star$. Then clearly $f(!x) = \star$. ☐

Lemma 2.31. *For any $\rho\colon \mathsf{Rel}(\sigma,\tau)$, $x\colon \sigma, y\colon \tau$*

$$x \rho y \supset\subset !x(!\rho)!y$$

Proof. Since $\epsilon(!x) =$ let $!y$ be $!x$ in $y = x$ this follows from Lemma 2.30. ☐

2.2.5 Admissible relations preserved by structure maps

We now proceed to show a couple of practical lemmas expressing that various structure maps preserve admissible relations. The maps that we are interested in are

$$\epsilon_\sigma\colon !\sigma \multimap \sigma$$
$$\delta_\sigma\colon\ !\sigma \multimap !!\sigma,$$

which, categorically in the models of PILL_Y, are the structure maps of a comonad, and

$$d_\sigma\colon !\sigma \multimap !\sigma \otimes !\sigma$$
$$e_\sigma\colon !\sigma \multimap I,$$

which are the maps that make the comonad into a linear category. The maps are syntactically given as

$$\begin{array}{rcl} \epsilon_\sigma & = & \lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } y \\ \delta_\sigma & = & \lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } !!y \\ d_\sigma & = & \lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } !y \otimes !y \\ e_\sigma & = & \lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } \star\,. \end{array}$$

Lemma 2.32. *For all admissible relations $\rho\colon \mathsf{AdmRel}(\sigma,\tau)$,*

$$(\epsilon_\sigma, \epsilon_\tau)\colon !\rho \multimap \rho, \quad (\delta_\sigma, \delta_\tau)\colon !\rho \multimap !!\rho$$

are maps of relations, i.e., $!\rho(x,y)$ implies $\rho(\epsilon_\sigma x, \epsilon_\tau y)$ and $!!\rho(\delta_\sigma x, \delta_\tau y)$.

Proof. The lemma clearly holds in the case of x, y of the form $!x', !y'$. Since $\rho(\epsilon_\sigma x, \epsilon_\tau y)$ and $!!\rho(\delta_\sigma x, \delta_\tau y)$ both define admissible relations from $!\sigma$ to $!\tau$, by Rule 2.18 we conclude that $!\rho(x, y)$ implies

$$(x \downarrow \supset\subset y \downarrow) \supset \rho(\epsilon_\sigma x, \epsilon_\tau y)$$

and $(x \downarrow \supset\subset y \downarrow) \supset !!\rho(\delta_\sigma x, \delta_\tau y)$. Since $!\rho(x, y) \supset (x \downarrow \supset\subset y \downarrow)$ we are done. $\square$

Lemma 2.33. *For all admissible relations $\rho\colon \mathsf{AdmRel}(\sigma, \tau)$,*

$$(d_\sigma, d_\tau)\colon !\rho \multimap !\rho \otimes !\rho, \quad (e_\sigma, e_\tau)\colon !\rho \multimap I_{Rel}$$

are maps of relations, i.e., $!\rho(x, y)$ implies $!\rho\otimes!\rho(d_\sigma x, d_\tau y)$ and $I_{Rel}(e_\sigma x, e_\tau y)$.

Proof. To prove that (d, d) is a map of relations, since

$$\text{let } x' \otimes x'' \text{ be (let } !y \text{ be } x \text{ in } !y \otimes !y) \text{ in } t\ x'\ x'' = \text{let } !y \text{ be } x \text{ in } t\ !y\ !y$$

we need to prove that

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

Since the expression on the right hand side of the first $\supset$ is admissible in x, y and $!\rho(x, y) \supset x \downarrow \supset\subset y \downarrow$, by Rule 2.18 it suffices to prove the implication in the case $x = !x', y = !y'$. In this case, let $!z$ be x in $t\ !z\ !z = t\ !x'\ !x'$, so the implication is trivial.

To prove that (e, e) is a map of relations, we need to prove that

$$\begin{gathered} !\rho(x, y) \supset (\forall \alpha, \beta, R\colon \mathsf{AdmRel}(\alpha, \beta)).\, \forall z\colon \alpha, w\colon \beta. \\ zRw \supset (\text{let } \star \text{ be (let } !v \text{ be } x \text{ in } \star) \text{ in } z) R (\text{let } \star \text{ be (let } !v \text{ be } y \text{ in } \star) \text{ in } w). \end{gathered}$$

The implication clearly holds in the case of x, y of the form $!x', !y'$, and so, since $!\rho(x, y) \supset x \downarrow \wedge y \downarrow$, as before we conclude from Rule 2.18 that the implication holds in general. $\square$

2.2.6 Extensionality and Identity Extension Schemes

Consider the two **extensionality schemes**:

$$\begin{gathered} (\forall x\colon \sigma.\, t\ x =_\tau u\ x) \supset t =_{\sigma \multimap \tau} u \\ (\forall \alpha\colon \mathsf{Type}.\, t\ \alpha =_\tau u\ \alpha) \supset t =_{\prod \alpha\colon \mathsf{Type}.\tau} u. \end{gathered}$$

These are taken as axioms in [21], but we shall not take these as axioms as we would like to be able to talk about models that are not necessarily extensional.

Lemma 2.34. *It is provable in the logic that*

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

In particular, extensionality implies

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

Proof. This is just a special case of Rule 2.18. $\square$

The schema

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

is called the **identity extension schema**. Here σ ranges over all types, and $eq_{\vec{\alpha}}$ is short notation for $eq_{\alpha_1}, \ldots, eq_{\alpha_n}$.

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

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

where, for readability, we have omitted $:\mathsf{Type}$ after β, β'.

Proposition 2.35. *The identity extension schema implies the parametricity schema.*

Proof. The identity extension schema tells us that

$$\forall \vec{\alpha} \forall u \colon (\prod \beta.\, \sigma).\, u (\prod \beta.\, \sigma)[eq_{\vec{\alpha}}] u.$$

Writing out this expression using Rule 2.23 for the relational interpretation of polymorphic types, one obtains the parametricity schema. $\square$

In the case of second-order lambda-calculus, the parametricity schema implied identity extension for the pure calculus, since it provided the case of polymorphic types in a proof by induction. It is interesting to notice that this does not seem to be the case for PILL$_Y$, since it seems that we need identity extension to prove for example $eq_\sigma \otimes eq_\tau \equiv eq_{\sigma \otimes \tau}$.

Lemma 2.36. *Given linear contexts C and C', suppose*

$$\forall x \colon \sigma.\, \forall y \colon \tau.\, C[x \otimes y] =_\omega C'[x \otimes y].$$

then

$$\forall z \colon \sigma \otimes \tau.\, \mathit{let}\ x \otimes y\ \mathit{be}\ z\ \mathit{in}\ C[x \otimes y] =_\omega \mathit{let}\ x \otimes y\ \mathit{be}\ z\ \mathit{in}\ C'[x \otimes y]$$

Proof. Consider

$$f = \lambda^\circ x \colon \sigma.\, \lambda^\circ y \colon \tau.\, C[x \otimes y] \qquad f' = \lambda^\circ x \colon \sigma.\, \lambda^\circ y \colon \tau.\, C'[x \otimes y]$$

then

$$f\ (eq_\sigma \multimap eq_\tau \multimap eq_\omega)\ f'.$$

If $z \colon \sigma \otimes \tau$ then by identity extension $eq_\sigma \otimes eq_\tau(z, z)$. By definition of $eq_\sigma \otimes eq_\tau$ we have

$$\text{let } x \otimes x' \text{ be } z \text{ in } f x x' =_\omega \text{let } x \otimes x' \text{ be } z \text{ in } f' x x'$$

which proves the lemma. $\square$

This completes our presentation of LAPL. In the following section we show how to use the logic to prove various consequences of parametricity. We shall write “using extensionality” and “using identity extension” to mean that we assume the extensionality schemes and the identity extension schema, respectively.

3 Proofs in LAPL

3.1 Logical Relations Lemma

Lemma 3.1. *In pure LAPL, for $\vec{\alpha}, \beta \vdash \tau$, $\vec{\alpha} \vdash \omega$ and $\vec{\rho}\colon \mathsf{Rel}(\vec{\sigma}, \vec{\tau})$,*

$$\tau[\omega/\beta][\vec{\rho}] \equiv \tau[\vec{\rho}, \omega[\vec{\rho}]]$$

Proof. Simple induction on the structure of τ. The cases $\tau = \alpha_i$ and $\tau = \beta$ are trivial. For the case $\tau = \tau' \otimes \tau''$,

$$(\tau' \otimes \tau'')[\omega/\beta]\,[\vec{\rho}] \equiv \tau'[\omega/\beta][\vec{\rho}] \otimes \tau''[\omega/\beta][\vec{\rho}] \equiv \\ \tau'[\rho, \omega[\vec{\rho}]] \otimes \tau''[\rho, \omega[\vec{\rho}]] \equiv (\tau' \otimes \tau'')[\rho, \omega[\vec{\rho}]]$$

Likewise for the cases of $\tau = \tau' \multimap \tau''$ and $\tau = !\tau'$. The last case is $\tau = \prod \alpha'.\, \tau'$ and in this case α' is not free in ω, so

$$(\textstyle\prod \alpha'.\, \tau')[\omega/\beta][\vec{\rho}] \equiv \forall \alpha'_0, \alpha'_1, R\colon \mathsf{AdmRel}(\alpha'_0, \alpha'_1).\, \tau'[\omega/\beta][\vec{\rho}, R] \equiv \\ \forall \alpha'_0, \alpha'_1, R\colon \mathsf{AdmRel}(\alpha'_0, \alpha'_1).\, \tau'[\vec{\rho}, R, \omega[\vec{\rho}]] \equiv (\textstyle\prod \alpha'.\, \tau')[\vec{\rho}, \omega[\vec{\rho}]]$$

□

Lemma 3.2. *Suppose $\Xi \mid \Gamma, z\colon \sigma; - \vdash t(z)\colon \tau$ and $\Xi \mid \Gamma, z'\colon \sigma'; - \vdash t'(z')\colon \tau'$ and*

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

Then

$$\Xi \mid \Gamma \mid \Theta \mid \quad \forall x\colon \sigma, y\colon \sigma'.\, \rho(x, y) \supset \rho'(t(x), t'(y)) \vdash \\ \forall x\colon !\sigma, y\colon !\sigma'.\, !\rho(x, y) \supset \rho'(\textit{let } !z \textit{ be } x \textit{ in } t(z), \textit{let } !z' \textit{ be } y \textit{ in } t'(z'))$$

Proof. Consider the special case of Rule 2.18 used on the relations $(x\colon !\sigma, y\colon !\sigma').\, !\rho(x, y)$ and

$$(x\colon !\sigma, y\colon !\sigma').\, \rho'(\text{let } !z \text{ be } x \text{ in } t(z), \text{let } !z' \text{ be } y \text{ in } t'(z')).$$

This gives us

$$\Xi \mid \Gamma \mid \Theta \mid \quad \forall x\colon \sigma, y\colon \sigma'.\, !\rho(!x, !y) \supset \rho'(\text{let } !z \text{ be } !x \text{ in } t(z), \text{let } !z' \text{ be } !y \text{ in } t'(z')) \vdash \\ \forall x\colon !\sigma, y\colon !\sigma'.\, !\rho(x, y) \wedge (x \downarrow \supset\subset y \downarrow) \supset \rho'(\text{let } !z \text{ be } x \text{ in } t(z), \text{let } !z' \text{ be } y \text{ in } t'(z'))$$

From this we conclude the desired implication using the fact that $!\rho(!x, !y) \supset\subset \rho(x, y)$ (Lemma 2.31), let $!z$ be $!x$ in $t(z) = t(x)$ and $!\rho(x, y) \supset x \downarrow \supset\subset y \downarrow$. □

Lemma 3.3 (Logical Relations Lemma). *In pure LAPL, for any closed term $- \mid -; - \vdash t\colon \tau$,*

$$t \tau t.$$

In words, any closed term of closed type, is related to itself in the relational interpretation of the type.

Proof. We will prove that for any term

$$\vec{\alpha} \mid \vec{x}'\colon \vec{\sigma}'(\alpha); \vec{x}\colon \vec{\sigma}(\alpha) \vdash t(\vec{\alpha}, \vec{x}', \vec{x})\colon \tau$$

in the pure calculus; the proposition

$$- \mid -; - \vdash \forall \vec{\alpha}, \vec{\beta}.\, \forall \vec{R}\colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}).\, \forall \vec{x}\colon \vec{\sigma}(\vec{\alpha}), \vec{y}\colon \vec{\sigma}(\vec{\beta}).\, \forall \vec{x}'\colon \vec{\sigma}'(\vec{\alpha}), \vec{y}'\colon \vec{\sigma}'(\vec{\beta}). \\ \vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}' \supset t(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}]t(\vec{\beta}, \vec{y}', \vec{y})$$

holds in the logic. Here $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ is short for

$$x_1\sigma_1[\vec{R}]y_1 \wedge \ldots \wedge x_n\sigma_n[\vec{R}]y_n$$

The special case of the vectors $\vec{\alpha}, \vec{\sigma}, \vec{\sigma}'$ of length zero is the statement of the lemma. The proof proceeds by structural induction on t, and it is for the induction we need the seemingly stronger induction hypothesis described above.

Case $t = x_i$: In this case $\vec{x}$ is of length one, and the proposition is trivial.

Case $t = x_i'$: In this case $\vec{x}$ is empty, and this case is also trivial.

Case $t = \star$: We always have $\star I[\vec{R}]\star$.

Case $t = Y$: This is Axiom 2.26

Case $t = \lambda^\circ x_{n+1} \colon \sigma_{n+1}.\, t'$: By induction, the proposition holds for t'. We must show that if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})(\sigma_{n+1} \multimap \tau)[\vec{R}]t(\vec{\beta}\vec{y}'\vec{y}).$$

The induction hypothesis tells us that if further $x_{n+1}\sigma_{n+1}[\vec{R}]y_{n+1}$, then

$$t'(\vec{\alpha}, \vec{x}', \vec{x}, x_{n+1})\tau[\vec{R}]t'(\vec{\beta}, \vec{y}', \vec{y}, y_{n+1}),$$

and since $t(\vec{\alpha}, \vec{x}', \vec{x})x_{n+1} = t'(\vec{\alpha}, \vec{x}', \vec{x}, x_{n+1})$ we have the desired result.

Case $t = t'\ t''$: By induction the proposition holds for the terms t', t'', and so since

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{y})\ t''(\vec{\alpha}, \vec{x}', \vec{z})$$

the proposition holds by definition of $(\tau \multimap \tau')[\vec{R}]$.

Case $t = t' \otimes t''$: By induction, the proposition holds for t', t''. Clearly

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{y}) \otimes t''(\vec{\alpha}, \vec{x}', \vec{z})$$

and so the proposition holds by Lemma 2.29.

Case $t = \Lambda\alpha_{m+1}.\, t'$: We must show that if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})(\textstyle\prod \alpha_{m+1}.\, \tau)[\vec{R}]t(\vec{\beta}, \vec{y}', \vec{y}),$$

i.e., for all $\alpha_{m+1}, \alpha'_{m+1}, R_{m+1} \colon \mathsf{AdmRel}(\alpha_{m+1}, \alpha'_{m+1})$,

$$t(\vec{\alpha}, \vec{x}', \vec{x})\ \alpha_{m+1}\tau[\vec{R}, R_{m+1}]t(\vec{\beta}, \vec{y}', \vec{y})\ \alpha'_{m+1},$$

By induction, the proposition holds for t'. But up to the position of the quantifiers

$$\forall \alpha_{m+1}, \alpha'_{m+1}, R_{m+1} \colon \mathsf{AdmRel}(\alpha_{m+1}, \alpha'_{m+1}),$$

this is exactly the proposition we need, and the rest of the proof is just simple logic.

Case $t = t'(\omega)$: By induction, the proposition holds for t'. So since $t(\vec{\alpha}, \vec{x}', \vec{x}) = t'(\vec{\alpha}, \vec{x}', \vec{x})(\omega(\vec{\alpha}))$, if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y} \wedge \vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$, then

$$t(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}, \omega[\vec{R}]]t(\vec{\beta}, \vec{y}', \vec{y}).$$

By Lemma 3.1, we get the desired result.

Case $t = !t'$: In this case $\vec{x}$ is of length zero. By induction, under the usual assumptions,

$$t'(\vec{\alpha}, \vec{x}')\tau[\vec{R}]t'(\vec{\beta}, \vec{y}').$$

Since $t(\vec{\alpha}, \vec{x}') = !t'(\vec{\alpha}, \vec{x}')$, we need to show

$$!t'(\vec{\alpha}, \vec{x}')!\tau[\vec{R}]!t'(\vec{\beta}, \vec{y}').$$

which follows from Lemma 2.31.

Case $t =$ let $z\colon \omega \otimes z'\colon \omega'$ be t' in t'': We know by induction that

$$t'(\vec{\alpha}, \vec{x}', \vec{x})(\omega \otimes \omega')[\vec{R}]t'(\vec{\beta}, \vec{y}', \vec{y}),$$

and if further $z\omega[R]v$ and $z'\omega'[R]v'$ then

$$t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')\tau[\vec{R}]t''(\vec{\beta}, \vec{y}', \vec{y}, v, v').$$

The latter tells us that

$$\lambda^{\circ} z, z'.\, t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')(\omega[R] \multimap \omega'[R] \multimap \tau[R])\lambda^{\circ} v, v'.\, t''(\vec{\beta}, \vec{y}', \vec{y}, v, v'),$$

so by definition of $\omega[R] \otimes \omega'[R]$, we get

$$\begin{array}{c}\text{let } z\colon \omega \otimes z'\colon \omega' \text{ be } t'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } t''(\vec{\alpha}, \vec{x}', \vec{x}, z, z')\tau[R] \\ \text{let } v\colon \omega \otimes v'\colon \omega' \text{ be } t'(\vec{\beta}, \vec{y}', \vec{y}) \text{ in } t''(\vec{\beta}, \vec{y}', \vec{y}, v, v')\end{array}$$

as desired.

Case $t =$ let $!z\colon \omega$ be t' in t'': By definition

$$t(\vec{\alpha}, \vec{x}', \vec{x}) = \text{let } !z \text{ be } t'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } t''(\vec{\alpha}, \vec{x}', \vec{x}, z).$$

Suppose we are given $\vec{\alpha}, \vec{\beta}, \vec{R}\colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta})$, and suppose $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ and $\vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$. If we further know $z\omega[R]z'$, then by induction

$$t''(\alpha, \vec{x}', \vec{x}, z)\tau[R]t''(\beta, \vec{y}', \vec{y}, z').$$

By Lemma 3.2 we conclude that if $v(!\omega[R])v'$ then

$$(\text{let } !z \text{ be } v \text{ in } t''(\alpha, \vec{x}', \vec{x}, z))\tau[R](\text{let } !z \text{ be } v' \text{ in } t''(\beta, \vec{y}', \vec{y}, z)).$$

Since by induction $t'(\vec{\alpha}, \vec{x}', \vec{x})!\omega[\vec{R}]t'(\vec{\alpha}, \vec{y}', \vec{y})$, we are done.

Case $s =$ let $\star$ be s' in s'': By induction, if $\vec{x}\vec{\sigma}[\vec{R}]\vec{y}$ and $\vec{x}'\vec{\sigma}'[\vec{R}]\vec{y}'$ then

$$s''(\vec{\alpha}, \vec{x}', \vec{x})\tau[\vec{R}]s''(\vec{\beta}, \vec{y}', \vec{y})$$

and

$$s'(\vec{\alpha}, \vec{x}', \vec{x})I_{Rel}s'(\vec{\beta}, \vec{y}', \vec{y}).$$

The definition of the latter tells us exactly that

$$(\text{let } \star \text{ be } s'(\vec{\alpha}, \vec{x}', \vec{x}) \text{ in } s''(\vec{\alpha}, \vec{x}', \vec{x}))\tau[\vec{R}](\text{let } \star \text{ be } s'(\vec{\beta}, \vec{y}', \vec{y}) \text{ in } s''(\vec{\beta}, \vec{y}', \vec{y}))$$

as desired.

$\square$

3.2 A category of linear functions

At this point we wish to show certain types definable via polymorphism. To state this precisely, we introduce for each kind context Ξ the category $\mathbf{LinType}_\Xi$ as follows:

Objects are closed types $\Xi \mid -; - \vdash \sigma \colon \mathsf{Type}$.

Morphisms $[\Xi \mid -; - \vdash f \colon \sigma \multimap \tau]$ are equivalence classes of terms of type $\sigma \multimap \tau$; the equivalence relation on these terms being *internal* equality.

Composition in this category is given by lambda abstraction, i.e. $f \colon \sigma \multimap \tau$ composed with $g \colon \omega \multimap \sigma$ yields $\lambda^\circ x \colon \omega.\, f(gx)$.

We now aim to prove that under the assumption of identity extension and extensionality, for all types $\Xi \vdash \sigma \colon \mathsf{Type}$ we have an isomorphism of objects of $\mathbf{LinType}_\Xi$:

$$\sigma \cong \prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$$

for α not free in σ. We can define terms

$$f \colon \sigma \multimap \prod \alpha.\, ((\sigma \multimap \alpha) \multimap \alpha)$$

and

$$g \colon \prod \alpha.\, ((\sigma \multimap \alpha) \multimap \alpha) \multimap \sigma$$

by

$$f = \lambda^\circ x \colon \sigma.\, \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \alpha.\, h\, x$$

and

$$g = \lambda^\circ x \colon \prod \alpha.\, ((\sigma \multimap \alpha) \multimap \alpha).\, x\, \sigma\, id_\sigma$$

Clearly

$$g\,(f\,x) = (f\,x)\,\sigma\, id_\sigma = x$$

so $gf = id_\sigma$. Notice that this only involve external equality and thus we did not need extensionality here.

Proposition 3.4. *Using identity extension and extensionality, one may prove that fg is internally equal to the identity.*

Proof. For a term $a \colon \prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ we have

$$f \circ g\; a = \Lambda\alpha.\, \lambda^\circ h \colon \sigma \multimap \alpha.\, h(a\,\sigma\, id_\sigma).$$

Using extensionality, it suffices to prove that

$$\Xi, \alpha \mid h \colon \sigma \multimap \alpha \mid - \vdash h(a\,\sigma\, id_\sigma) =_\alpha a\,\alpha\,h$$

holds in the internal logic.

By the parametricity schema we know that for any admissible relation $\rho \colon \mathsf{AdmRel}(\tau, \tau')$

$$(a\,\tau)((eq_\sigma \multimap \rho) \multimap \rho)(a\,\tau')$$

If we instantiate this with the admissible relation $\langle h \rangle$, we get

$$(a\,\sigma)((eq_\sigma \multimap \langle h \rangle) \multimap \langle h \rangle)(a\,\alpha)$$

Since $id_\sigma(eq_\sigma \multimap \langle h \rangle)h$ we know that $(a\,\sigma\, id_\sigma)\langle h \rangle(a\,\alpha\,h)$, i.e.,

$$h(a\,\sigma\, id_\sigma) =_\alpha a\,\alpha\,h,$$

as desired. $\square$

This proof may essentially be found in [5].

Intuitively, what happens here is that σ is a subtype of $\prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$, where the inclusion f maps x to application at x. We use parametricity to show that $\prod \alpha.\, (\sigma \multimap \alpha) \multimap \alpha$ does not contain anything that is not in σ.

3.3 Tensor types

The goal of this section is to prove

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

for $\Xi \vdash \sigma\colon \mathsf{Type}$ and $\Xi \vdash \tau\colon \mathsf{Type}$ types in the same context. The isomorphism is in the category $\mathbf{LinType}_\Xi$.

This isomorphism leads to the question of weather tensor types are actually superfluous in the language. The answer is yes in the following sense: Call the language without tensor types (and I) t and the language as is T. Then there are transformations $p : T \to t$ and $i : t \to T$, i being the inclusion, such that $p \circ i = id_T$ and $i \circ p \cong id_t$. This is all being stated more precisely, not to mention proved, in [14, 13]. In this paper we settle for the isomorphism above.

We can construct terms

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

and

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

by

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

and

$$g\, y = y\, \sigma \otimes \tau\; \mathit{pairing},$$

where the map *pairing* $: \sigma \multimap \tau \multimap \sigma \otimes \tau$ is

$$\mathit{pairing} = \lambda^\circ x\colon \sigma.\, \lambda^\circ x'\colon \tau.\, x \otimes x'.$$

Let us show that the composition $g \circ f$ is the identity.

$$\begin{gathered} g \circ f\, y = g(\text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } y \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x') = \\ (\text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } y \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x')\, \sigma \otimes \tau\; \mathit{pairing} = \\ (\Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, \text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } y \text{ in } h\, x\, x')\, \sigma \otimes \tau\; \mathit{pairing} = \\ \text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } y \text{ in } x \otimes x' = y. \end{gathered}$$

Proposition 3.5. *Using extensionality and identity extension one may prove that the composition*

$$fg\colon (\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha) \multimap (\prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha)$$

is internally equal to the identity.

Proof. We compute

$$\begin{gathered} f \circ g\, y = f(y\, \sigma \otimes \tau\; \mathit{pairing}) = \\ \text{let } x \otimes x'\colon \sigma \otimes \tau \text{ be } (y\, \sigma \otimes \tau\; \mathit{pairing}) \text{ in } \Lambda\alpha.\, \lambda^\circ h\colon \sigma \multimap \tau \multimap \alpha.\, h\, x\, x' \end{gathered}$$

Suppose we are given a type α and a map $h \colon \sigma \multimap \tau \multimap \alpha$. We can define $\phi_h : \sigma \otimes \tau \multimap \alpha$ as

$$\phi_h = \lambda^\circ y \colon \sigma \otimes \tau.\, \text{let } x \otimes x' : \sigma \otimes \tau \text{ be } y \text{ in } h\, x\, x'.$$

Then $\phi_h(\mathit{pairing}\ x\ x') = h\ x\ x'$, which means that $\mathit{pairing}(\mathit{eq}_\sigma \multimap \mathit{eq}_\tau \multimap \langle \phi_h \rangle)h$. By the parametricity schema

$$\begin{array}{c} \Xi, \alpha \mid h \colon \sigma \multimap \tau \multimap \alpha, y \colon \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha \mid - \mid \top \vdash \\ (y\ \sigma \otimes \tau)((\mathit{eq}_\sigma \multimap \mathit{eq}_\tau \multimap \langle \phi_h \rangle) \multimap \langle \phi_h \rangle)(y\ \alpha) \end{array}$$

so

$$(y\ \sigma \otimes \tau\ \mathit{pairing}) \langle \phi_h \rangle (y\ \alpha\ h),$$

i.e,

$$\phi_h(y\ \sigma \otimes \tau\ \mathit{pairing}) =_\alpha y\ \alpha\ h.$$

Writing this out we get

$$\begin{array}{c} \Xi, \alpha \mid h \colon \sigma \multimap \tau \multimap \alpha, y \colon \prod \alpha.\, (\sigma \multimap \tau \multimap \alpha) \multimap \alpha \mid - \mid \top \vdash \\ \text{let } x \otimes x' : \sigma \otimes \tau \text{ be } (y\ \sigma \otimes \tau\ \mathit{pairing}) \text{ in } h\ x\ x' =_\alpha y\ \alpha\ h. \end{array}$$

Using extensionality we get

$$\Lambda \alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, \text{let } x \otimes x' : \sigma \otimes \tau \text{ be } (y\ \sigma \otimes \tau\ \mathit{pairing}) \text{ in } (h\ x\ x') =_\alpha y.$$

This is enough, since by the rules for external equality the left hand side is

$$\text{let } x \otimes x' : \sigma \otimes \tau \text{ be } (y\ \sigma \otimes \tau\ \mathit{pairing}) \text{ in } (\Lambda \alpha.\, \lambda^\circ h \colon \sigma \multimap \tau \multimap \alpha.\, h\ x\ x').$$

□

3.4 Unit object

The goal of this section is to prove that identity extension together with extensionality implies

$$I \cong \prod \alpha.\, \alpha \multimap \alpha.$$

The isomorphism holds in $\mathbf{LinType}_\Xi$ for all Ξ.

We first define maps $f \colon I \multimap \prod \alpha.\, \alpha \multimap \alpha$ and $g \colon (\prod \alpha.\, \alpha \multimap \alpha) \multimap I$ as

$$\begin{array}{c} f = \lambda^\circ x \colon I.\, \text{let } \star \text{ be } x \text{ in } \mathit{id}, \\ g = \lambda^\circ t \colon \prod \alpha.\, \alpha \multimap \alpha.\, t\ I\ \star, \end{array}$$

where

$$\mathit{id} = \Lambda \alpha.\, \lambda^\circ y \colon \alpha.\, y.$$

We first notice that

$$\begin{array}{c} g(f(x)) = (\text{let } \star \text{ be } x \text{ in } \mathit{id})\ I\ \star = \\ \text{let } \star \text{ be } x \text{ in } (\mathit{id}\ I\ \star) = \text{let } \star \text{ be } x \text{ in } \star = x. \end{array}$$

Proposition 3.6. *Using identity extension and extensionality, we have that fg is internally equal to the identity on $\prod \alpha.\, \alpha \multimap \alpha$.*

Proof. First we write out the definition

$$fg = \lambda^\circ t \colon (\textstyle\prod \alpha.\, \alpha \multimap \alpha).\, \text{let } \star \text{ be } (t\, I\, \star) \text{ in } id.$$

We show that for any $t\colon \prod \alpha.\, \alpha \multimap \alpha$, for any type σ, and any $x\colon \sigma$ we have $fg(t)\;\sigma\; x =_\sigma t\; \sigma\; x$.

Given σ, x as above, we can define $h\colon I \multimap \sigma$ as $h = \lambda^\circ z\colon I.\, \text{let } \star \text{ be } z \text{ in } x$. Then $\langle h\rangle$ is admissible, so by identity extension

$$(t\; I)(\langle h\rangle \multimap \langle h\rangle)(t\; \sigma).$$

Since $h(\star) = x$ we have $h(t\; I\, \star) =_\sigma t\; \sigma\; x$, and by definition

$$\begin{gathered} h(t\; I\, \star) = \text{let } \star \text{ be } (t\; I\, \star) \text{ in } x = \text{let } \star \text{ be } (t\; I\, \star) \text{ in } (id\; \sigma\; x) = \\ (\text{let } \star \text{ be } (t\; I\, \star) \text{ in } id)\; \sigma\; x = f \circ g(t)\; \sigma\; x. \end{gathered}$$

□

3.5 Natural Numbers

We define the type of natural numbers as

$$\mathbb{N} = \textstyle\prod \alpha.\, (\alpha \multimap \alpha) \to \alpha \multimap \alpha.$$

We further define terms $0\colon \mathbb{N}$, $s\colon \mathbb{N} \multimap \mathbb{N}$ as

$$0 = \Lambda\alpha.\, \lambda f\colon \alpha \multimap \alpha.\, \lambda^\circ x\colon \alpha.\, x, \quad s = \lambda^\circ y\colon \mathbb{N}.\, \Lambda\alpha.\, \lambda f : \alpha \multimap \alpha.\, \lambda^\circ x\colon \alpha.\, f(y\; \alpha\; !f\; x)$$

and prove that $(\mathbb{N}, 0, s)$ is a weak natural numbers object in each $\mathbf{LinType}_\Xi$, and, using parametricity and extensionality, an honest natural numbers object.

Suppose we are given a type σ, a term $a\colon \sigma$ and a morphism $b\colon \sigma \multimap \sigma$. We can then define $h\colon \mathbb{N} \multimap \sigma$ as $h(y) = y\, \sigma\, !b\, a$. Then clearly $h(0) = a$, and $h(s\, x) = b(x\, \sigma\, !b\, a) = b(h(x))$, so $(\mathbb{N}, 0, s)$ is a weak natural numbers object.

We can express the weak natural numbers object property as: for all a, b, there exists an h such that

$$\begin{tikzcd} I \arrow[r, multimap, "0"] \arrow[dr, multimap, "a"'] & \mathbb{N} \arrow[r, multimap, "s"] \arrow[d, multimap, "h"] & \mathbb{N} \arrow[d, multimap, "h"] \\ & \sigma \arrow[r, multimap, "b"] & \sigma \end{tikzcd}$$

commutes.

Lemma 3.7. *Identity Extension and extensionality implies*

$$\forall x\colon \mathbb{N}.\, x\; \mathbb{N}\; !s\; 0 =_\mathbb{N} x$$

Proof. Suppose we are given σ, a, b and define h as above. Since $b \circ h = h \circ s$ and $h\; 0 = a$, we have $s(\langle h\rangle \multimap \langle h\rangle)b$ and $0\langle h\rangle a$, by parametricity of x, $(x\; \mathbb{N}\; !s\; 0)\langle h\rangle(x\; \sigma\; !b\; a)$, i.e.,

$$(x\; \mathbb{N}\; !s\; 0)\; \sigma\; !b\; a =_\sigma x\; \sigma\; !b\; a.$$

Letting σ range over all types and a, b over all terms, using extensionality and Lemma 2.34, we have

$$x\; \mathbb{N}\; !s\; 0 =_\mathbb{N} x,$$

as desired. □

We can now prove that $\mathbb{N}$ is a natural numbers object in each $\mathbf{LinType}_{\Xi}$.

Lemma 3.8. *Assuming identity extension and extensionality, given σ, a, b, the map h defined as above is up to internal equality the unique h' such that $h'(0) = a$, $h'(s\ x) = b(h'\ x)$.*

Proof. Suppose h' satisfies the requirements of the lemma. Then $s(\langle h' \rangle \multimap \langle h' \rangle)b$ and $0\langle h' \rangle a$ (this is just a reformulation of the requirements), so for arbitrary $x\colon \mathbb{N}$, by parametricity of x,

$$x\ \sigma\ !b\ a =_{\sigma} h'(x\ \mathbb{N}\ !s\ 0) =_{\sigma} h'(x).$$

Thus, by extensionality, $h' =_{\mathbb{N} \multimap \sigma} h$. ∎

3.5.1 Induction principle

The parametricity principle for the natural numbers implies, that if $R\colon \mathsf{AdmRel}(\mathbb{N}, \mathbb{N})$, and $x\colon \mathbb{N}$, then

$$(x\ \mathbb{N})((R \multimap R) \to R \multimap R)(x\ \mathbb{N}).$$

So if $s(R \multimap R)s$ and $R(0, 0)$, then

$$(x\ \mathbb{N}\ !s\ 0)R(x\ \mathbb{N}\ !s\ 0).$$

By Lemma 3.7, $x\ \mathbb{N}\ !s\ 0 =_{\mathbb{N}} x$, so we can conclude that $R(x, x)$. If ϕ is a proposition on $\mathbb{N}$ such that $(x\colon \mathbb{N}, y\colon \mathbb{N}).\ \phi(x)$ is admissible, then from parametricity we obtain the usual induction principle

$$(\phi(0) \wedge \forall x\colon \mathbb{N}.\ \phi(x) \supset \phi(s(x))) \supset \forall x\colon \mathbb{N}.\ \phi(x).$$

3.6 Types as functors

Definition 3.9. We say that $\vec{\alpha} \vdash \sigma\colon \mathsf{Type}$ is an inductively constructed type, if it can be constructed from free variables $\vec{\alpha}$ and closed types using the type constructors of PILL$_Y$, i.e., $\multimap, \otimes, I, !$ and $\prod \alpha.$.

For example, all types of pure PILL$_Y$ are inductively defined, and if σ is a closed type then $\prod \alpha.\ \sigma \times \alpha$ is an inductively constructed type. However, some models may contain types that are not inductively constructed! For example, in syntactical models, any basic open type, such as the type $\alpha \vdash$ *lists*(α) is not inductively constructed.

We define positive and negative occurences of free type variables in inductively defined types as usual. The type variable α occurs positive in the type α and the positive occurences of a type variable α in $\sigma \multimap \tau$ are the positive occurences of α in τ and the negative in σ. The negative occurences of α in $\sigma \multimap \tau$ are the positive in σ and the negative in τ. The positive and negative occurences of α in $\prod \beta.\ \sigma$ are the positive and negative occurences in σ for $\alpha \neq \beta$. The rest of the type constructors preserve positive and negative occurences of type variables.

If $\sigma(\alpha, \beta)$ is an inductively defined type in which the free type variable α appears only negatively and the free type variable β appears only positively, then we can consider σ as a functor $\mathbf{LinType}^{op} \times \mathbf{LinType} \to \mathbf{LinType}$ by defining the term

$$M_{\sigma(\alpha,\beta)}\colon \prod \alpha, \beta, \alpha', \beta'.\ (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to \sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta'),$$

which behaves as the morphism part of a functor, i.e., it respects composition and preserves identities. We define $M_{\sigma(\alpha,\beta)}$ by structural induction on σ. This construction immediately generalizes to types with less or more than two free type variables, all of which appear only positively or negatively.

For the (nontrivial) base case of the induction, if $\sigma(\alpha, \beta) = \beta$, define

$$M_\beta = \Lambda\alpha, \beta, \alpha', \beta'.\, \lambda f, g.\, g.$$

In the case $\sigma(\beta, \alpha) \multimap \tau(\alpha, \beta)$ we define the term

$$M_{\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta)} \colon$$
$$\prod \alpha, \beta, \alpha', \beta'.\, (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to (\sigma(\beta, \alpha) \multimap \tau(\alpha, \beta)) \multimap \sigma(\beta', \alpha') \multimap \tau(\alpha', \beta')$$

by

$$M_{\sigma(\beta,\alpha)\multimap\tau(\alpha,\beta)} = \Lambda\alpha, \beta, \alpha', \beta'.\, \lambda f, g.$$
$$\lambda^\circ h \colon \sigma(\beta, \alpha) \multimap \tau(\alpha, \beta).\, (M_\tau\ \alpha\ \beta\ \alpha'\ \beta'\ f\ g) \circ h \circ (M_\sigma\ \beta'\ \alpha'\ \beta\ \alpha\ g\ f).$$

For bang types, we define:

$$M_{!\sigma(\alpha,\beta)} = \Lambda\alpha, \beta, \alpha', \beta'.\, \lambda f \colon \alpha' \multimap \alpha.\, \lambda g \colon \beta \multimap \beta'.\, \lambda^\circ x \colon !\sigma(\alpha, \beta).$$
$$\text{let } !y \text{ be } x \text{ in } !(M_{\sigma(\alpha,\beta)}\ \alpha\ \beta\ \alpha'\ \beta'\ f\ g\ y).$$

For tensor types, we define:

$$M_{\sigma(\alpha,\beta)\otimes\tau(\alpha,\beta)} = \Lambda\alpha, \beta, \alpha', \beta'.\, \lambda f, g.\, \lambda^\circ z \colon \sigma(\alpha, \beta) \otimes \tau(\alpha, \beta).$$
$$\text{let } x \otimes y \colon \sigma(\alpha, \beta) \otimes \tau(\alpha, \beta) \text{ be } z \text{ in } (M_\sigma \alpha\ \beta\ \alpha'\ \beta'\ f\ g\ x) \otimes (M_\tau \alpha\ \beta\ \alpha'\ \beta'\ f\ g\ y).$$

The last case is the case of polymorphic types:

$$M_{\prod \omega.\sigma(\alpha,\beta)} = \Lambda\alpha, \beta, \alpha', \beta'.\, \lambda f, g.\, \lambda^\circ z \colon\ \prod \omega.\, \sigma(\alpha, \beta).$$
$$\Lambda\omega \colon \mathsf{Type}.\, M_{\sigma(\alpha,\beta)}\ \alpha\ \beta\ \alpha'\ \beta'\ f\ g\ (z\ \omega).$$

Lemma 3.10. *The term M_σ respects composition and preserves identities, i.e., for $f' \colon \alpha'' \multimap \alpha'$, $f \colon \alpha' \multimap \alpha$, $g \colon \beta \multimap \beta'$, and $g' \colon \beta' \multimap \beta''$,*

- $M_{\sigma(\alpha,\beta)}\ \alpha\ \beta\ \alpha''\ \beta''!(f \circ f')\ !(g' \circ g) = (M_{\sigma(\alpha,\beta)}\ \alpha'\ \beta'\ \alpha''\ \beta''\ !f'\ !g') \circ (M_{\sigma(\alpha,\beta)}\ \alpha\ \beta\ \alpha'\ \beta'\ !f\ !g)$,
- $M_{\sigma(\alpha,\beta)}\alpha\ \beta\ \alpha\ \beta\ !id_\alpha !id_\beta = id_{\sigma(\alpha,\beta)}$.

Proof. The proof proceeds by induction over the structure of σ, and most of it is the same as in [21], except the case of tensor-types and !. These cases are essentially proved in [2]. ☐

Notice that in the proof of Lemma 3.10 we do not need parametricity. Suppose

$$\Xi \mid -; - \vdash f \colon \alpha' \multimap \alpha, g \colon \beta \multimap \beta'.$$

We shall write $\sigma(f, g)$ for

$$M_{\sigma(\alpha,\beta)}\alpha\ \beta\ \alpha'\ \beta'\ !f\ !g.$$

The type of $\sigma(f, g)$ is $\sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta')$. Notice that we apply M to $!f, !g$, since M is of intuitionistic function type ($\to$ instead of $\multimap$). By the previous lemma, σ defines a bifunctor $\mathbf{LinType}^{op} \times \mathbf{LinType} \to \mathbf{LinType}$.

First we consider this in the case of only one argument:

Lemma 3.11 (Graph lemma). *Assuming identity extension, for any type $\alpha \vdash \sigma$ with α occuring only positively and any map $f \colon \tau \multimap \tau'$*

$$\sigma[\langle f \rangle] \equiv \langle \sigma(f) \rangle.$$

Likewise, suppose $\alpha \vdash \sigma'$ is a type with α only occuring negatively. Then identity extension implies

$$\sigma[\langle f \rangle] \equiv \langle \sigma(f) \rangle^{\mathrm{op}},$$

where $\sigma(f))^{\mathrm{op}}$ is $(x \colon \sigma(\tau), y \colon \sigma(\tau')).\, \langle \sigma(f) \rangle(y, x)$.

Proof. We will only prove the first half of the lemma; the other half is proved the same way. Since α occurs only positively in σ, we will assume for readability that M_σ has type $\prod \alpha, \beta. (\alpha \multimap \beta) \to \sigma(\alpha) \multimap \sigma(\beta)$.

By parametricity of M_σ, for any pair of admissible relations $\rho\colon \mathsf{AdmRel}(\alpha, \alpha')$ and $\rho'\colon \mathsf{AdmRel}(\beta, \beta')$

$$(M_\sigma\ \alpha\ \beta)((\rho \multimap \rho') \to (\sigma[\rho] \multimap \sigma[\rho']))(M_\sigma\ \alpha'\ \beta'). \tag{2}$$

Let $f : \tau \multimap \tau'$ be arbitrary. If we instantiate (2) with $\rho = eq_\tau$ and $\rho' = \langle f \rangle$, we get

$$(M_\sigma\ \tau\ \tau)((eq_\tau \multimap \langle f \rangle) \to (eq_{\sigma(\tau)} \multimap \sigma[\langle f \rangle]))(M_\sigma\ \tau\ \tau'),$$

using the identity extension schema. Since $id_\tau(eq_\tau \multimap \langle f \rangle)f$,

$$!id_\tau!(eq_\tau \multimap \langle f \rangle)!f,$$

and using $M_\sigma\ \tau\ \tau'\ !f = \sigma(f)$ we get

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

i.e.,

$$\forall x\colon \sigma(\tau).\, x(\sigma[\langle f \rangle])(\sigma(f)x).$$

We have thus proved $\langle \sigma(f) \rangle$ implies $\sigma[\langle f \rangle]$.

To prove the other direction, instantiate (2) with the admissible relations $\rho = \langle f \rangle$, $\rho' = eq_{\tau'}$ for $f\colon \tau \multimap \tau'$. Since $f(\langle f \rangle \multimap eq_{\tau'})id_{\tau'}$,

$$\sigma(f)(\sigma[\langle f \rangle] \to eq_{\sigma(\tau')})id_{\sigma(\tau')}.$$

So for any $x\colon \sigma(\tau)$ and $y\colon \sigma(\tau')$ we have $x(\sigma[\langle f \rangle])y$ implies $\sigma(f)x =_{\sigma(\tau')} y$. This just means that $\sigma[\langle f \rangle]$ implies $\langle \sigma(f) \rangle$. ☐

3.7 Initial algebras

Suppose $\alpha \vdash \sigma\colon \mathsf{Type}$ is an inductively constructed type in which α occurs only positively. As we have just seen, such a type induces a functor

$$\mathbf{LinType}_\Xi \to \mathbf{LinType}_\Xi$$

for each Ξ. We aim to define an initial algebra for this type.

Define the closed type

$$\mu\alpha.\, \sigma(\alpha) = \prod \alpha.\, (\sigma(\alpha) \multimap \alpha) \to \alpha,$$

and define

$$\mathit{fold}\colon \prod \alpha.\, (\sigma(\alpha) \multimap \alpha) \to (\mu\alpha.\, \sigma(\alpha) \multimap \alpha)$$

as

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

and

$$\mathit{in}\colon \sigma(\mu\alpha.\, \sigma(\alpha)) \multimap \mu\alpha.\, \sigma(\alpha)$$

as

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

Lemma 3.12. *For any algebra $f\colon \sigma(\tau) \multimap \tau$, fold $\tau\ !f$ is a map of algebras from $(\mu\alpha.\,\sigma(\alpha), in)$ to (τ, f), i.e., the diagram*

$$\begin{tikzcd}
\sigma(\mu\alpha.\,\sigma(\alpha)) \arrow[r, "in", multimap] \arrow[d, "\sigma(\mathit{fold}\ \tau\ !f)"', multimap] & \mu\alpha.\,\sigma(\alpha) \arrow[d, "\mathit{fold}\ \tau\ !f", multimap] \\
\sigma(\tau) \arrow[r, "f", multimap] & \tau
\end{tikzcd}$$

commutes.

Proof. For $x\colon \sigma(\mu\alpha.\,\sigma(\alpha))$

$$(\mathit{fold}\ \tau\ !f) \circ \mathit{in}\ x = \mathit{in}\ x\ \tau\ !f = f(\sigma(\mathit{fold}\ \tau\ !f)\ x),$$

as desired. $\square$

In words we have shown that *in* defines a weakly initial algebra for the functor defined by σ in $\mathbf{LinType}_\Xi$ for each Ξ. Notice that parametricity was not needed in this proof.

Lemma 3.13. *Suppose $\Xi \mid \Gamma; - \vdash f\colon \sigma(\tau) \multimap \tau$ and $\Xi \mid \Gamma; - \vdash g\colon \sigma(\omega) \multimap \omega$ are algebras for σ, and $\Xi \mid \Gamma; - \vdash h\colon \tau \multimap \omega$ is a map of algebras, i.e., $h\ f = g\ \sigma(h)$. Then, assuming identity extension and extensionality,*

$$h \circ (\mathit{fold}\ \tau\ !f) =_{\mu\alpha.\sigma(\alpha)\multimap\omega} \mathit{fold}\ \omega\ !g.$$

Proof. Since h is a map of algebras

$$f(\langle\sigma(h)\rangle \multimap \langle h\rangle)g,$$

so by the Graph Lemma (3.11)

$$f(\sigma[\langle h\rangle] \multimap \langle h\rangle)g$$

and by Lemma 2.31

$$!f(!(\sigma[\langle h\rangle] \multimap \langle h\rangle))!g.$$

Clearly $(\mathit{fold},\mathit{fold}) \in eq_{\prod\alpha.(\sigma(\alpha)\multimap\alpha)\to(\mu\alpha.\sigma(\alpha)\multimap\alpha)}$, and thus, by identity extension,

$$(\mathit{fold},\mathit{fold}) \in \textstyle\prod\alpha.\,(\sigma(\alpha) \multimap \alpha) \to (\beta \multimap \alpha)[eq_{\mu\alpha.\sigma(\alpha)}/\beta],$$

so for any $x\colon \mu\alpha.\,\sigma(\alpha)$,

$$(\mathit{fold}\ \tau\ !f\ x)\langle h\rangle(\mathit{fold}\ \omega\ !g\ x),$$

i.e.,

$$h \circ (\mathit{fold}\ \tau\ !f) =_{\mu\alpha.\sigma(\alpha)\multimap\omega} \mathit{fold}\ \omega\ !g,$$

as desired. $\square$

Lemma 3.14. *Using identity extension and extensionality,*

$$\mathit{fold}\ \mu\alpha.\,\sigma(\alpha)\ !\mathit{in} =_{\mu\alpha.\sigma(\alpha)\multimap\mu\alpha.\sigma(\alpha)} id_{\mu\alpha.\sigma(\alpha)}.$$

Proof. By Lemma 3.13 we know that for any type τ, $f\colon \sigma(\tau) \multimap \tau$ and $u\colon \mu\alpha.\,\sigma(\alpha)$

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

The left hand side of this equation becomes

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

and, since the right hand side is simply

$$u\ \tau\ !f,$$

the lemma follows from Lemma 2.34. $\square$

Theorem 3.15. *Suppose* $\Xi \mid -; - \vdash f\colon \sigma(\tau) \multimap \tau$ *is an algebra and* $\Xi \mid -; - \vdash h\colon \mu\alpha.\,\sigma(\alpha) \multimap \tau$ *is a map of algebras from in to* f*. Then if we assume identity extension and extensionality,* $h =_{\mu\alpha.\sigma(\alpha)\multimap\tau} \mathit{fold}\ \tau\ !f$*.*

Proof. By Lemma 3.13 we have

$$h \circ (\mathit{fold}\ \mu\alpha.\,\sigma(\alpha)\ !\mathit{in}) =_{\mu\alpha.\sigma(\alpha)\multimap\tau} \mathit{fold}\ \tau\ !f.$$

Lemma 3.14 finishes the job. □

We have shown that *in* defines an initial algebra.

3.8 Final Coalgebras

As in section 3.7 we will assume that $\alpha \vdash \sigma(\alpha)\colon \mathsf{Type}$ is a type in which α occurs only positively, and this time we construct final coalgebras for the induced functor.

Given any $\alpha \vdash \tau(\alpha)\colon \mathsf{Type}$ we can define the closed type

$$\coprod \alpha.\,\tau(\alpha) = \prod \beta.\,(\prod \alpha.\,(\tau(\alpha) \multimap \beta)) \multimap \beta$$

with combinator

$$\mathit{pack}\colon \prod \alpha.\,(\tau(\alpha) \multimap \coprod \beta.\,\tau(\beta))$$

defined as

$$\mathit{pack} = \Lambda\alpha.\,\lambda^\circ x\colon \tau(\alpha).\,\Lambda\beta.\,\lambda^\circ f\colon \prod \alpha.\,(\tau(\alpha) \multimap \beta).\,f\ \alpha\ x.$$

Define

$$\nu\alpha.\,\sigma(\alpha) = \coprod \alpha.\,!(\alpha \multimap \sigma(\alpha)) \otimes \alpha = \prod \beta.\,(\prod \alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta)) \multimap \beta$$

with combinators

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

defined by

$$\begin{aligned} \mathit{unfold} &= \Lambda\alpha.\,\lambda^\circ f\colon !(\alpha \multimap \sigma(\alpha)).\,\lambda^\circ x\colon \alpha.\,\mathit{pack}\ \alpha\ (f \otimes x)\\ \mathit{out} &= \lambda^\circ x\colon \nu\alpha.\,\sigma(\alpha).\,x\ \sigma(\nu\alpha.\,\sigma(\alpha))\ r, \end{aligned}$$

where

$$\begin{gathered} r : \prod \alpha.\,!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \sigma(\nu\alpha.\,\sigma(\alpha))\\ r = \Lambda\alpha.\,\lambda^\circ y\colon !(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\,\text{let } w \otimes z \text{ be } y \text{ in } \sigma(\mathit{unfold}\ \alpha\ w)(\text{let } !f \text{ be } w \text{ in } f\ z). \end{gathered}$$

Lemma 3.16. *For any coalgebra* $f\colon \tau \multimap \sigma(\tau)$*, the map* $\mathit{unfold}\ \tau\ !f$ *is a map of coalgebras from* f *to out.*

Proof. We need to prove that the following diagram commutes

$$\begin{tikzcd} \tau \arrow[r, "f"] \arrow[d, "\mathit{unfold}\ \tau\ !f"'] & \sigma(\tau) \arrow[d, "\sigma(\mathit{unfold}\ \tau\ !f)"] \\ \nu\alpha.\,\sigma(\alpha) \arrow[r, "\mathit{out}"] & \sigma(\nu\alpha.\,\sigma(\alpha)). \end{tikzcd}$$

But this is done by a simple computation

$$\begin{gathered} \mathit{out}(\mathit{unfold}\ \tau\ !f\ x) = \mathit{out}(\mathit{pack}\ \ \tau(!f) \otimes x) =\\ \mathit{pack}\ \tau(!f) \otimes x\ \sigma(\nu\alpha.\,\sigma(\alpha))\ r = r\ \tau\ ((!f) \otimes x) =\\ \sigma(\mathit{unfold}\ \tau\ (!f))\ (f\ x). \end{gathered}$$

□

Lemma 3.16 shows that *out* is a weakly final coalgebra for the functor induced by σ on $\mathbf{LinType}_\Xi$ for each Ξ. Notice that parametricity was not needed here.

Lemma 3.17. *Suppose $h\colon (f\colon \tau \multimap \sigma(\tau)) \multimap (f'\colon \tau \multimap \sigma(\tau))$ is a map of coalgebras. If we assume identity extension, then the diagram*

$$\begin{tikzcd}
\tau \arrow[r, "\mathit{unfold}\,\tau\,!f"] \arrow[d, "h"'] & \nu\alpha.\,\sigma(\alpha) \\
\tau' \arrow[ur, "\mathit{unfold}\,\tau'\,!f'"'] &
\end{tikzcd}$$

commutes internally.

Proof. Using the Graph Lemma, the notion of h being a map of coalgebras can be expressed as

$$f(\langle h\rangle \multimap \sigma[\langle h\rangle])f'.$$

Now, by parametricity of *unfold*,

$$\mathit{unfold}\,\tau\,!f(\langle h\rangle \multimap eq_{\nu\alpha.\sigma(\alpha)})\mathit{unfold}\,\tau'\,!f',$$

which is exactly what we wanted to prove. $\square$

Lemma 3.18. *Using extensionality and identity extension,*

$$\mathit{unfold}\,\nu\alpha.\,\sigma(\alpha)\ !\mathit{out}$$

is internally equal to the identity on $\nu\alpha.\,\sigma(\alpha)$.

Proof. Set $h = \mathit{unfold}\,\nu\alpha.\,\sigma(\alpha)\ !\mathit{out}$ in the following.

By Lemma 3.16 h is a map of coalgebras from *out* to *out*, so by Lemma 3.17, $h = h^2$. Intuitively, all we need to prove now is that h is "surjective".

Consider any $g : \prod\alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta)$. For any coalgebra map $k : (f : \alpha \multimap \sigma(\alpha)) \multimap (f' : \alpha' \multimap \sigma(\alpha'))$, we must have, by Lemmas 3.11, 2.31, and 2.29,

$$(!f \otimes x)(!(\langle k\rangle \multimap \sigma[\langle k\rangle]) \otimes \langle k\rangle)(!f' \otimes kx),$$

so by identity extension and parametricity of g,

$$\forall x\colon \alpha.\, g\ \alpha\ (!f) \otimes x =_\beta g\ \alpha'\ (!f') \otimes k(x).$$

Using this on the coalgebra map $\mathit{unfold}\ \alpha\ !f$ from f to *out* we obtain

$$\forall x\colon \alpha.\, g\ \alpha\ (!f) \otimes x =_\beta g\ \nu\alpha.\,\sigma(\alpha)\ (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ !f\ x.$$

By Lemma 2.34 this implies that

$$\forall f\colon !(\alpha \multimap \sigma(\alpha)), x\colon \alpha.\, g\ \alpha\ f \otimes x =_\beta g\ \nu\alpha.\,\sigma(\alpha)\ (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ f\ x,$$

which implies

$$\forall z\colon !(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\, g\ \alpha\ z =_\beta g\ \nu\alpha.\,\sigma(\alpha)\ (\text{let } f \otimes x \text{ be } z \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ f\ x)$$

using Lemma 2.36.

In other words, if we define

$$k\colon \prod \alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \tau),$$

where $\tau = !(\nu\alpha.\,\sigma(\alpha) \multimap \sigma(\nu\alpha.\,\sigma(\alpha))) \otimes \nu\alpha.\,\sigma(\alpha)$, to be

$$k = \Lambda\alpha.\,\lambda^{\circ}y :!(\alpha \multimap \sigma(\alpha)) \otimes \alpha.\,\text{let } f \otimes x \text{ be } y \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ f\ x,$$

then

$$\forall\alpha.\, g\ \alpha =_{!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta} (g\ \nu\alpha.\,\sigma(\alpha)) \circ (k\ \alpha). \tag{3}$$

Now, suppose we are given $\alpha, \alpha', R\colon \mathsf{Rel}(\alpha, \alpha')$ and terms f, f' such that

$$f(!(R \multimap \sigma[R]) \otimes R)f'.$$

Then, by (3) and parametricity of g

$$g\ \alpha\ f =_{\beta} g\ \alpha'\ f' =_{\beta} (g\ \nu\alpha.\,\sigma(\alpha))(k\ \alpha'\ f'),$$

from which we conclude

$$g(\forall(\alpha, \beta, R\colon \mathsf{Rel}(\alpha, \beta)).\,(!(R \multimap \sigma[R]) \otimes R \multimap \langle g\ \nu\alpha.\,\sigma(\alpha)\rangle^{op}))k.$$

(Here we use S^{op} for the inverse relation of S.) Using parametricity, this implies that, for any $x\colon \nu\alpha.\,\sigma(\alpha)$, we have

$$x\ \beta\ g =_{\beta} g\ \nu\alpha.\,\sigma(\alpha)\ (x\ \tau\ k).$$

Thus, since g was arbitrary, we may apply the above to $g = k$ and get

$$x\ \tau\ k =_{\tau} k\ \nu\alpha.\,\sigma(\alpha)\ (x\ \tau\ k) = \text{let } f \otimes z \text{ be } (x\ \tau\ k) \text{ in } (!\mathit{out}) \otimes \mathit{unfold}\ \alpha\ f\ z.$$

If we write

$$l = \lambda x\colon \nu\alpha.\,\sigma(\alpha).\,\text{let } f \otimes z \text{ be } (x\ \tau\ k) \text{ in } \mathit{unfold}\ \alpha\ f\ z,$$

then, since k is a closed term, so is l, and from the above calculations we conclude that we have

$$\forall\beta.\,\forall g : \prod \alpha.\,!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta.\, x\ \beta\ g =_{\beta} g\ \nu\alpha.\,\sigma(\alpha)\ (!\mathit{out}) \otimes (l\ x).$$

Now, finally,

$$\begin{gathered} h(l\ x) = \mathit{unfold}\ \nu\alpha.\,\sigma(\alpha)\ !\mathit{out}\ (l\ x) = \\ \mathit{pack}\ \nu\alpha.\,\sigma(\alpha)\ !\mathit{out} \otimes (l\ x) = \\ \Lambda\beta.\,\lambda g : \prod \alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta).\, g\ \nu\alpha.\,\sigma(\alpha)\ !\mathit{out} \otimes (l\ x) =_{\nu\alpha.\sigma(\alpha)} \\ \Lambda\beta.\,\lambda g : \prod \alpha.\,(!(\alpha \multimap \sigma(\alpha)) \otimes \alpha \multimap \beta).\, x\ \beta\ g = x, \end{gathered}$$

where we have used extensionality. Thus l is a right inverse to h, and we conclude

$$h\,x =_{\nu\alpha.\sigma(\alpha)} h^2(l\ x) =_{\nu\alpha.\sigma(\alpha)} h(l\ x) =_{\nu\alpha.\sigma(\alpha)} x.$$

$\square$

Theorem 3.19. *Suppose* $\Xi \mid -; - \vdash f\colon \tau \multimap \sigma(\tau)$ *is a coalgebra and* $\Xi \mid -; - \vdash h\colon \tau \multimap \mu\alpha.\,\sigma(\alpha)$ *is a map of algebras from* f *to out. Then if we assume identity extension and extensionality* $h =_{\tau \multimap \mu\alpha.\sigma(\alpha)}$ *unfold* $\alpha\,!f$.

Proof. Consider a map of coalgebras into *out*:

$$\begin{tikzcd}
\tau \arrow[r, "f", multimap] \arrow[d, "h", multimap] & \sigma(\tau) \arrow[d, "\sigma(h)", multimap] \\
\nu\alpha.\,\sigma(\alpha) \arrow[r, "out", multimap] & \sigma(\nu\alpha.\,\sigma(\alpha)).
\end{tikzcd}$$

By Lemmas 3.17 and 3.18,

$$\mathit{unfold}\,\tau\,!f =_{\tau\multimap\nu\alpha.\sigma(\alpha)} (\mathit{unfold}\,\nu\alpha.\,\sigma(\alpha)\,!\mathit{out})\circ g =_{\tau\multimap\nu\alpha.\sigma(\alpha)} g.$$

□

Theorem 3.19 shows that *out* is a final coalgebra for the endofunctor on $\mathbf{LinType}_{\Xi}$ induced by σ for each Ξ.

3.9 Recursive type equations

In this section we consider inductively constructed types $\alpha \vdash \sigma(\alpha)$ and construct closed types τ such that $\sigma(\tau) \cong \tau$. In Sections 3.7 and 3.8 we solved the problem in the special case of α occuring only positively in σ, by finding initial algebras and final coalgebras for the functor induced by σ.

The first observation we use is that we may split the occurences of α in σ in positive and negative occurences. So our standard assumption in this section is that we are given a type $\alpha, \beta \vdash \sigma(\alpha, \beta)$, in which α occurs only negatively and β only positively, and we look for a type τ, such that $\sigma(\tau, \tau) \cong \tau$. This section details the sketch of [20].

3.9.1 Parametrized initial algebras

Set $\omega(\alpha) = \mu\beta.\,\sigma(\alpha,\beta) = \prod\beta.\,(\sigma(\alpha,\beta) \multimap \beta) \multimap \beta$. Now, ω induces a contravariant functor from types to types.

Lemma 3.20. *Assuming identity extension and extensionality, for $f\colon \alpha' \multimap \alpha$, $\omega(f)\colon \omega(\alpha) \multimap \omega(\alpha')$ is (up to internal equality) the unique h such that*

$$\begin{tikzcd}
\sigma(\alpha,\omega(\alpha)) \arrow[r, "in", multimap] \arrow[d, "\sigma(id{,}h)", multimap] & \omega(\alpha) \arrow[dd, "h", multimap] \\
\sigma(\alpha,\omega(\alpha')) \arrow[d, "\sigma(f{,}id)", multimap] & \\
\sigma(\alpha',\omega(\alpha')) \arrow[r, "in", multimap] & \omega(\alpha')
\end{tikzcd}$$

commutes internally.

Proof. One may define *in* as a polymorphic term

$$in\colon \prod\alpha.\,\sigma(\alpha,\omega(\alpha)) \multimap \omega(\alpha)$$

by

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

By parametricity we have

$$in\ \alpha'(\sigma(\langle f\rangle, \omega(\langle f\rangle)) \multimap \omega(\langle f\rangle))in\ \alpha,$$

which, by the Graph Lemma (Lemma 3.11), means that

$$in\ \alpha'(\langle\sigma(f, \omega(f))\rangle^{\mathrm{op}} \multimap \langle\omega(f)\rangle^{\mathrm{op}})in\ \alpha,$$

which in turn amounts to internal commutativity of the diagram of the lemma.

Uniqueness is by initiality of *in* (in $\mathbf{LinType}_\alpha$, proved as before) used on the diagram

$$\begin{tikzcd}
\sigma(\alpha, \omega(\alpha)) \arrow[rr, multimap, "in"] \arrow[d, multimap, "\sigma(id,h)"'] & & \omega(\alpha) \arrow[d, multimap, "h"] \\
\sigma(\alpha, \omega(\alpha')) \arrow[r, multimap, "\sigma(f,id)"] & \sigma(\alpha', \omega(\alpha')) \arrow[r, multimap, "in"] & \omega(\alpha').
\end{tikzcd}$$

□

3.9.2 Dialgebras

Definition 3.21. A dialgebra for σ is a quadruple (τ, τ', f, f') such that τ and τ' are types, and $f\colon \sigma(\tau', \tau) \multimap \tau$ and $f'\colon \tau' \multimap \sigma(\tau, \tau')$ are morphisms. A morphism of dialgebras from $(\tau_0, \tau_0', f_0, f_0')$ to $(\tau_1, \tau_1', f_1, f_1')$ is a pair of morphisms $h\colon \tau_0 \multimap \tau_1$, $h'\colon \tau_1' \multimap \tau_0'$, such that

$$\begin{tikzcd}
\sigma(\tau_0', \tau_0) \arrow[r, multimap, "f_0"] \arrow[d, multimap, "\sigma(h',h)"'] & \tau_0 \arrow[d, multimap, "h"] \\
\sigma(\tau_1', \tau_1) \arrow[r, multimap, "f_1"'] & \tau_1
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau_1' \arrow[r, multimap, "f_1'"] \arrow[d, multimap, "h'"'] & \sigma(\tau_1, \tau_1') \arrow[d, multimap, "\sigma(h,h')"] \\
\tau_0' \arrow[r, multimap, "f_0'"'] & \sigma(\tau_0, \tau_0').
\end{tikzcd}$$

Lemma 3.22. *If (h, h') is a map of dialgebras and h, h' are isomorphisms, then (h, h') is an isomorphism of dialgebras.*

Proof. The only thing to prove here is that $(h^{-1}, (h')^{-1})$ is in fact a map of dialgebras, which is trivial. □

Remark 3.23. If we for the type $\alpha, \beta \vdash \sigma\colon$ Type consider the endofunctor

$$\langle\sigma^{\mathrm{op}}, \sigma\rangle\colon \mathbf{LinType^{op}}_\Xi \times \mathbf{LinType}_\Xi \to \mathbf{LinType^{op}}_\Xi \times \mathbf{LinType}_\Xi$$

defined by $(\alpha, \beta) \mapsto (\sigma(\beta, \alpha), \sigma(\alpha, \beta))$, then dialgebras for σ are exactly the algebras for $\langle\sigma^{\mathrm{op}}, \sigma\rangle$, maps of dialgebras are maps of algebras for $\langle\sigma^{\mathrm{op}}, \sigma\rangle$ and initial dialgebras correspond to initial algebras.

Theorem 3.24. *Assuming identity extension and extensionality, initial dialgebras exist for all functors induced by types $\sigma(\alpha, \beta)$, up to internal equality.*

Proof. In this proof, commutativity of diagrams will mean commutativity up to internal equality.

Set $\omega(\alpha) = \mu\beta.\, \sigma(\alpha, \beta)$. Then, ω defines a contravariant functor. Define

$$\tau' = \nu\alpha.\, \sigma(\omega(\alpha), \alpha), \qquad \tau = \omega(\tau') = \mu\beta.\, \sigma(\tau', \beta).$$

Since τ' is defined as the final coalgebra for a functor, we have a morphism

$$out\colon \tau' \multimap \sigma(\omega(\tau'), \tau') = \sigma(\tau, \tau'),$$

and since τ is defined to be an initial algebra, we get a morphism

$$in\colon \sigma(\tau', \tau) \multimap \tau.$$

We will show that (τ, τ', in, out) is an initial dialgebra.

Suppose we are given a dialgebra (τ_0, τ_0', g, g'). Since *in* is an initial algebra, there exists a unique map a, such that

$$\begin{tikzcd}
\sigma(\tau_0', \omega(\tau_0')) \arrow[r, "in"] \arrow[d, "\sigma(id,a)"'] & \omega(\tau_0') \arrow[d, "a"] \\
\sigma(\tau_0', \tau_0) \arrow[r, "g"] & \tau_0,
\end{tikzcd}$$

and thus, since *out* is a final coalgebra, we find a map h' making the diagram

$$\begin{tikzcd}
\tau_0' \arrow[r, "g'"] \arrow[d, "h'"'] & \sigma(\tau_0, \tau_0') \arrow[r, "\sigma(a,id)"] & \sigma(\omega(\tau_0'), \tau_0') \arrow[d, "\sigma(\omega(h'),h')"] \\
\tau' \arrow[rr, "out"] & & \sigma(\omega(\tau'), \tau')
\end{tikzcd} \tag{4}$$

commute. Set $h = a \circ \omega(h')$. We claim that (h, h') defines a map of dialgebras. The second diagram of Definition 3.21 is simply (4). The first diagram of 3.21 follows from the commutativity of the composite diagram

$$\begin{tikzcd}
\sigma(\tau', \omega(\tau')) \arrow[r, "in"] \arrow[d, "\sigma(h',\omega(h'))"'] & \omega(\tau') \arrow[d, "\omega(h')"] \\
\sigma(\tau_0', \omega(\tau_0')) \arrow[r, "in"] \arrow[d, "\sigma(id,a)"'] & \omega(\tau_0') \arrow[d, "a"] \\
\sigma(\tau_0', \tau_0) \arrow[r, "g"] & \tau_0,
\end{tikzcd} \tag{5}$$

where the top diagram commutes by Lemma 3.20.

Finally, we will prove that (h, h') is the unique dialgebra morphism. Suppose we are given a map of dialgebras (k, k') from (τ, τ', in, out) to (τ_0, τ_0', g, g'). By the first diagram of Definition 3.21, we have a commutative diagram

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[rr, "in"] \arrow[d, "\sigma(id,k)"'] & & \tau \arrow[d, "k"] \\
\sigma(\tau', \tau_0) \arrow[r, "\sigma(k',id)"] & \sigma(\tau_0', \tau_0) \arrow[r, "g"] & \tau_0.
\end{tikzcd}$$

Since clearly (5) also commutes when k' is substituted for h', by (strong) initiality of *in*, we conclude that $k =_{\tau \multimap \tau'} a \circ \omega(k')$. Finally, by the second diagram of Definition 3.21 we have commutativity of

$$\begin{tikzcd}
\tau_0' \arrow[r, "g'"] \arrow[d, "k'"'] & \sigma(\tau_0, \tau_0') \arrow[r, "\sigma(a,id)"] & \sigma(\omega(\tau_0'), \tau_0') \arrow[d, "\sigma(\omega(k'),k')"] \\
\tau' \arrow[rr, "out"] & & \sigma(\omega(\tau'), \tau').
\end{tikzcd}$$

So since *out* is a final coalgebra we conclude $k' =_{\tau_0' \multimap \tau'} h'$. ☐

3.9.3 Compactness

Theorem 3.25 (Compactness). *Assuming identity extension and extensionality, for all types $\alpha \vdash \sigma(\alpha)$ in which α occurs only positively, in^{-1} is internally a final coalgebra and out^{-1} is internally an initial algebra. Furthermore in^{-1} and out^{-1} can be written as terms of $PILL_Y$.*

Proof. By Theorems 3.15 and 3.19 *in* is an initial algebra, and *out* is a final coalgebra for σ. Consider

$$h = Y\ (\nu\alpha.\,\sigma(\alpha)) \multimap \mu\alpha.\,\sigma(\alpha)\ (\lambda h\colon \nu\alpha.\,\sigma(\alpha) \multimap \mu\alpha.\,\sigma(\alpha).\, in \circ \sigma(h) \circ out).$$

Since Y is a fixed-point operator, we know that

$$\begin{tikzcd} \sigma(\nu\alpha.\,\sigma(\alpha)) \arrow[d, "\sigma(h)"'] & \nu\alpha.\,\sigma(\alpha) \arrow[l, "out"'] \arrow[d, "h"] \\ \sigma(\mu\alpha.\,\sigma(\alpha)) \arrow[r, "in"] & \mu\alpha.\,\sigma(\alpha) \end{tikzcd}$$

commutes. Since in^{-1} is a coalgebra, we also have a map k going the other way, and since *out* is a final coalgebra, $kh =_{\nu\alpha.\sigma(\alpha)\multimap\nu\alpha.\sigma(\alpha)} id_{\nu\alpha.\sigma(\alpha)}$. Since *in* is an initial algebra, we know that $hk =_{\mu\alpha.\sigma(\alpha)\multimap\mu\alpha.\sigma(\alpha)} id_{\mu\alpha.\sigma(\alpha)}$. So $in^{-1} \cong out$ as coalgebras and $out^{-1} \cong in$ as algebras, internally. □

Lemma 3.26. *Assume identity extension and extensionality. Let (τ, τ', in, out) be the initial dialgebra from the proof of Theorem 3.24. Then $(\tau', \tau, out^{-1}, in^{-1})$ is also an initial dialgebra internally.*

Proof. In this proof, commutativity of diagrams is up to internal equality.

Suppose we are given a dialgebra (τ_0, τ_0', g, g'). We will show that there exists a unique morphism of dialgebras from $(\tau', \tau, out^{-1}, in^{-1})$ to (τ_0, τ_0', g, g').

By Theorem 3.25, for all types α, $in^{-1}\colon \omega(\alpha) \multimap \sigma(\alpha, \omega(\alpha))$ is a final coalgebra for the functor $\beta \mapsto \sigma(\alpha, \beta)$, and $out^{-1}\colon \sigma(\tau, \tau') \multimap \tau'$ is an initial algebra for the functor $\alpha \mapsto \sigma(\omega(\alpha), \alpha)$.

Let a be the unique map making the diagram

$$\begin{tikzcd} \tau_0' \arrow[r, "g'"] \arrow[d, "a"'] & \sigma(\tau_0, \tau_0') \arrow[d, "\sigma(id,a)"] \\ \omega(\tau_0) \arrow[r, "in^{-1}"] & \sigma(\tau_0, \omega(\tau_0)) \end{tikzcd}$$

commute. Define h to be the unique map making

$$\begin{tikzcd} \sigma(\tau, \tau') \arrow[rr, "out^{-1}"] \arrow[d, "\sigma(\omega(h),h)"'] & & \tau' \arrow[d, "h"] \\ \sigma(\omega(\tau_0), \tau_0) \arrow[r, "\sigma(a,id)"] & \sigma(\tau_0', \tau_0) \arrow[r, "g"] & \tau_0 \end{tikzcd} \tag{6}$$

commute. We define h' to be $\omega(h) \circ a$ and prove that (h, h') is a map of dialgebras. The first diagram of Definition 3.21 is simply (6). Commutativity of the second diagram follows from commutativity of

$$\begin{tikzcd} \tau_0' \arrow[r, "g'"] \arrow[d, "a"'] & \sigma(\tau_0, \tau_0') \arrow[d, "\sigma(id,a)"] \\ \omega(\tau_0) \arrow[r, "in^{-1}"] \arrow[d, "\omega(h)"'] & \sigma(\tau_0, \omega(\tau_0)) \arrow[d, "\sigma(h,\omega(h))"] \\ \omega(\tau') \arrow[r, "in^{-1}"] & \sigma(\tau', \omega(\tau')), \end{tikzcd} \tag{7}$$

where commutativity of the last diagram follows from Lemma 3.20.

Finally, we will show that if (k, k') is another map of dialgebras from $(\tau', \tau, out^{-1}, in^{-1})$ to (τ_0, τ'_0, g, g') then $h =_{\tau' \multimap \tau_0} k$ and $h' =_{\tau'_0 \multimap \tau} k'$. By the second diagram of Definition 3.21 we know that

$$\begin{tikzcd}
\tau'_0 \arrow[r, "g'"] \arrow[d, "k'"'] & \sigma(\tau_0, \tau'_0) \arrow[r, "\sigma(k,id)"] & \sigma(\tau', \tau'_0) \arrow[d, "\sigma(id,k')"] \\
\tau \arrow[rr, "in^{-1}"] & & \sigma(\tau', \tau)
\end{tikzcd} \tag{8}$$

commutes. Clearly, if we substitute k for h in (7), we obtain a diagram that commutes by Lemma 3.20. So, using the fact that in^{-1} is a final coalgebra on (8), we get $k' =_{\tau'_0 \multimap \tau} \omega(k) \circ a$.

The first diagram of Definition 3.21 implies that

$$\begin{tikzcd}
\sigma(\tau, \tau') \arrow[rr, "out^{-1}"] \arrow[d, "\sigma(\omega(k),k)"'] & & \tau' \arrow[d, "k"] \\
\sigma(\omega(\tau_0), \tau_0) \arrow[r, "\sigma(a,id)"] & \sigma(\tau'_0, \tau_0) \arrow[r, "g"] & \tau_0
\end{tikzcd}$$

commutes. Comparing this to (6) we obtain $h =_{\tau' \multimap \tau_0} k$, by initiality of out^{-1}. $\square$

Theorem 3.27. *Assuming identity extension and extensionality, for all types $\sigma(\alpha, \beta)$ where α occurs only negatively and β only positively, there exists a type τ and a map $f\colon \sigma(\tau, \tau) \multimap \tau$, such that (τ, τ, f, f^{-1}) is an initial dialgebra up to internal equality.*

Proof. As usual commutativity of diagrams will be up to internal equality.

We have a unique map of dialgebras

$$(h, h')\colon (\tau, \tau', in, out) \to (\tau', \tau, out^{-1}, in^{-1})$$

We claim that (h', h) is also a map of dialgebras from (τ, τ', in, out) to $(\tau', \tau, out^{-1}, in^{-1})$. To prove this we need to prove commutativity of the diagrams

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[r, "in"] \arrow[d, "\sigma(h,h')"'] & \tau \arrow[d, "h'"] \\
\sigma(\tau, \tau') \arrow[r, "out^{-1}"] & \tau'
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau \arrow[r, "in^{-1}"] \arrow[d, "h"'] & \sigma(\tau', \tau) \arrow[d, "\sigma(h',h)"] \\
\tau' \arrow[r, "out"] & \sigma(\tau, \tau')
\end{tikzcd}
\quad ,$$

but the fact that (h, h') is a map of dialgebras tells us exactly that

$$\begin{tikzcd}
\sigma(\tau', \tau) \arrow[r, "in"] \arrow[d, "\sigma(h',h)"'] & \tau \arrow[d, "h"] \\
\sigma(\tau, \tau') \arrow[r, "out^{-1}"] & \tau'
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau \arrow[r, "in^{-1}"] \arrow[d, "h'"'] & \sigma(\tau', \tau) \arrow[d, "\sigma(h,h')"] \\
\tau' \arrow[r, "out"] & \sigma(\tau, \tau')
\end{tikzcd},$$

and these two diagram are the same as the above but in opposite order. Thus, by uniqueness of maps of dialgebras out of (τ, τ', in, out), we get $h =_{\tau \multimap \tau'} h'$. Since (h, h) is a map between initial dialgebras, h is an isomorphism.

Now define $f\colon \sigma(\tau,\tau) \multimap \tau$ to be $in \circ \sigma(h^{-1}, id_\tau)$. Then clearly (id_τ, h^{-1}) is a morphism of dialgebras from (τ,τ,f,f^{-1}) to (τ,τ',in,out), since the diagrams proving (id_τ, h^{-1}) to be a map of dialgebras are

$$\begin{tikzcd}
\sigma(\tau,\tau) \arrow[r, "\sigma(h^{-1},id)"] \arrow[rr, bend right, "f"'] \arrow[d, "\sigma(h^{-1},id)"'] & \sigma(\tau',\tau) \arrow[r, "in"] & \tau \arrow[d, "id"] \\
\sigma(\tau',\tau) \arrow[rr, "in"] & & \tau
\end{tikzcd}
\qquad
\begin{tikzcd}
\tau' \arrow[rr, "out"] \arrow[d, "h^{-1}"'] & & \sigma(\tau,\tau') \arrow[d, "\sigma(id,h^{-1})"] \\
\tau \arrow[r, "in^{-1}"] \arrow[rr, bend right, "f^{-1}"'] & \sigma(\tau',\tau) \arrow[r, "\sigma(h,id)"] & \sigma(\tau,\tau).
\end{tikzcd}$$

Clearly the first diagram commutes, and the second diagram is just part of the definition of (h,h) being a map of dialgebras. Thus (id_τ, h^{-1}) defines an isomorphism of dialgebras from (τ,τ,f,f^{-1}) to (τ,τ',in,out), as desired. $\square$

Corollary 3.28. *Assuming identity extension and extensionality, for all types $\alpha,\beta \vdash \sigma(\alpha,\beta)$, where α occurs only negatively and β only positively, there exists a type τ such that $\sigma(\tau,\tau) \cong \tau$ in each $\mathbf{LinType}_\Xi$.*

Proof. The isomorphism is $in \circ \sigma(h^{-1}, id)$. $\square$

Notice that the closed terms $\tau \multimap \sigma(\tau,\tau)$ and $\sigma(\tau,\tau) \multimap \tau$ always exist, independent of the assumption of parametricity. We use parametricity to prove that they are each others inverses.

4 LAPL-structures

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

First, however, we call to mind what a model of PILL is and how PILL is interpreted in such a model (for a full description of models for PILL and interpretations in these, see e.g. [17, 15, 2, 12, 5]).

A model of PILL is a fibred symmetric monoidal adjunction

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

such that $\mathbf{LinType}$ is fibred symmetric monoidal closed; the tensor in $\mathbf{Type}$ is a fibred cartesian product; $\mathbf{Type}$ is equivalent to the category of finite products of free coalgebras for the comonad FG on $\mathbf{LinType}$; $\mathbf{Kind}$ is cartesian; p has a generic object and simple products with respect to projections forgetting Ω, where Ω is p of the generic object. See [15] for detailed explanation of this definition.

PILL is interpreted in such models as follows. A type σ is interpreted as an object $[\![\sigma]\!] \in \mathbf{LinType}$ using the SMCC structure to interpret $\otimes, \multimap, I$ and the comonad FG to interpret !, and we interpret a term

$$\vec{\alpha} \mid \vec{x}\colon \vec{\sigma}; \vec{x}'\colon \vec{\sigma}' \vdash t\colon \tau$$

as a morphism

$$![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_n]\!] \otimes [\![\sigma'_1]\!] \otimes \ldots \otimes [\![\sigma'_m]\!] \multimap [\![\tau]\!]$$

in $\mathbf{LinType}$, where $! = FG$. Notice that we denote the morphisms in $\mathbf{LinType}$ by $\multimap$.

The comonad structure on $\mathbf{LinType}$ induced by the adjunction gives us two natural transformations $\delta\colon ! \multimap !!$ and $\epsilon\colon ! \multimap id$. These are defined in the internal language as

$$\delta_\sigma = \lambda^\circ x\colon !\sigma.\, \text{let } !y \text{ be } x \text{ in } !!y,$$
$$\epsilon_\sigma = \lambda^\circ x\colon !\sigma.\, \text{let } !z \text{ be } x \text{ in } z.$$

It turns out that we may interpret the intuitionistic part of the calculus, that is, the terms in the calculus with no free linear variables, in $\mathbf{Type}$. For suppose we are given such a term

$$\Xi \mid \vec{x} \colon \vec{\sigma}; - \vdash t \colon \tau.$$

Then the interpretation of this term in $\mathbf{LinType}$ is

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

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

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

in $\mathbf{Type}$. It is easy to prove that

$$\begin{gathered}[\![\Xi \mid \Gamma; - \vdash s[t/x]]\!]_{\mathbf{Type}} = \\ [\![\Xi \mid \Gamma, x \colon \sigma; - \vdash s \colon \tau]\!]_{\mathbf{Type}} \circ \langle id_{[\![\Xi \mid \Gamma; -]\!]}, [\![\Xi \mid \Gamma; - \vdash t]\!]_{\mathbf{Type}} \rangle,\end{gathered}$$

using Lemma 3.2.2 of [2].

Definition 4.1. A model of PILL$_Y$ is a model of PILL, which models a fixed point operator

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

Definition 4.2. A **pre-LAPL**-structure is

1. a schema of categories and functors

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

   such that

   - the diagram

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

     is a model of PILL$_Y$.
   - q is a fibration with fibred finite products
   - (r, q) is an indexed first-order logic fibration [5] which has products and coproducts with respect to projections $\Xi \times \Omega \to \Xi$ in $\mathbf{Kind}$ [5], where Ω is p applied to the generic object of p.
   - I is a faithful product-preserving map of fibrations.

2. a contravariant morphism of fibrations:

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

3. a family of bijections

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

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

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

$$\Psi(\rho^*(u)) = (\bar{\rho})^*(\Psi(u))$$

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

Notice that Ψ is only defined on vertical morphisms.

By contravariance of the fibred functor U we mean that U is contravariant in each fibre. Since 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. Strictly speaking, we should denote the bijection Ψ by $\Psi_{\Xi,\xi,\sigma,\tau}$ since it depends on all these, but for ease of notation we simply write Ψ.

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

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

is interpreted as

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

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

For notational convenience we shall write $\llbracket\Xi \mid \Gamma \mid \Theta \vdash t\colon \tau\rrbracket$ for the interpretation of t in $\mathbf{Ctx}$, that is for

$$I(\llbracket\Xi \mid \Gamma; - \vdash t\colon \tau\rrbracket_{\mathbf{Type}}) \circ \pi$$

(note the subscript $\mathbf{Type}$), where π is the projection

$$\pi\colon \llbracket\Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket\Xi \mid \Gamma \mid -\rrbracket$$

in $\mathbf{Ctx}_{\llbracket\Xi\rrbracket}$.

The propositions in the logic are interpreted in $\mathbf{Prop}$ as follows.

Let $\Delta_I\colon I \to I \times I$ denote the diagonal map, then

$$\llbracket\Xi \mid x\colon \tau, y\colon \tau \vdash x =_\tau y\rrbracket = \textstyle\coprod_{\Delta_{\llbracket\tau\rrbracket}}(\top),$$

where $\coprod_{\Delta_{[\![\tau]\!]}}$ denotes the left adjoint to reindexing along Δ. Now we can define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash t =_\sigma u]\!] = \langle [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash u]\!] \rangle^* [\![\Xi \mid x\colon \tau, y\colon \tau \vdash x =_\tau y]\!].$$

To interpret $\forall x\colon \sigma_{i_0}.\phi$, recall that a context $\Xi \mid x_1\colon \sigma_1, \ldots, x_n\colon \sigma_n \mid \Theta$ is interpreted as

$$\textstyle\prod_i IG[\![\sigma_i]\!] \times [\![\Theta]\!],$$

where $[\![\sigma_i]\!]$ is the usual interpretation of types in $\mathbf{LinType}$ and the product refers to the fibrewise product in $\mathbf{Ctx}$. We may therefore interpret $\forall x\colon \sigma_{i_0}.\phi$ using the right adjoint to reindexing along the projection

$$\pi\colon \textstyle\prod_i IG[\![\sigma_i]\!] \times [\![\Theta]\!] \to \prod_{i \neq i_0} IG[\![\sigma_i]\!] \times [\![\Theta]\!].$$

Likewise, $\forall R\colon \mathsf{Rel}(\sigma, \tau.)\phi$ is interpreted using right adjoints to reindexing functors related to the appropriate projection in $\mathbf{Ctx}$. The existential quantifiers $\exists x\colon \sigma_{i_0}.\phi$ and $\exists R\colon \mathsf{Rel}(\sigma, \tau.)\phi$ are interpreted using left adjoints to the same reindexing functors.

Quantification over types $\forall \alpha.\phi$ and $\exists \alpha.\phi$ is interpreted using respectively right and left adjoints to $\bar{\pi}^*$ where $\bar{\pi}$ is the lift of the projection $\pi : [\![\Xi, \alpha\colon \mathsf{Type}]\!] \to [\![\Xi]\!]$ in $\mathbf{Kind}$ to $\mathbf{Ctx}$. To be more precise, one may easily show that for $\Xi \mid \Gamma \mid \Theta$ wellformed $[\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] = \pi^*[\![\Xi \mid \Gamma \mid \Theta]\!]$ using the corresponding result for the interpretation of PILL_Y, and so the cartesian lift of π is a map:

$$\bar{\pi}\colon [\![\Xi, \alpha \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma \mid \Theta]\!]$$

and we define

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \forall \alpha.\, \phi]\!] = \textstyle\prod_{\bar{\pi}} [\![\Xi,\ \alpha \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where $\prod_{\bar{\pi}}$ is the right adjoint to $\bar{\pi}^*$.

Definable relations with domain σ and codomain τ in contexts $\Xi \mid \Gamma \mid \Theta$ are interpreted as maps from $[\![\Xi \mid \Gamma \mid \Theta]\!]$ into $U([\![\sigma]\!], [\![\tau]\!])$. The definable relation

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

is interpreted as the projection, and

$$[\![\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi\colon \mathsf{Rel}(\sigma, \tau)]\!] = \Psi^{-1}([\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi]\!]).$$

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

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

Next, if $\Xi \mid \Gamma \vdash t\colon \sigma, s\colon \tau$, then

$$\begin{gathered}[\![\Xi \mid \Gamma \mid \Theta \vdash \rho(t, s)]\!] = \\ \langle \langle \pi, \langle [\![\Xi \mid \Gamma \mid \Theta \vdash t]\!], [\![\Xi \mid \Gamma \mid \Theta \vdash s]\!] \rangle \rangle, \pi' \rangle^* [\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \rho(x, y)]\!],\end{gathered}$$

where π, π' are the projections

$$\pi\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma]\!] \qquad \pi'\colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid - \mid \Theta]\!].$$

One may think of the isomorphism Ψ as a model-theoretic version of Lemma 2.27.

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

We may interpret the logic containing admissible relations by interpreting $S\colon \mathsf{AdmRel}(\sigma,\tau)$ as $V(\llbracket\sigma\rrbracket, \llbracket\tau\rrbracket)$. Admissible relations are interpreted as maps into $V(\sigma,\tau)$. 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 4 and Rule 2.18 holds.

Lemma 4.4. *In the interpretation given above of the subset of LAPL excluding the relational interpretation of types in a pre-LAPL structure modeling admissible relations, if*

$$\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}$$

is a proposition in the logic, and

$$\Xi \mid \Gamma \vdash t\colon \sigma$$

is a term, then

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash \phi[t/x]\colon \mathsf{Prop}\rrbracket = \langle\langle \pi, \llbracket \Xi \mid \Gamma \mid \Theta \vdash t\colon \sigma\rrbracket\rangle, \pi'\rangle^* \llbracket \Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi\colon \mathsf{Prop}\rrbracket,$$

where π, π' are the projections

$$\pi\colon \llbracket \Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket \Xi \mid \Gamma\rrbracket \qquad \pi'\colon \llbracket \Xi \mid \Gamma \mid \Theta\rrbracket \to \llbracket \Xi \mid - \mid \Theta\rrbracket.$$

Proof. By induction on the structure of ϕ. Cases $R(s,s')$ and $s =_\tau s'$ are easy from definitions, simply using the fact that

$$\begin{gathered}\llbracket \Xi \mid \Gamma, x\colon \sigma \vdash s[t/x]\rrbracket_{\mathbf{Type}} = \\ \llbracket \Xi \mid \Gamma, x\colon \sigma \vdash s\colon \tau\rrbracket_{\mathbf{Type}} \circ \langle \pi_{\llbracket \Xi \mid \Gamma; -\rrbracket}, \llbracket \Xi \mid \Gamma \vdash t\rrbracket_{\mathbf{Type}}\rangle\end{gathered}$$

in the PILL model. The cases $\phi \wedge \phi'$, $\phi \supset \phi'$, etc., are just the fact that the fibrewise structure of $\mathbf{Prop}$ is preserved by reindexing, and the cases of the quantifiers is by the Beck-Chevalley condition. $\square$

Lemma 4.5. *For $\llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)\rrbracket\colon \llbracket \Xi \mid \Gamma \mid \Theta\rrbracket \to U(\sigma,\tau)$ and $t\colon \sigma' \multimap \sigma$ and $s\colon \tau' \multimap \tau$,*

$$\llbracket \Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma', y\colon \tau').\, \rho(t\,x, s\,y)\rrbracket\colon \llbracket \Xi \mid \Gamma \mid \Theta\rrbracket \to U(\sigma',\tau') = U(t,s) \circ \llbracket \Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)\rrbracket.$$

Proof. This follows from Lemma 4.4 and naturality of Ψ in σ, τ: Assume for simplicity that Γ and Θ are empty.

Observe $\llbracket \Xi \mid x\colon \sigma' \vdash t\,x\colon \sigma\rrbracket_{\mathbf{Type}} = G\,\sigma' \xrightarrow{\eta_{G\,\sigma'}} GFG\,\sigma' \xrightarrow{G\,\epsilon_{\sigma'}} G\,\sigma' \xrightarrow{G\,t} G\,\sigma = G(t)$, where η is the unit of the adjunction $F \dashv G$. Now

$$\llbracket \Xi \mid - \mid - \vdash (x:\sigma, y:\tau).\rho(t\,x, s\,y)\rrbracket = \Psi_\Xi^{-1}(\llbracket \Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(t\,x, s\,y)\rrbracket)$$

which using Lemma 4.4 and the calculation above gives

$$\begin{gathered}\Psi_\Xi^{-1}((t \times s)^*(\llbracket \Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(x,y)\rrbracket)) = \\ U(t,s) \circ \Psi_\Xi^{-1}(\llbracket \Xi \mid x:\sigma, y:\tau \mid - \vdash \rho(x,y)\rrbracket) = U(t,s) \circ \llbracket \Xi \mid - \mid - \vdash \rho\colon \mathsf{Rel}(\sigma,\tau)\rrbracket.\end{gathered}$$

$\square$

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

$$\begin{tikzcd} \mathbf{LinAdmRelations} \arrow[d] \\ \mathbf{AdmRelCtx} \end{tikzcd},$$

which we think of as a model consisting of admissible relations. We first define the category $\mathbf{AdmRelCtx}$ by the pullback

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

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

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

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

in $\mathbf{Ctx}$. Here π, π' are first and second projection respectively out of $\Xi \times \Xi'$.

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

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

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

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

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

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

Remark 4.6. In the internal language, objects of $\mathbf{LinAdmRelations}$ are admissible relations

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

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

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

holds (this follows directly from Lemma 4.4).

There exist two canonical maps of fibrations:

$$\begin{pmatrix} \mathbf{LinAdmRelations} \\ \Big\downarrow \\ \mathbf{AdmRelCtx} \end{pmatrix} \underset{\partial_1}{\overset{\partial_0}{\rightrightarrows}} \begin{pmatrix} \mathbf{LinType} \\ \Big\downarrow \\ \mathbf{Kind} \end{pmatrix}.$$

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

Lemma 4.7. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has products in the base, a generic object and simple products with respect to projections in* $\mathbf{AdmRelCtx}$ *forgetting the generic object. The maps* ∂_0, ∂_1 *preserve this structure.*

Proof. The category $\mathbf{AdmRelCtx}$ has products:

$$(\Xi_1, \Xi_1' \mid \Theta_1) \times (\Xi_2, \Xi_2' \mid \Theta_2) = \Xi_1 \times \Xi_2, \Xi_1' \times \Xi_2' \mid \pi^*(\Theta_1) \times \pi'^*(\Theta_2)$$

(see [9, Proposition 9.2.1]).

The fibration has a generic object $\Omega, \Omega \mid V(\widehat{id_\Omega}, \widehat{id_\Omega})$, since a morphism into this from $\Xi, \Xi' \mid \Theta$ in $\mathbf{AdmRelCtx}$ consists of pairs of types $(f : \Xi \to \Omega, g : \Xi' \to \Omega)$ and a morphism from Θ to $V(\hat{f}, \hat{g})$.

We now show that we have products with respect to projections forgetting the generic object. Given a relation

$$\Xi, \alpha; \Xi', \beta \mid \Theta, R \colon \mathsf{AdmRel}(\alpha, \beta) \vdash \rho \colon \mathsf{AdmRel}(\tau, \tau')$$

we can define

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

as

$$\forall(\alpha, \beta, R \colon \mathsf{AdmRel}(\alpha, \beta)).\, \rho = (t, u).\, \forall \alpha, \beta \colon \mathsf{Type}.\, \forall R \colon \mathsf{AdmRel}(\alpha, \beta).\, (t\alpha)\rho(u\beta).$$

We will to show that this defines a right adjoint to weakening. Suppose we have another relation

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

We will use the usual adjunction in $\mathbf{LinType}$, where a map $\Xi, \alpha \mid - \vdash t \colon \sigma \multimap \tau$, with $\Xi \vdash \sigma \colon \mathsf{Type}$ corresponds to

$$\Xi \mid - \vdash \hat{t} = \lambda^\circ x \colon \sigma.\, \Lambda\alpha.\, (t\, x) \colon \sigma \multimap \textstyle\prod \alpha.\, \tau.$$

We need to prove that (t, u) preserves relations iff $(\hat{t}, \hat{u})$ does, but it is clear that

$$\Xi, \alpha; \Xi', \beta \mid x : \sigma, y : \sigma' \mid \Theta, R \colon \mathsf{AdmRel}(\alpha, \beta) \mid x\omega y \vdash (t\, x)\rho(u\, y)$$

iff

$$\Xi, \Xi' \mid x : \sigma, y : \sigma' \mid \Theta \mid x\omega y \vdash \forall \alpha, \beta \colon \mathsf{Type}.\, \forall R \colon \mathsf{AdmRel}(\alpha, \beta).\, (\hat{t}\, x\, \alpha)\rho(\hat{u}\, y\, \beta),$$

which establishes the bijective correspondence between maps

$$\frac{\pi^*\omega \multimap \rho}{\omega \multimap \forall(\alpha, \beta, R \colon \mathsf{Rel}(\alpha, \beta)).\, \rho}$$

proving that we have in fact defined a product. $\square$

Lemma 4.8. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has a fibrewise SMCC-structure and the two maps* ∂_0, ∂_1 *are fibred strict symmetric monoidal functors.*

Proof. We prove that the constructions $\otimes$, $\multimap$ on definable relations given in Section 2.2.2 define a fibrewise symmetric monoidal structure on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$. Notice that since V is closed under the rules of Figure 4, Proposition 2.3 tells us that the constructions on definable relations of Section 2.2.2 indeed do define operations on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$.

First we will prove that the two operators $-\otimes-$, $\rho \multimap -$ do in fact define functors on $\mathbf{LinAdmRelations}$. That is, we need to check that if

$$(t_0, s_0) \colon \rho_0 \multimap \rho_0' \quad (t_1, s_1) \colon \rho_1 \multimap \rho_1',$$

then

$$(t_0 \otimes t_1, s_0 \otimes s_1) \colon \rho_0 \otimes \rho_1 \multimap \rho_0' \otimes \rho_1',$$

and, if $(t, s) \colon \rho' \multimap \rho''$, then

$$(t \circ -, s \circ -) \colon (\rho \multimap \rho') \multimap (\rho \multimap \rho'').$$

To see that $\otimes$ defines a functor, suppose $x(\rho_0 \otimes \rho_1)y$ and $f(\rho_0' \multimap \rho_1' \multimap R)g$. We need to show that

$$R(\text{let } z \otimes z' \text{ be } (t_0 \otimes t_1)(x) \text{ in } f\ z\ z', \text{let } z \otimes z' \text{ be } (s_0 \otimes s_1)(y) \text{ in } g\ z\ z').$$

Recall that $(t_0 \otimes t_1)x = \text{let } \omega \otimes \omega' \text{ be } x \text{ in } t_0\ \omega \otimes t_1\ \omega'$ in PILL. Notice then that

$$\text{let } z \otimes z' \text{ be } (t_0 \otimes t_1)(y) \text{ in } f\ z\ z' = \text{let } z \otimes z' \text{ be } y \text{ in } f\ (t_0(z))\ (t_1(z')),$$

and

$$(\lambda^\circ z, z'. f\ (t_0(z))\ (t_1(z')))(\rho_0 \multimap \rho_1 \multimap R)(\lambda^\circ z, z'. g\ (s_0(z))\ (s_1(z'))).$$

The result now follows from the assumption that $x(\rho_0 \otimes \rho_1)y$.

To prove that $\rho \multimap -$ is a functor, suppose $(f, g) \colon \rho \multimap \rho'$ and $\rho(x, y)$. Then clearly $\rho''(t \circ f(x), s \circ g(y))$, as required.

We need to show that $\rho \multimap -$ is right adjoint to $- \otimes \rho$, and that the adjoint components are natural in ρ. Since we are given a similar adjunction in $\mathbf{LinType}$, all we need to show is that

$$(t, s) \colon \rho \multimap (\rho' \multimap \rho'')$$

iff

$$(\hat{t}, \hat{s}) \colon \rho \otimes \rho' \multimap \rho'',$$

where $\hat{t}, \hat{s}$ are the maps corresponding to t, s in the adjunction on $\mathbf{LinType}$. Suppose first that

$$(t, s) \colon \rho \multimap (\rho' \multimap \rho'') \text{ and } x(\rho \otimes \rho')y.$$

The definition of the latter says exactly that, for all $(t, s) \colon \rho \multimap (\rho' \multimap \rho'')$, we must have $\rho''(\hat{t}\ x, \hat{s}\ y)$.

Now, suppose $(\hat{t}, \hat{s}) \colon \rho \otimes \rho' \multimap \rho''$ and $x \rho y \wedge x' \rho' y'$. By Lemma 2.29 $\rho \otimes \rho'(x \otimes x', y \otimes y')$ and so $\rho''(\hat{t}(x \otimes x'), \hat{s}(y \otimes y'))$. Hence, since $\hat{t}(x \otimes x') = t\ x\ x'$ (likewise for s), we are done.

We now proceed to prove that the functors $- \otimes -, \rho \multimap -$ define a fibred SMCC structure on

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

The unit in a fibre is $I_{Rel}\colon \mathsf{AdmRel}(I, I)$ where I is the unit in the appropriate fibres of $\mathbf{LinType}$. The maps giving the isomorphisms

$$(-) \otimes ((=) \otimes (\equiv)) \cong ((-) \otimes (=)) \otimes (\equiv),$$
$$(-) \otimes I \cong (-), \quad (-) \otimes (=) \cong (=) \otimes (-)$$

are simply pairs of the corresponding maps in the fibrewise SMC-structure of $\mathbf{LinType}$. These maps satisfy the coherence properties simply because the maps in $\mathbf{LinType}$ do the same. One has to check that the maps defined by pairing maps in fact define maps in $\mathbf{LinAdmRelations}$, i.e., that they preserve relations.

One direction of the isomorphism $\sigma \otimes I \cong \sigma$ is given by the map $\lambda^\circ x\colon \sigma.\, x \otimes \star$. To see that this preserves relations, suppose $x \rho y$. Since $\star I_{Rel} \star$, $(x \otimes \star)(\rho \otimes I_{Rel})(y \otimes \star)$ by Lemma 2.29. For the other direction, consider $f_\sigma\colon \sigma \multimap I \multimap \sigma$ given as $f = \lambda^\circ x\colon \sigma \lambda^\circ x'\colon I.$ let $\star$ be x' in x. Then the map $\sigma \otimes I \multimap \sigma$ is simply $\hat{f}_\sigma$, and we have proved earlier that it now suffices to prove that f_σ preserves relations. So suppose $x \rho y \wedge x' I_{Rel} y'$. By definition of $x' I_{Rel} y'$ we can conclude that $\rho(f_\sigma\ x\ x', f_\tau\ y\ y')$.

The isomorphism $(\rho \otimes \rho') \otimes \rho'' \cong \rho \otimes (\rho' \otimes \rho'')$ is obtained using the adjunction as follows

$$\cfrac{\cfrac{\cfrac{(\rho \otimes \rho') \otimes \rho'' \multimap \rho'''}{\rho \multimap \rho' \multimap \rho'' \multimap \rho'''}}{\rho \multimap (\rho' \otimes \rho'') \multimap \rho'''}}{\rho \otimes (\rho' \otimes \rho'') \multimap \rho'''}$$

and it is easily seen that this isomorphism is given by pairs of the usual maps in $\mathbf{LinType}$. Likewise the isomorphism $\rho \otimes \rho' \cong \rho' \otimes \rho$ comes from

$$\cfrac{\cfrac{\cfrac{\rho \otimes \rho' \multimap \rho''}{\rho \multimap \rho' \multimap \rho''}}{\rho' \multimap \rho \multimap \rho''}}{\rho' \otimes \rho \multimap \rho''.}$$

By construction, the functors ∂_0, ∂_1 are fibred strict symmetric monoidal functors. □

Lemma 4.9. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has a fibred comonad structure induced by the functor* $\rho \mapsto !\rho$. *The maps* ∂_0, ∂_1 *map this comonad to the fibred comonad* $!$ *on the nose.*

Proof. We need to check that $!$ defines a functor, i.e., that if $(f, g)\colon \rho \multimap \rho'$, then $(!f, !g)\colon !\rho \multimap !\rho'$. It is easy to see that

$$\forall x, y.\, !\rho(!x, !y) \supset !\rho'((!f)(!x), (!g)(!y))$$

since $(!f)(!x) = !(f(x))$. Now the result follows from Rule 2.18.

The comonad maps are given by $(\epsilon, \epsilon)\colon !\rho \multimap \rho$ and $(\delta, \delta)\colon !\rho \multimap !!\rho$. These preserve relations by Lemma 2.32, and the commutative diagrams of a fibred comonad are preserved since they hold for the fibred comonad on $\mathbf{LinType} \to \mathbf{Kind}$. □

Lemma 4.10. *The fibration* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *has natural transformations*

$$d\colon !(-) \Rightarrow !(-) \otimes !(-), e\colon !(-) \Rightarrow I_{Rel}$$

making it a fibred linear fibration. The maps ∂_0, ∂_1 *preserve this structure on the nose.*

Proof. The maps d, e are given by (d, d) and (e, e), which preserve relations by Lemma 2.33. The necessary diagrams commute by the same diagrams for the fibred comonad on $\mathbf{LinType} \to \mathbf{Kind}$, and the functors ∂_0, ∂_1 preserve the structure on the nose by construction. $\square$

If we define $\mathbf{AdmRelations}$ to be the category of finite products of coalgebras [15], we obtain a PILL-model

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

and two maps of PILL-models ∂_0, ∂_1. This model need not be a PILL_Y-model, since for pre-LAPL-structures Y does not necessarily preserve relations.

Definition 4.11. An **LAPL-structure** is a pre-LAPL-structure modeling admissible relations, together with a map of PILL-models J from

$$\begin{tikzcd}
\mathbf{LinType} \arrow[rr, bend right=10] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{Type} \arrow[ll, bend right=10] \arrow[dl] \\
& \mathbf{Kind} &
\end{tikzcd}$$

to

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

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

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

We need to show how to interpret the rule

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

in LAPL-structures.

Since J preserves products in the base and generic objects, $J([\![\vec{\alpha} \vdash \sigma(\vec{\alpha})]\!])$ is a relation from $\sigma(\vec{\alpha})$ to $\sigma(\vec{\beta})$ in context $[\![\vec{\alpha}; \vec{\beta} \mid \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta})]\!]$. It thus makes sense to define

$$[\![\vec{\alpha}, \vec{\beta} \mid - \mid \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[\vec{R}]]\!]$$

to be $J([\![\vec{\alpha} \mid \sigma(\vec{\alpha})]\!])$, so all we need to do now is to reindex this object. We reindex it to the right Kind context using

$$\langle \vec{\tau}, \vec{\tau}' \rangle : [\![\Xi]\!] \to \Omega^{2n},$$

thus obtaining

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

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

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \sigma[\vec{\rho}] \colon \mathsf{AdmRel}(\sigma(\vec{\tau}), \sigma(\vec{\tau}'))]\!] =$$
$$[\![\Xi \mid - \mid \vec{R} \colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}') \vdash \sigma[\vec{R}]]\!] \circ [\![\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho} \colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')]\!].$$

where by $[\![\Xi \mid \Gamma \mid \Theta \vdash \vec{\rho} \colon \mathsf{AdmRel}(\vec{\tau}, \vec{\tau}')]\!]$ we mean the pairing

$$\langle [\![\Xi \mid \Gamma \mid \Theta \vdash \rho_1]\!], \ldots, [\![\Xi \mid \Gamma \mid \Theta \vdash \rho_n]\!] \rangle.$$

Remark 4.12. To model a version of Linear Abadi & Plotkin Logic for unary or other arities of parametricity as in Remark 2.2, the functor U should, have corresponding arity and the domain and codomain of the bijection Ψ should be changed accordingly. Furthermore instead of considering the fibration of binary relations $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ we should consider a fibration of relations of the appropriate arity.

4.1 Soundness

In this section we prove that the interpretation of LAPL in LAPL-structures is sound. First, we present a series of reindexing lemmas.

Lemma 4.13. *If* $\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \phi \colon \mathsf{Prop}$ *is a proposition in the logic, and*

$$\Xi \mid \Gamma \vdash t \colon \sigma$$

is a term, then

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi[t/x] \colon \mathsf{Prop}]\!] = \langle \langle \pi, [\![\Xi \mid \Gamma \mid \Theta \vdash t \colon \sigma]\!] \rangle, \pi' \rangle^* [\![\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \phi \colon \mathsf{Prop}]\!]$$

where π, π' *are the projections*

$$\pi \colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid \Gamma]\!] \qquad \pi' \colon [\![\Xi \mid \Gamma \mid \Theta]\!] \to [\![\Xi \mid - \mid \Theta]\!].$$

Lemma 4.14. *If* $\Xi \mid \Gamma, x \mid \Theta \vdash \rho \colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation in the logic, and*

$$\Xi \mid \Gamma \vdash t \colon \sigma$$

is a term, then

$$[\![\Xi \mid \Gamma, x \mid \Theta \vdash \rho]\!] \circ \langle \langle \pi, [\![\Xi \mid \Gamma \mid \Theta \vdash t \colon \sigma]\!] \rangle, \pi' \rangle = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho[t/x]]\!].$$

Notice that Lemma 4.13 differs from Lemma 4.4 since the latter only concerns the interpretation of the part of the logic not including the relational interpretation of types.

Proof. The two lemmas above are proved simultaneously. We only include the proof of the former, for which we only need to extend the proof of Lemma 4.4 to the case of $\rho(s, u)$. But this follows easily by induction using the latter lemma. $\square$

Lemma 4.15. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi \colon \mathsf{Prop}$ *then*

$$[\![\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \phi \colon \mathsf{Prop}]\!] = \pi^* [\![\Xi \mid \Gamma \mid \Theta \vdash \phi \colon \mathsf{Prop}]\!],$$

where π *is the obvious projection. Likewise, if* $\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)$ *then*

$$[\![\Xi \mid \Gamma, x \colon \sigma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)]\!] = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho \colon \mathsf{Rel}(\sigma, \tau)]\!] \circ \pi,$$

where π *is the obvious projection.*

Proof. The lemma can be proved in a way similar to Lemmas 4.13 and 4.15. $\square$

Lemma 4.16. *If* $\Xi \vdash \sigma\colon \mathsf{Type}$ *then*

$$[\![\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \vdash \phi[\sigma/\alpha]]\!] = \overline{\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle}^{*}[\![\Xi, \alpha\colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash \phi]\!],$$

and

$$[\![\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha] \vdash \rho[\sigma/\alpha]]\!] = \overline{\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle}^{*}[\![\Xi, \alpha\colon \mathsf{Type} \mid \Gamma \mid \Theta \vdash \rho]\!],$$

where the vertical line in $\overline{\langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle}$ *denotes the cartesian lift.*

Proof. We know that

$$[\![\Xi \mid \Gamma[\sigma/\alpha] \mid \Theta[\sigma/\alpha]]\!] = \langle id_{[\![\Xi]\!]}, [\![\sigma]\!]\rangle^{*}[\![\Xi, \alpha\colon \mathsf{Type} \mid \Gamma \mid \Theta]\!]$$

since the corresponding statement holds in the PILL-model and the functors F, G, I commute with reindexing.

Now one proceeds by simultaneous induction on ϕ and ρ. For $\rho = R$ and for $\rho = \tau[\vec{\rho}]$ one uses that Ψ commutes with reindexing. For $\phi = u =_\tau u'$ one uses the Beck-Chevalley condition, as is also done for the cases of $\exists$ and $\forall$. The remaining cases either follow by induction or from the fact that the fibrewise structure in $\mathbf{Prop}$ ($\supset, \wedge$, etc.) is preserved by reindexing.

□

Lemma 4.17. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi$ *then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!] = \overline{\pi}^{*}_{\Xi,\alpha\to\Xi}[\![\Xi, \alpha \mid \Gamma \mid \Theta \vdash \phi]\!].$$

Likewise, if If $\Xi \mid \Gamma \mid \Theta \vdash \rho$ *then*

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!] = \overline{\pi}^{*}_{\Xi,\alpha\to\Xi}[\![\Xi, \alpha \mid \Gamma \mid \Theta \vdash \rho]\!].$$

Proof. By simultaneous induction. □

Lemma 4.18. *If* $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation and*

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

then

$$[\![\Xi \mid \Gamma \mid \Theta \vdash \phi[\rho/R]]\!] = (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\rho]\!]\rangle)^{*}[\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau, \tau') \vdash \phi]\!].$$

Likewise, If $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\tau, \tau')$ *is a definable relation and*

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

then

$$[\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\tau, \tau') \vdash \rho']\!] \circ (\langle id_{[\![\Xi|\Gamma|\Theta]\!]}, [\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!]\rangle) = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho'[\rho/R]]\!].$$

The same holds for substitution of admissible relations.

Proof. By simultaneous induction on ϕ and ρ', using naturality of Ψ, Beck-Chevalley and the fact that the fibrewise structure in Prop is preserved by reindexing. □

Lemma 4.19. *If* $\Xi \mid \Gamma \mid \Theta \vdash \phi$ *is a proposition then*

$$[\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \vdash \phi]\!] = \pi^*[\![\Xi \mid \Gamma \mid \Theta \vdash \phi]\!],$$

where π *is the obvious projection. Likewise, if* $\Xi \mid \Gamma \mid \Theta \vdash \rho\colon \mathsf{Rel}(\sigma',\tau')$ *is a definable relation then*

$$[\![\Xi \mid \Gamma \mid \Theta, R\colon \mathsf{Rel}(\sigma,\tau) \vdash \rho]\!] = [\![\Xi \mid \Gamma \mid \Theta \vdash \rho]\!] \circ \pi,$$

where π *is the obvious projection. The same holds for substitution of admissible relations.*

Proof. Again by simultaneous induction. □

Theorem 4.20 (Soundness). *The interpretation given above of LAPL in LAPL-structures is sound with respect to the Rules and Axioms 2.9-2.26.*

Proof. Rules 2.9-2.16 hold since the interpretation of quantification is given by adjoints to weakening, considering Lemmas 4.15, 4.17, 4.19 above.

Rules 2.4-2.7 hold since substitution corresponds to reindexing as in the lemmas above.

Rule 2.8 is proved exactly as in [5].

Rule 2.17 holds since externally equal maps are interpreted equally in the model, by soundness of the interpretation of PILL$_Y$. Clearly internal equality is an equivalence relation.

Rule 2.18 is required to hold in Definition 4.3.

Rules 2.19-2.24 all hold since J preserves SMCC-structure, generic objects, simple products and !.

To prove soundness of Rule 2.25, it suffices to prove soundness of

$$\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \mid \top \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(x,y) \supset\subset \phi,$$

but

$$[\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash ((x\colon \sigma, y\colon \tau).\, \phi)(x,y)]\!] = \Psi([\![\Xi \mid \Gamma \mid \Theta \vdash (x\colon \sigma, y\colon \tau).\, \phi]\!]) = \\ \Psi \circ \Psi^{-1}([\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi]\!]) = [\![\Xi \mid \Gamma, x\colon \sigma, y\colon \tau \mid \Theta \vdash \phi]\!].$$

To prove Axiom 2.26, notice that J is required to be a functor. This means that it maps $[\![Y]\!]\colon I \multimap [\![\prod \alpha.\, (\alpha \to \alpha) \to \alpha]\!]$ to a morphism from I_{Rel} to the relational interpretation of $\prod \alpha.\, (\alpha \to \alpha) \to \alpha$. By the requirement, that $(J, \partial_0, \partial_1)$ is a reflexive graph, this map must be $([\![Y]\!], [\![Y]\!])$. Since $\star I_{Rel} \star$ and $[\![Y]\!](\star) = Y$ we get $Y(\prod \alpha.\, (\alpha \to \alpha) \to \alpha)Y$. □

4.2 Completeness

Theorem 4.21 (Completeness). *There exists an LAPL-structure with the property that any formula of LAPL over pure PILL$_Y$ holds in this model iff it is provable in LAPL.*

Proof. We construct the LAPL-structure syntactically, giving the categories in question the same names as in the diagrams of the definitions of pre-LAPL- and LAPL-structures.

- The category $\mathbf{Kind}$ has as objects sequences of the form

  $$\alpha_1\colon \mathsf{Type}, \dots, \alpha_n\colon \mathsf{Type},$$

  where we identify these contexts up to renaming (in other words, we may think of objects as natural numbers). A morphism from Ξ into

  $$\alpha_1\colon \mathsf{Type}, \dots, \alpha_n\colon \mathsf{Type}$$

  is a sequence of types $(\sigma_1, \dots, \sigma_n)$ such that all σ_i are well-formed in context Ξ.

- Objects in the fibre of $\mathbf{LinType}$ over Ξ are well-formed types in this context. Morphisms in this fibre from σ to τ are equivalence classes of terms t such that $\Xi \mid -; x\colon \sigma \vdash t\colon \tau$, where we identify terms up to external equality. Equivalently, we may think of morphisms as terms $\Xi \mid -; - \vdash t\colon \sigma \multimap \tau$. Composition is by substitution, and reindexing with respect to morphisms in $\mathbf{Kind}$ is by substitution.

- Objects in the fibre of $\mathbf{Type}$ over Ξ are well-formed sequences of types in this context. Morphism in this fibre from $\sigma_1, \ldots, \sigma_n$ to $\tau_1, \ldots, \tau_m$ are equivalence classes of sequences of terms $(t_i)_{i \leq m}$, such that for each i the term
$$\Xi \mid \vec{x}\colon \vec{\sigma}; - \vdash t_i\colon \tau_i$$
is well-formed, where the sequences (t_i) and (t'_i) are identified if, for each i, t_i is externally equal to t'_i. Reindexing with respect to morphisms in $\mathbf{Kind}$ is by substitution.

- The functor $\mathbf{LinType} \to \mathbf{Type}$ maps a morphism $-; x\colon \sigma \vdash t\colon \tau$ to $x\colon \sigma; - \vdash t\colon \tau$. The functor going the other way maps a sequence of objects (σ_i) to $\otimes_i !\sigma_i$. It maps a morphism (t_i) from (σ_i) to (τ_i) to
$$\Xi \mid -; y\colon \otimes_i !\sigma_i \vdash \text{let } \otimes_i x'_i\colon \otimes_i !\sigma_i \text{ be } y \text{ in let } !\vec{x} \text{ be } \vec{x}' \text{ in } \otimes_i !t_i.$$
For further details of the term model for PILL see [2].

- The category $\mathbf{Ctx}$ has as objects in the fibre over Ξ well-formed contexts of LAPL: $\Xi \mid \Gamma \mid \Theta$. A vertical morphism from $\Xi \mid \Gamma \mid \Theta$ to
$$\Xi \mid \Gamma' \mid R_1\colon \mathsf{Rel}(\sigma_1, \tau_1), \ldots, R_n\colon \mathsf{Rel}(\sigma_n, \tau_n), S_1\colon \mathsf{AdmRel}(\sigma'_1, \tau'_1), \ldots, S_m\colon \mathsf{AdmRel}(\sigma'_m, \tau'_m)$$
is a triple, consisting of a morphism $\Xi \mid \Gamma \to \Xi \mid \Gamma'$ in the sense of morphisms in $\mathbf{Type}$, a sequence of definable relations $(\rho_1, \ldots, \rho_n)$, and a sequence of admissible relations $(\omega_1, \ldots, \omega_m)$, such that $\Xi \mid \Gamma \mid \Theta \vdash \rho_i\colon \mathsf{Rel}(\sigma_i, \tau_i)$ and $\Xi \mid \Gamma \mid \Theta \vdash \omega_i\colon \mathsf{AdmRel}(\sigma'_i, \tau'_i)$. We identify two such morphisms represented by the same type morphism and the definable relations $(\rho_1, \ldots, \rho_n)$ and $(\rho'_1, \ldots, \rho'_n)$ and admissible relations $(\omega_1, \ldots, \omega_m)$ and $(\omega'_1, \ldots, \omega'_m)$, respectively, if, for each i, j, the formulas $\rho_i \equiv \rho'_i$ and $\omega_j \equiv \omega'_j$ are provable in the logic, where, as usual, $\rho_i \equiv \rho'_i$ is short for
$$\forall x\colon \sigma_i, y\colon \tau_i.\, \rho_i(x_i, y_i) \supset\subset \rho'_i(x_i, y_i),$$
and likewise for $\omega_j \equiv \omega'_j$. The inclusion functor I is the obvious one. Reindexing is by substitution.

- The fibre of the category $\mathbf{Prop}$ over a context $\Xi \mid \Gamma \mid \Theta$ has as objects formulas in that context, where we identify two formulas if they are provably equivalent. These are ordered by the implication in the logic. Reindexing is done by substitution, that is, reindexing with respect to lifts of morphisms from $\mathbf{Kind}$ is done by substitution in type-variables, whereas reindexing with respect to vertical maps in $\mathbf{Ctx}$ is by substitution in term variables and relation variables.

An easy fibred version of the completeness proof in [2] shows that $\mathbf{Kind}, \mathbf{Type}, \mathbf{LinType}$ together with the functors described above form a PILL$_Y$ model. The fibration $\mathbf{Ctx} \to \mathbf{Kind}$ clearly has fibred products formed by appending contexts, and the inclusion functor I is clearly faithful and product-preserving.

We need to prove that $\mathbf{Prop} \to \mathbf{Ctx} \to \mathbf{Kind}$ is an indexed first-order logic fibration with products and coproducts with respect to simple projections in $\mathbf{Kind}$. The fibrewise bicartesian structure is given by $\vee, \wedge, \supset, \bot, \top$. Fibred simple products and coproducts are given by quantifying over relations and variables, simple products in the composite fibration is given by quantifying over types. We can in fact prove that the composite fibration has all indexed products and coproducts (in particular, that it has equality).

Suppose $(\vec{t}, \vec{\rho})$ represents a morphism from $\Xi \mid \vec{x}\colon \vec{\sigma} \mid \vec{R}$ to $\Xi \mid \vec{y}\colon \vec{\tau} \mid \vec{S}$ (the vectors $\vec{R}, \vec{S}$ consist of both definable and admissible relations, and the vector $\vec{\rho}$ is a concatenation of the corresponding vectors of admissible and definable relations from the definition above). We can then define the product functor in $\mathbf{Prop}$ as:

$$\prod_{(\vec{t},\vec{\rho})}(\Xi \mid \vec{x} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) =$$
$$\Xi \mid \vec{y} \mid \vec{S} \vdash \forall \vec{x}.\, \forall \vec{R}(\vec{t}\vec{x} = \vec{y} \wedge (\vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S}) \supset \phi(\vec{x}, \vec{R})).$$

We define coproduct as:

$$\coprod_{(\vec{t},\vec{\rho})}(\Xi \mid \vec{x} \mid \vec{R} \vdash \phi(\vec{x}, \vec{R})) =$$
$$\Xi \mid \vec{y} \mid \vec{S} \vdash \exists \vec{x}.\, \exists \vec{R}.\, \vec{t}\vec{x} = \vec{y} \wedge \vec{\rho}(\vec{x}, \vec{R}) \equiv \vec{S} \wedge \phi(\vec{x}, \vec{R})).$$

We remark that the equality we will actually be using in the model is the obvious

$$(\Xi \mid \Gamma, x\colon \sigma \mid \Theta \vdash \phi) \mapsto (\Xi \mid \Gamma, x\colon \sigma, y\colon \sigma \mid \Theta \vdash \phi \wedge x =_\sigma y)$$

The functor U of item 2 is defined as

$$U(\sigma, \tau) = R\colon \mathsf{Rel}(\sigma, \tau)$$

and

$$U(t\colon \sigma \multimap \sigma', u\colon \tau \multimap \tau') = \Xi \mid R\colon \mathsf{Rel}(\sigma', \tau') \vdash (x\colon \sigma, y\colon \tau).\, R(tx, uy).$$

The required isomorphism Ψ is just the isomorphism given by Lemma 2.27. The functor V is defined as

$$V(\sigma, \tau) = R\colon \mathsf{AdmRel}(\sigma, \tau)$$

and

$$V(t\colon \sigma \multimap \sigma', u\colon \tau \multimap \tau') = \Xi \mid R\colon \mathsf{AdmRel}(\sigma', \tau') \vdash (x\colon \sigma, y\colon \tau).\, R(tx, uy).$$

We have defined a pre-LAPL-structure modeling admissible relations. If we construct $\mathbf{AdmRelCtx}$ as in the definition of LAPL-structure, we obtain:

Objects $\vec{\alpha}, \vec{\beta} \mid \Gamma \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$.

Morphisms A morphism from

$$\vec{\alpha}, \vec{\beta} \mid \Gamma \ \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$$

to

$$\vec{\alpha}', \vec{\beta}' \mid \Gamma' \ \mid \vec{S}\colon \mathsf{AdmRel}(\vec{\omega}(\vec{\alpha}'), \vec{\kappa}(\vec{\beta}')), \vec{S}'\colon \mathsf{Rel}(\vec{\omega}'(\vec{\alpha}'), \vec{\kappa}'(\vec{\beta}'))$$

consists of two morphism in $\mathbf{Kind}$:

$$\vec{\mu} : \vec{\alpha} \to \vec{\alpha}'$$

and

$$\vec{\nu} : \vec{\beta} \to \vec{\beta}',$$

a morphism from $\vec{\alpha}, \vec{\beta} \mid \Gamma$ to $\vec{\alpha}, \vec{\beta} \mid \Gamma'[\vec{\mu}, \vec{\nu}/\vec{\alpha}', \vec{\beta}']$ in $\mathbf{LinType}_{\vec{\alpha},\vec{\beta}}$, and a sequence of admissible relations $\vec{\rho}$ and a sequence of definable relations $\vec{\rho}'$ such that, for all i,j,

$$\vec{\alpha}, \vec{\beta} \mid \Gamma \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta})) \vdash \rho_i\colon \mathsf{AdmRel}(\omega_i(\vec{\mu}), \kappa_i(\vec{\nu}))$$
$$\vec{\alpha}, \vec{\beta} \mid \Gamma' \mid \vec{R}\colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}'\colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta})) \vdash \rho'_j\colon \mathsf{Rel}(\omega'_j(\vec{\mu}), \kappa'_j(\vec{\nu})).$$

As in $\mathbf{Ctx}$ these morphisms are identified up to provable equivalence of the definable relations.

The fibre of $\mathbf{LinAdmRelations}$ over an object $\vec{\alpha}, \vec{\beta} \mid \Gamma \mid R \colon \mathsf{AdmRel}(\vec{\sigma}(\vec{\alpha}), \vec{\tau}(\vec{\beta})), \vec{R}' \colon \mathsf{Rel}(\vec{\sigma}'(\vec{\alpha}), \vec{\tau}'(\vec{\beta}))$ in $\mathbf{AdmRelCtx}$ becomes:

Objects Equivalence classes of definable relations

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

Morphisms A morphism from $\rho \colon \mathsf{AdmRel}(\sigma(\vec{\alpha}), \tau(\vec{\beta}))$ to $\rho' \colon \mathsf{AdmRel}(\sigma'(\vec{\alpha}), \tau'(\vec{\beta}))$ is a pair of morphisms $t : \sigma \multimap \sigma'$, $u : \tau \multimap \tau'$ such that it is provable in the logic that:

$$\forall x \colon \sigma. \forall y \colon \tau. \rho(x, y) \supset \rho'(tx, uy)$$

We will construct the map J as a map of fibred linear categories from $\mathbf{LinType} \to \mathbf{Kind}$ to

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

as follows. On the base categories J is defined on objects as

$$J(\alpha_1, \ldots, \alpha_n) = \alpha_1, \ldots, \alpha_n; \beta_1, \ldots, \beta_n \mid R_1 \colon \mathsf{AdmRel}(\alpha_1, \beta_1), \ldots, R_n \colon \mathsf{AdmRel}(\alpha_n, \beta_n).$$

We define J on the objects of the total categories (and on the morphisms of the base category) as

$$J(\vec{\alpha} \vdash \sigma \colon \mathsf{Type}) = \vec{\alpha}, \vec{\beta} \mid \vec{R} \colon \mathsf{AdmRel}(\vec{\alpha}, \vec{\beta}) \vdash \sigma[R] \colon \mathsf{AdmRel}(\sigma(\vec{\alpha}), \sigma(\vec{\beta})).$$

To define J on morphisms of the fibre categories, suppose $\vec{\alpha} \mid -; - \vdash t \colon \sigma \multimap \tau$. We define $J(t) = (t, t)$. To see that (t, t) in fact is a map from $\sigma[\vec{R}]$ to $\tau[\vec{R}]$, notice that the Logical Relations Lemma (3.3) tells us that

$$\Lambda \vec{\alpha}. t (\prod \vec{\alpha}. \sigma \multimap \tau) \Lambda \vec{\alpha}. t,$$

which means exactly that $(t, t) \colon \sigma[\vec{R}] \multimap \tau[\vec{R}]$.

Rules 2.19-2.24 tell us that J is a strict fibred symmetric monoidal closed functor preserving products and ! on the nose. Since the ϵ and δ of the fibred comonad on $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ are simply (ϵ, ϵ) and (δ, δ) it is clear that J preserve these as well.

Now, by definition, a formula holds in this LAPL-structure iff it is provable LAPL. $\square$

5 Parametric LAPL-structures

Definition 5.1. A **parametric LAPL-structure** is an LAPL-structure with very strong equality in which identity extension hold 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 will be used to conclude that properties proved in the internal logic also hold externally, as exemplified in the following subsection, where we show how to solve recursive domain equations in parametric LAPL-structures.

5.1 Solving recursive domain equations in parametric LAPL-structures

Suppose we are given a fibred functor

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

where by $(-)^{\mathrm{op}}$ we mean taking the opposite category in each fibre. As in [5] we can easily prove that F is isomorphic to a functor whose object part is given by some type $\sigma \in \mathbf{LinType}_{\Omega^2}$, since $\mathbf{LinType}^{\mathrm{op}} \times \mathbf{LinType} \to \mathbf{Kind}$ has as generic object $\Omega \times \Omega$ in $\mathbf{Kind}$. So in the following we shall assume that F is on that form.

Definition 5.2. An endofunctor $T : \mathbb{B}^{\mathrm{op}} \times \mathbb{B} \to \mathbb{B}$, for $\mathbb{B}$ an SMCC, is called **strong** if there exists a natural transformation $t_{\sigma,\tau,\sigma',\tau'} : \sigma^{\sigma'} \otimes (\tau')^{\tau} \to T(\sigma', \tau')^{T(\sigma,\tau)}$ preserving identity and composition:

$$\begin{tikzcd}
I \arrow[r, "\widehat{id_\sigma} \otimes \widehat{id_\tau}"] \arrow[dr, "\widehat{id_{T\sigma}}"'] & \sigma^{\sigma} \otimes \tau^{\tau} \arrow[d, "t_{\sigma,\tau,\sigma,\tau}"] \\
& T(\sigma,\tau)^{T(\sigma,\tau)}
\end{tikzcd}
\qquad
\begin{tikzcd}
(\sigma^{\sigma'} \otimes (\tau')^{\tau}) \otimes ((\sigma')^{\sigma''} \otimes (\tau'')^{\tau'}) \arrow[r, "comp"] \arrow[d, "t \otimes t"'] & \sigma^{\sigma''} \otimes (\tau'')^{\tau} \arrow[d, "t"] \\
T(\sigma',\tau')^{T(\sigma,\tau)} \otimes T(\sigma'',\tau'')^{T(\sigma',\tau')} \arrow[r, "comp"] & T(\sigma'',\tau'')^{T(\sigma,\tau)}.
\end{tikzcd}$$

The natural transformation t is called the **strength** of the functor T. (Note that we here used exponential notation X^Y for the closed structure in $\mathbb{B}$.)

One should note that t in the definition above represents the morphism part of the functor T in the sense that it makes the diagram

$$\begin{tikzcd}
I \arrow[r, "\widehat{f} \otimes \widehat{g}"] \arrow[dr, "\widehat{T(f,g)}"'] & \sigma^{\sigma'} \otimes (\tau')^{\tau} \arrow[d, "t_{\sigma,\tau,\sigma',\tau'}"] \\
& T(\sigma',\tau')^{T(\sigma,\tau)}
\end{tikzcd}$$

commute, for any pair of morphisms $f \colon \sigma' \multimap \sigma, g \colon \tau \multimap \tau'$. This follows from the commutative diagram

$$\begin{tikzcd}
I \arrow[dr, "\widehat{id} \otimes \widehat{id}"] \arrow[drr, bend left, "\widehat{id}"] \arrow[ddr, bend right, "\widehat{f} \otimes \widehat{g}"'] & & \\
& \sigma^{\sigma} \otimes \tau^{\tau} \arrow[r, "t"] \arrow[d, "\sigma^f \otimes g^{\tau}"] & T(\sigma,\tau)^{T(\sigma,\tau)} \arrow[d, "T(f,g)^{T(\sigma,\tau)}"] \\
& \sigma^{\sigma'} (\tau')^{\tau} \arrow[r, "t"] & T(\sigma',\tau')^{T(\sigma,\tau)}.
\end{tikzcd}$$

Definition 5.3. A **strong fibred functor** is a fibred functor

$$\begin{tikzcd}
\mathbb{E}^{\mathrm{op}} \times \mathbb{E} \arrow[rr, "T"] \arrow[dr] & & \mathbb{E} \arrow[dl] \\
& \mathbb{B} &
\end{tikzcd}$$

on a fibred SMCC, for which there exists a fibred natural transformation t from the fibred functor $(=)^{(-)} \otimes (=')^{(-')}$ to $T(-,=')^{T(=,-')}$ satisfying commutativity of the two diagrams of Definition 5.2 in each fibre. The natural transformation t is called the **strength** of the functor T.

Definition 5.4. We will say that the fibred functor

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

assumed to be defined on objects as $\alpha, \beta \mapsto \sigma$ for some $\sigma \in \mathbf{LinType}_{\Omega^2}$ as above, is **polymorphically strong** if there exists a closed term

$$t \colon \prod \alpha, \beta, \alpha', \beta'. (\alpha' \multimap \alpha) \to (\beta \multimap \beta') \to \sigma(\alpha, \beta) \multimap \sigma(\alpha', \beta')$$

such that the family $(t\,\alpha\,\beta\,\alpha'\,\beta')_{(\alpha,\beta,\alpha'\,\beta')\in\mathbf{LinType}^4}$ (each fibre to the 4'th power) is a strength of the functor σ in the sense of Definition 5.3. The term t is called the **polymorphic strength** of F.

For example, the interpretation of any inductively constructed type induces a polymorphically strong fibred functor as described in Section 3.6.

Theorem 5.5. *In a parametric LAPL-structure, for any polymorphically strong fibred functor F there exists a closed type τ such that $F(\tau, \tau) \cong \tau$ in $\mathbf{LinType}_1$ for 1 the terminal object of $\mathbf{Kind}$.*

Proof. The proof of the earlier sections give us this result in the internal language, since we may use identity extension and extensionality in the internal language. The functorial interpretation of types of Section 3.6 should be substituted by the polymorphic strength of the functor F. Since internal equality implies external equality, we get the result in the "real world". □

Remark 5.6. Since the functor F is fibred, we may reindex τ to get a family of objects $(!_\Xi^*\tau)_{\Xi\in\mathbf{Kind}}$ such that for each Ξ, $!_\Xi^*\tau$ satisfies $F(!_\Xi^*\tau, !_\Xi^*\tau) \cong !_\Xi^*\tau$ in the fibre, where $!_\Xi$ is the unique map $\Xi \to 1$ in $\mathbf{Kind}$.

Remark 5.7. Parametric LAPL-structures do not in general model recursive types, that is, we do not have for all types $\sigma \colon \mathbf{LinType}_\Omega$ a type τ such that $\sigma(\tau) \cong \tau$.

6 Concrete Models

In this section we describe a parametric LAPL structure based on admissible pers over a universal domain as advocated by Plotkin [20]. Pers are known to model the typed λ-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 [10].

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 ω-chain-complete partial order such that we have

$$\Phi \colon D \to [D \to D] \qquad \text{and} \qquad \Psi \colon [D \to D] \to D,$$

both Scott-continuous and satisfying

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

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

$$\langle \cdot, \cdot \rangle \colon D \times D \to D, \qquad \pi \colon D \to D \qquad \text{and} \qquad \pi' \colon D \to D,$$

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

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

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

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

strict $\perp_D R \perp_D$,

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

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

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

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

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 $[f] \colon R \to S$ 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* $Classes \colon \mathbf{AP}(D) \to \mathbf{Set}$ *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 to 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

$$d\,1\,d' \Leftrightarrow d = \perp_D = d'.$$

The binary product of R and S is

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

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

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

□

Definition 6.7. The category $\mathbf{AP}(D)_\perp$ of admissible pers and strict continuous functions is the full-on-objects subcategory of $\mathbf{AP}(D)$ with morphisms $[f]\colon R \to S$ 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 π, π', and $\langle\cdot,\cdot\rangle$ are strict. □

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

Proof. The tensor of R and S is

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

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

$$dId' \quad \Leftrightarrow \quad d = d' = \perp_D \vee d = d' = \langle \imath, \perp_D\rangle.$$

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:

$$\begin{array}{rl} d\; I\; d & \Rightarrow \quad d = \perp_D \quad \vee \quad d = \langle \imath, \perp_D\rangle \\ & \Rightarrow \quad \pi(d) = \perp_D \quad \vee \quad \pi(d) = \imath \\ & \Rightarrow \quad \Phi(\pi(d)) = \perp_{[D\to D]} \quad \vee \quad \Phi(\pi(d)) = id_{[D\to D]} \end{array}$$

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

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

$$d\, R \multimap S\, d' \quad \Leftrightarrow \quad d\, S^R\, d' \ \wedge\ (d''\, R\, \bot_D \Rightarrow \Phi(d)(d'')\ S\ \bot_D\ S\ \Phi(d')(d''))$$

The proof consist of a series of straightforward verifications. □

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

Lemma 6.10.

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

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

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

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 ω^{op}-chain

$$\bot = 0 \xrightarrow{!} T0 \xrightarrow{T!} T^2 0 \xrightarrow{T^2!} \cdots$$

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

$$\begin{tikzcd} D_n \arrow[r, bend right, "i_n"'] \arrow[r, phantom, "\triangleleft"] & D \arrow[l, bend right, "j_n"'] \end{tikzcd}$$

Defining $p_n = i_n \circ j_n \colon D \to D$ 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 λ-calculus and calling upon parametricity[2]. 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 [10], specialized to the partial combinatory algebra defined by D, Φ and Ψ.

Let PD denote the power set of D. Define the map $L_0' \colon PD \to PD$ by

$$L_0'(A) = \{d \in D | \pi(\Phi(d)(\imath)) = \imath \wedge \pi'(\Phi(d)(\imath)) \in A\},$$

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

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

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

$$\begin{array}{rcl} d\, L_0(R)\, d & \Rightarrow & d = \perp_D \quad \vee \quad \pi(\Phi(d)(\imath)) = \imath \\ & \Rightarrow & \pi(\Phi(d)(\imath)) = \perp_D \quad \vee \quad \pi(\Phi(d)(\imath)) = \imath \\ & \Rightarrow & \Phi(\pi(\Phi(d)(\imath))) = \perp_{[D \to D]} \quad \vee \quad \Phi(\pi(\Phi(d)(\imath))) = id_{[D \to D]} \end{array}$$

Thus $\Phi(\pi(\Phi(d)(\imath)))(d')$ 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 $\Psi(\lambda d' \in D.\langle \imath, d\rangle) \in L_0(A)$ and if $d \in L_0(A)$ then $\pi'(\Phi(d)(\imath)) \in A$. This is convenient for constructing realizers.

Similarly define, for admissible pers R and S, the map $L_1' \colon \mathcal{F}(R, S) \to \mathcal{F}(L_0(R), L_0(S))_\perp$ by

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

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

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

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

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

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

[2] And calling upon parametricity is, as far as we know, only possible after the deployment of a parametric completion process.

Proof. One first shows that L is left adjoint to U in the ordinary sense. The unit of the adjunction is given by $(\eta_R : R \to UL(R))_{R \in APD_0}$, all tracked by

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

For $f : R \to U(S)$ in $\mathbf{AP}(D)_\perp$, the required unique $h : L(R) \to S$ 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 [15] for an explanation). To this end, we must exhibit an isomorphism $m_I : I \to L(1)$ and a natural isomorphism $m_{R,S} : L(R) \otimes L(S) \to L(R \times S)$. This is mostly straightforward; we just include the definition of $m_{R,S} : L(R) \otimes L(S) \to L(R \times S)$: it is the morphism tracked by the realizer

$$
\begin{array}{l}
\lambda d \in D. \\
\Phi(\pi(\Phi(\pi(d))(1)))(\\
\qquad \Phi(\pi(\Phi(\pi'(d))(1)))(\\
\qquad\qquad \Psi(\lambda d' \in D.\langle 1, \langle \pi'(\Phi(\pi(d))(1)), \pi'(\Phi(\pi'(d))(1))\rangle\rangle) \\
\qquad) \\
)
\end{array}
$$

which reads

"if $\pi(d) \neq \perp$ then
  if $\pi'(d) \neq \perp$ then
    lift of $\langle$unlift$(\pi(d))$, unlift$(\pi'(d))\rangle$
  else $\perp_D$
else $\perp_D$".

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

$$
\begin{array}{l}
\lambda d \in D. \\
\Phi(\pi(\Phi(d)(1)))(\\
\qquad \langle \Psi(\lambda d' \in D.\langle 1, \pi(\pi'(\Phi(d)(1)))\rangle), \Psi(\lambda d' \in D.\langle 1, \pi'(\pi'(\Phi(d)(1)))\rangle)\rangle \\
)
\end{array}
$$

which reads

"if $d \neq \perp$ then
  $\langle$lift of $\pi($unlift$(d))$, lift of $\pi'($unlift$(d))\rangle$
else $\perp_D$".

$\square$

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

$$\begin{tikzcd}
\mathbf{UFam}(\mathbf{AP}(D)_\perp) \arrow[rr, bend right, "U"'] \arrow[dr, "q"'] & \perp & \mathbf{UFam}(\mathbf{AP}(D)) \arrow[ll, bend right, "L"'] \arrow[dl, "p"] \\
& \mathbf{Set}. &
\end{tikzcd} \tag{9}$$

Define the contravariant functor $P : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ 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: $(\alpha_i)_{i\in I} \colon (R_i)_{i\in I} \to (S_i)_{i\in I}$, where, for all $i \in I$, $\alpha_i \in \mathbf{AP}(D)(R_i, S_i)$ and $\exists \alpha \in [D \to D]$. $\forall i \in I.\ \alpha_i = [\alpha]_{\simeq_{R_i,S_i}}$.

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

Define the contravariant functor $Q : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$ 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: $(\alpha_i)_{i\in I} \colon (R_i)_{i\in I} \to (S_i)_{i\in I}$ where for all $i \in I$, $\alpha_i \in \mathbf{AP}(D)_\perp(R_i, S_i)$ and $\exists \alpha \in [D \to D]$. $\forall i \in I.\ \alpha_i = [\alpha]_{\simeq_{R_i,S_i}}$.

For a function $f \colon I \to J$, 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, $p \colon \mathbf{UFam}(\mathbf{AP}(D)) \to \mathbf{Set}$ and $q \colon \mathbf{UFam}(\mathbf{AP}(D)_\perp) \to \mathbf{Set}$. 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. □

6.4 A domain-theoretic model of PILL

To show that (9) is a model of PILL it remains to be shown that q has a generic object and simple products.

Lemma 6.14. *The set $\Omega = \mathrm{Obj}(\mathbf{AP}(D)_\perp) = \mathrm{Obj}(\mathbf{AP}(D))$ is a split generic object of the fibration q. The fibration q has simple split Ω-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 [9] that given any projection $\pi_A \colon A \times \Omega \to A$ in $\mathbf{Set}$, the right adjoint $\forall_A$ to π_A^* is given on objects by intersection:

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

By lemma 6.11 the resulting per is admissible. □

Theorem 6.15. *The diagram* (9) *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 $y \in [[D \to D] \to D]$ giving fixed-points through iterated application at $\perp_D$. Since realizers are Scott-continuous functions in $[D \to D]$, every realizer α 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, α 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 α. 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 $!R \multimap R = R \to R$, 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 $\mathcal{F}(R \to R, R)$, for any R, and thus its code is a member of $\forall\alpha\mathsf{Type}.(\alpha \to \alpha) \multimap \alpha$. Precomposing with a uniform realizer for ϵ before taking the code, one easily obtains the polymorphic fixed-point combinator $Y : \forall\alpha\mathsf{Type}.(\alpha \to \alpha) \to \alpha$. Writing this out, one arrives at

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

For (2), observe that by [15, 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 λ_2 models (as mentioned in the Introduction, we will generalize that completion process to produce parametric LAPL-structures in [14]).

We will arrive at the diagram

$$\begin{tikzcd} \mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[rr, shift right, "U"'] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[ll, shift right, "L"'] \arrow[dl] \\ & \mathbf{PAP}(D) & \end{tikzcd} \tag{10}$$

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 $\mathrm{RefGrph}$ of diagrams

$$A \rightarrowtail R \times S,$$

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

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

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

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

satisfying

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

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

We now describe $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PAP}(D)$ and $\mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PAP}(D)$. 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 $A \rightarrowtail R$ for $A \in \mathrm{Obj}(\mathrm{RegSub}_{\mathbf{AP}(D)_\perp}(R))$.

We plan to use the Grothendieck construction, and so define indexed categories: $(\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ is defined with

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

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

$$\alpha_{\vec{R}}\colon f^p(\vec{R}) \to g^p(\vec{R}).$$

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

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

Furthermore this α should respect relations:

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

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

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

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

$$\alpha_{\vec{R}}\colon U(f^p(\vec{R})) \to U(g^p(\vec{R}))$$

where $U\colon \mathbf{AP}(D)_\perp \to \mathbf{AP}(D)$ 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 α should respect relations:

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

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 $\mathbb{L}\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PFam}(\mathbf{AP}(D)_\perp)$ 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

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

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

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

Lemma 6.18. $\mathbb{L}\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{PFam}(\mathbf{AP}(D)_\perp)$ *and* $\mathbb{U}\colon \mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PFam}(\mathbf{AP}(D))$ *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_η for the unit of $L \dashv U$ also defines a natural transformation $id \Rightarrow \mathbb{UL}$ with the required universal property. □

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. □

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 $f^r(\vec{R}, \vec{S})(\vec{A}) \otimes g^r(\vec{R}, \vec{S})(\vec{A}) \to f^p(\vec{R}) \otimes g^p(\vec{R}) \times f^p(\vec{S}) \otimes g^p(\vec{S})$ 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

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

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* $\mathbf{PFam}(\mathbf{AP}(D)_\perp) \to \mathbf{PAP}(D)$.

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*

$$\begin{array}{c}
(\bigcap f^p)(R_1, \ldots, R_n)(x, y) \iff \\
\bigcap_{R \in \mathrm{Obj}(\mathbf{AP}(D)_\perp)} f^p(R_1, \ldots, R_n, R)(x, y) \wedge \forall R, S, A \rightarrowtail R \times S.\, \langle x, y \rangle f^r(Eq_{R_1}, \ldots, Eq_{R_n}, A) \langle x, y \rangle
\end{array}$$

and

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

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

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

Proof. The construction is as in [9, Section 8.4]. Given a projection $\pi \colon n + m \to n$, we must define a right adjoint to π^*. This is done by extending the construction of the previous lemma in an obvious way to a functor. $\square$

Proposition 6.25. *The diagram (10) constitutes a 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 PILL$_Y$ model can be extended to an LAPL-structure. For this we need just two more fibrations, $q\colon \mathbf{Fam}(\mathbf{Set}) \to \mathbf{PAP}(D)$ and $r\colon \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set})$. The fibre of $\mathbf{Fam}(\mathbf{Set})$ over n has as

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

Morphisms $t : f \to g$ is a family

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

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

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

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

Morphisms There is a morphism $g \to g'$ if

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

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 Ω-products and -coproducts.*

Proof.

$$\begin{tikzcd} \mathrm{Sub}(\mathbf{Set}) \arrow[d] \\ \mathbf{Set} \end{tikzcd}$$

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

$$\begin{tikzcd} \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[r] \arrow[d] & \mathrm{Sub}(\mathbf{Set}) \arrow[d] \\ \mathbf{Set}^{\to} \arrow[r, "\mathrm{dom}"] & \mathbf{Set} \end{tikzcd}$$

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

$$\begin{tikzcd} \mathbf{Fam}(\mathbf{Set}) \arrow[rr, "\simeq"] \arrow[dr] & & \mathbf{Set}^{\to} \arrow[dl, "cod"] \\ & \mathbf{Set} & \end{tikzcd}$$

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

$$\begin{tikzcd}
\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[rr] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{Fam}(\mathbf{Set}) \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbf{Fam}(\mathbf{Set}) \arrow[r, "\simeq"] \arrow[dr] & \mathbf{Set}^{\to} \arrow[d, "cod"] \\
\mathbf{PAP}(D) \arrow[rr] & & \mathbf{Set}
\end{tikzcd}$$

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

$$n \;\mapsto\; obj(\mathbf{AP}(D))^n \qquad (\vec{f^p}, \vec{f^r})\colon n \to m \;\mapsto\; \vec{f^p}\colon obj(\mathbf{AP}(D))^n \to obj(\mathbf{AP}(D))^m$$

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

We can then define the functor $I\colon \mathbf{PFam}(\mathbf{AP}(D)) \to \mathbf{Fam}(\mathbf{Set})$ 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

$$\begin{tikzcd}
\mathbf{PFam}(\mathbf{AP}(D)_\perp)^2 \arrow[rr, "U"] \arrow[dr] & & \mathbf{Fam}(\mathbf{Set}) \arrow[dl] \\
& \mathbf{PAP}(D) &
\end{tikzcd}$$

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 $U(\alpha : f \to f', \beta : g \to g') = \vec{R} \mapsto$

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

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 $(\chi_n)_{n\in \mathrm{Obj}(\mathbf{PAP}(D))}$ such that for all $f, g \in (\mathbf{PFam}(\mathbf{AP}(D)_\perp))_n$ and $M \in (\mathbf{Fam}(\mathbf{Set}))_n$

$$\chi_n\colon \mathbf{Fam}(\mathbf{Set})(M, U_n(f,g))_n \to \mathrm{Obj}(\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set}))_{M\times I_n(\mathbb{U}_n(f)\times \mathbb{U}_n(g))})$$

by

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

Lemma 6.29. *χ 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*

$$\begin{tikzcd}
& & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \arrow[d] \\
\mathbf{PFam}(\mathbf{AP}(D)_\perp) \arrow[r, bend right, "U"'] \arrow[r, phantom, "\perp"] \arrow[dr] & \mathbf{PFam}(\mathbf{AP}(D)) \arrow[l, bend right, "L"'] \arrow[r, hook] \arrow[d] & \mathbf{Fam}(\mathbf{Set}) \arrow[dl] \\
& \mathbf{PAP}(D) &
\end{tikzcd} \tag{11}$$

constitutes a pre-LAPL structure.

Now we define a subfunctor V of U on

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

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 (11) 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_σ: Equality on a type σ is modeled as the diagonal subobject of $[\![\sigma]\!] \times [\![\sigma]\!]$. This corresponds to an admissible relation because it is isomorphic to $[\![\sigma]\!]$ by the continuous functions $\lambda d \in D.\, \langle d, d\rangle$ and π.

$\rho(t\ x, u\ y)$: 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 ρ. Since ρ is chain-complete their limits are related in ρ, and since (t, u) is continuous the limits of the original chains is in the inverse image of the limit in ρ.

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

$x\downarrow \supset \rho(x, y)$: In our model we are allowed to reason classically and thus rephrase the formula to $x = \perp \vee \rho(x, y)$. We recognize this as ρ with the strip $(\perp, y)$ added. This is easily admissible.

$(x\colon !\sigma, y\colon !\tau).\, x\downarrow \supset\subset y\downarrow$: This is all classes except those where exactly one component is $\perp$. Thus we have all the lifted classes and then $(\perp, \perp)$. We recognize this as the lift of truth.

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

$x\downarrow \wedge y\downarrow \supset \phi$: Again we reason classically: Either ϕ holds or it does not. If it holds the formula equals $\top$, if it does not we get all the classes where $x = \perp$ or $y = \perp$. This is admissible since no chain can converge out of $\perp$.

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

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

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

$\square$

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

$$\begin{tikzcd} \mathbf{AdmRelCtx} \arrow[r] \arrow[d, "\langle \partial_0, \partial_1 \rangle"'] & \mathbf{Fam}(\mathbf{Set}) \arrow[d] \\ \mathbf{PAP}(D) \times \mathbf{PAP}(D) \arrow[r, "\times"] & \mathbf{PAP}(D) \end{tikzcd}$$

which means that $\mathbf{AdmRelCtx}$ has as

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

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

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

where Φ and Φ' are evaluated on the combined lists of admissible pers.

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

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

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

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

$$(t\colon f \to f', u\colon g \to g')$$

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

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

Note that we now have two obvious projections ∂_0 and ∂_1.

Finally we can define the required functor J.

$$\begin{pmatrix} \mathbf{PFam}(\mathbf{AP}(D)_\perp) \\ \downarrow \\ \mathbf{PAP}(D) \end{pmatrix} \longrightarrow \begin{pmatrix} \mathbf{LinAdmRelations} \\ \downarrow \\ \mathbf{AdmRelCtx} \end{pmatrix}$$

For the base categories, J is defined on

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

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 α preserves relations.

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

Lemma 6.32. *The tensor product in* $\mathbf{LinAdmRelations} \to \mathbf{AdmRelCtx}$ *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. □

Lemma 6.33. *J is a map of linear λ_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 $!\rho\colon \mathsf{AdmRel}(\sigma, \tau)$:

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

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

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

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* (11) *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 [9], 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 $A \rightarrowtail R \times S$, as we do in LAPL. We will also leave Ψ, Φ implicit, and simply write $f\ x$ for $\Phi(f)(x)$.

We consider the type $\mathrm{Nat} = [\![\prod \alpha. (\alpha \multimap \alpha) \to \alpha \multimap \alpha]\!]$. By definition

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

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

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

The domain of Nat contains the elements $\bot = \lambda f \lambda x. \bot$ 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 $f, g \colon D \to D$ given by $f(d) = \langle d, \iota \rangle$, where ι 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 $x_n = f^n(\bot)$. This sequence is strictly increasing.

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

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

so $n \leq m$. Further,

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

so $m = n$. □

Lemma 6.37. *The per*

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

is a admissible.

Proof. Direct consequence of the lemma above. □

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

Proof. Consider the discrete admissible per D:

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

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

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

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

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

i.e., if $d\, succ\, 0 = \bot$, then $d\, f\, x = \bot$ for all f, x and so $d = \bot$, and if $d\, succ\, 0 = \underline{n}$ 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 $d\, succ\, 0$. □

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

Proof. Analyzing the above proof we see that

$$d = d\, succ\, 0$$

By considering the regular subobject $A \rightarrowtail \mathrm{Nat} \times \mathrm{Nat}$ given by

$$(\bot, \bot) \in A, \qquad \forall n.\, (\underline{n}, \underline{n}) \in A$$

we conclude

$$d\, succ\, 0 = d'\, succ\, 0.$$

□

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, 3.6, 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, 5.1, 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] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'Etat, Université Paris VII, 1972. 1

[8] 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

[9] 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

[10] 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

[11] M. Maietti, P. Maneggia, and E. Ritter. Relating categorical semantics for intuitionistic linear logic. *Applied Categorical Structures*, 2004. To Appear. 1.1

[12] Maria E Maietti, Paola Maneggia, Valeria de Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. Technical Report CSR-01-7, University of Birmingham, School of Computer Science, August 2001. 4

[13] R. E. Møgelberg. *Category theoretic and domain theoretic models of parametric polymorphism*. PhD thesis, IT University of Copenhagen, 2005. 3.3

[14] 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

[15] 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

[16] 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

[17] R.L. Petersen and J. Thamsborg. Polymorphism and linearity all in one pill. Student Project, 2003. 4

[18] B.C. Pierce. *Types and Programming Languages*. MIT Press, 2002. 1

[19] A. M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in computer Science*, 10:321–359, 2000. 1

[20] G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993. 1, 3.9, 6

[21] 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, 3.6

[22] 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

[23] J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983. 1

[24] J.C. Reynolds. Private communication, June 2000. 1

[25] G. Rosolini and A. Simpson. Using synthetic domain theory to prove operational properties of a polymorphic programming language based on strictness. Manuscript, 2004. 1

[26] Izumi Takeuti. An axiomatic system of parametricity. *Fund. Inform.*, 33(4):397–432, 1998. Typed lambda-calculi and applications (Nancy, 1997). 2.2

[27] P. Wadler. The Girard-Reynolds isomorphism (second edition). Manuscript, March 2004. 2.2