Operational Semantics and Models of Linear Abadi-Plotkin Logic

L. Birkedal[1], R.L. Petersen[1], R.E. Møgelberg[1], and C. Varming[1]

IT University of Copenhagen {birkedal, rusmus, mogel, varming}@itu.dk

Abstract. We present a model of Linear Abadi and Plotkin Logic for parametricity [8] based on the operational semantics of LILY, a polymorphic linear lambda calculus endowed with an operational semantics [3]. We use it to formally prove definability of general recursive types in LILY and to derive reasoning principles for the recursive types.

1 Introduction

The combination of parametric polymorphism and recursion on the level of terms yields a type theory expressive enough to solve general recursive type equations[19, 8]. However, as realized by Plotkin [19], for this combination to give a consistent theory, the parametricity principle must be weakened, and so he suggested studying a dual intuitionistic / linear type theory in combination with parametric polymorphism, in which for example the parametricity principle would apply to graphs of linear, but not intuitionistic maps.

Based on these ideas, in 2000 Bierman, Pitts, and Russo [3] presented the programming language LILY, a polymorphic intuitionistic / linear lambda calculus endowed with an operational semantics. In their paper, they sketched how to carry out Plotkin's ideas in the operational setting. The formulation of the parametricity principle was based on the notion of $\top\top$-closed relations, and the idea was to use the strong connection between these and ground contextual equivalence, to show correctness of recursive type encodings up to ground contextual equivalence. In their paper, however, only correctness of the encodings of sum types was proved.

In recent work the first three authors [8, 12] have presented an adaptation of Abadi & Plotkin's logic for parametricity [18] to PILL_Y, a polymorphic dual intuitionistic / linear type theory with fixed points. The resulting logic — called LAPL for Linear Abadi and Plotkin Logic — contains an axiomatized abstract notion of admissible relations on which the formulation of the parametricity principle is based. Admissible relations give the necessary weakening of the parametricity principle. In *loc. cit.* we showed in detail, following Plotkin's suggestions, that LAPL can be used to define a wide range of types, including general nested recursive types. We also defined a sound and complete class of categorical models for LAPL called LAPL-structures,

In this paper we present a concrete LAPL-structure constructed from the operational semantics of LILY. This defines formally an interpretation of LAPL

into LILY, transferring the general results proved abstractly in LAPL to LILY. The construction of the LAPL-structure involves showing that the $\top\top$-closed relations define a notion of admissible relations.

This new model of LAPL based on the operational semantics of LILY is of interest for the following reasons:

- It shows that *our definition of LAPL in [8] is* indeed a *general* one: In [8] we presented a model based on admissible partial equivalence relations over a universal model of the untyped lambda calculus and in [15] we presented a model based on synthetic domain theory.
- The present model is *simpler* than our previous models mentioned above in that it does not require any domain theory, realizability, or synthetic domain theory. We hope this will make it more accessible. To make the model accessible also for readers who are not that familiar with category theory we deliberately choose to show not only the necessary categorical properties required of an LAPL-structure but also sketch in more explicit terms the interpretation of LAPL in our new model.
- The previous models mentioned above were constructed via a so-called parametric completion process, which roughly means that they were constructed in two steps: first a simple non-parametric model was constructed and then it was made parametric by filtering out all the non-parametric elements. (See [12] for more on parametric completions.) The present model is the first such[1] that is *not* based on a parametric completion process.
- It allows us to conclude that a wide range of types are definable up to ground contextual equivalence in LILY, as claimed but not formally proved in [3] (except for the simple case of finite coproducts). Hence our model can be used to prove correct program transformations based on parametricity for a language with general recursive types, an improvement over earlier work [10], which only dealt with algebraic data types.
  In fact, we also proved this LILY definability of types in [15] from the model based on synthetic domain theory by combining it with an adequate denotational semantics of LILY, but here we can do it by much simpler techniques.
- The model we present here is, to our knowledge, the first model of general recursive types as formalized via algebraic compactness based on operational semantics. Algebraic compactness is a categorical formulation of what it means to solve recursive domain equations; it ensures that the solutions are universal in an appropriate categorical sense, thus allowing for the derivation of (mixed) inductive-coinductive reasoning principles, c.f. [16]. In earlier work by the first author and Harper [4] a reasoning principle similar to the one derived here was presented for a recursive type, but it only worked for a single top-level recursive type. For other related work on operational models of recursive types, e.g., [1, 11], one may probably also establish useful reasoning principles for the recursive types but as far as we know it has not been done and, at any rate, it is pleasing that the reasoning principles presented here are an immediate consequence of general results about LAPL-structures.

[1] Except for the syntactic one constructed to prove completeness of LAPL-structures.

The remainder of this paper is organized as follows.

In Section 2 we recall the notion of an LAPL structure. In Section 3 we extend the LILY language with tensor types and associated terms. Moreover, we extend the operational properties of LILY studied in [3] to the new language and slightly generalize the results in [3] (to work for terms with free term and type variables, e.g.) and include a few new ones needed for the model construction. We have included tensor types, even though we in the end show that they are definable using parametricity, because it eases the construction of the model.

In Section 4 we construct a PILL_Y-model from the operational semantics of our extension of LILY. This is the first step in the construction of the LAPL-structure. The next step in the construction is the logic fibration. The interpretation of the logic is basically set theoretic, interpreting propositions on types, for example, as subsets of sets of ground contextual equivalence classes of terms. As mentioned, the $\top\top$-closed relations play the role of admissible relations in LAPL, and thus need to satisfy certain closure properties. These are also established in Section 4.

The last step in the construction of the LAPL-structure is the relational interpretation of types. We show how the inductive definition of the interpretation of LILY-types as $\bot\bot$-closed relations presented in [3] defines a relational interpretation of types satisfying the requirements for LAPL-structures. Finally, it is shown that the parametricity schema does indeed hold in the constructed LAPL-structure.

In Section 5 we describe the interpretation of PILL_Y into the model we have constructed out of the operational semantics of LILY.

Finally, in Section 6 we derive some consequences of parametricity for LILY, and we conclude in Section 7.

2 LAPL-structures

The equational theory PILL_Y is a variant of DILL [2] extended with polymorphism and fixed points given by a fixed point combinator of type $\prod \alpha.(\alpha \to \alpha) \to \alpha$, where in general we use $\sigma \to \tau$ as notation for $!\sigma \multimap \tau$.

We start off by sketching the notion of LAPL-structure as described in [8]. Some readers may prefer to first look at the concrete LAPL-structure constructed in Section 4 and the explicit description of the interpretation in Section 5 and then refer back to this section later. LAPL-structures model a variant of Abadi & Plotkin's logic for parametricity [20, 19] designed for reasoning about PILL_Y. Propositions in the logic exist in contexts of free type variables, free variables of PILL_Y and free relational variables. The free relational variables may be relations or admissible relations. Propositions are written as

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

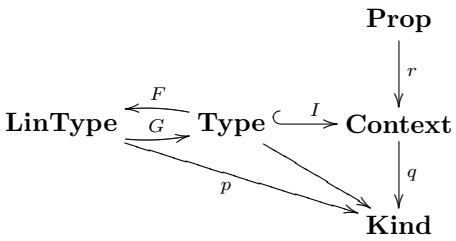
The vector $\boldsymbol{\alpha}$ is a list of free type variables. We will not describe the logic in details, but only mention a few main points. The variables $\boldsymbol{x} : \boldsymbol{\sigma}$ are treated intuitionistically in the logic. We may reason about linear terms by for example

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

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

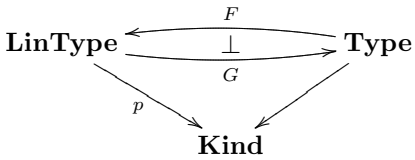
For any type $\boldsymbol{\alpha} \vdash \sigma(\boldsymbol{\alpha})$: **Type** with n free variables, and any n-vector of *admissible* relations $\boldsymbol{R}$: $\mathsf{AdmRel}(\boldsymbol{\tau}, \boldsymbol{\tau'})$, we can form an admissible relation $\sigma[\boldsymbol{R}] : \mathsf{AdmRel}(\sigma(\boldsymbol{\tau}), \sigma(\boldsymbol{\tau'}))$. This is called the relational interpretation of σ, and it is important for reasoning about parametricity. For example we can express the *identity extension schema* [21] as $\sigma[eq_{\boldsymbol{\alpha}}] \equiv eq_{\sigma(\boldsymbol{\alpha})}$, which we use as our definition of parametricity.

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



(1)

such that the diagram



(2)

is a model of PILL_Y [14] (a fibred version of a model of DILL [2], with generic object $\Omega \in$ **Kind** for p, simple products modeling polymorphism in p, and a term modeling the fixed point combinator).

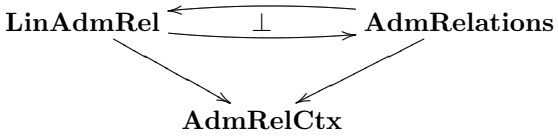
We further require that the fibration q has fibred products and that I is a faithful map preserving fibred products. The pair of fibrations (r, q) is an indexed first-order logic fibration which has products and coproducts with respect to projections $\Xi \times \Omega \to \Xi$ in **Kind** [5], meaning that each fibre of r over an object Ξ in **Kind** is a first-order logic fibration with structure preserved under reindexing, and that the logic models quantification along the mentioned projections in **Kind**.

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

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

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

From a pre-LAPL-structure with a notion of admissible relations one can define a PILL model (a PILL_Y model that does not necessarily model Y)



(3)

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

$$\partial_0 \circ J = \partial_1 \circ J = \mathsf{id}.$$

The functor J models the relational interpretation of types. It enables us to talk about parametricity at all types in the model, not just the interpretations of types in *pure* PILL_Y.

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

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

should hold and the model should have very strong equality. The latter means that if two terms of PILL_Y are provably equal in the logic, then they are in fact equal in the model.

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

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

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

preserving composition and identities. In the model σ induces an endo-functor $[\![\sigma]\!]$ on $\mathbf{LinType}_1$, the fibre over the terminal object 1, which is the model of the closed types.

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

$$\begin{gathered} \mathit{in} : \sigma(\mu\alpha.\sigma(\alpha)) \multimap \mu\alpha.\sigma(\alpha), \\ \mathit{fold} : \prod \beta.(\sigma(\beta) \multimap \beta) \to (\mu\alpha.\sigma(\alpha) \multimap \beta) \end{gathered}$$

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

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

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

$$\begin{gathered} \mathit{out} : \nu\alpha.\sigma(\alpha) \multimap \sigma(\nu\alpha.\sigma(\alpha)), \\ \mathit{unfold} : \prod \beta.(\beta \multimap \sigma(\beta)) \to (\nu\alpha.\sigma(\alpha) \multimap \beta) \end{gathered}$$

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

Theorem 1 ([8]). *Suppose $\alpha \vdash \sigma(\alpha)$ is a type in pure PILL_Y in which α occurs only positively. In any parametric LAPL-structure* in *is interpreted as an initial algebra and* out *as a final coalgebra for $[\![\sigma]\!] : \mathbf{LinType}_1 \to \mathbf{LinType}_1$.*

The next proposition provides solutions to recursive domain equations.

Theorem 2 ([8]). *Suppose $\alpha \vdash \sigma(\alpha) : \mathbf{Type}$ is a type in pure PILL_Y (α may appear both positively and negatively). There exists a closed type $\mathit{rec}\ \alpha.\sigma(\alpha)$ in PILL_Y and terms*

$$\begin{gathered} f : \mathit{rec}\ \alpha.\sigma(\alpha) \multimap \sigma(\mathit{rec}\ \alpha.\sigma(\alpha)), \\ g : \sigma(\mathit{rec}\ \alpha.\sigma(\alpha)) \multimap \mathit{rec}\ \alpha.\sigma(\alpha) \end{gathered}$$

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

We refer to [8, 15] for further details, in particular for a formulation of the above theorem for recursive types with parameters (as needed for nested recursive types), which we omitted here for simplicity.

3 The LILY Language

We have extended the work in [3] in two main ways: The language has been extended with tensor and all the definitions and theorems have been extended to cover terms and types with free term and type variables.

The extension of the language was rather straightforward while the extension to open terms and types required incorporation and adaptation of techniques described in [17]. Among other things, it was necessary to define a frame stack semantics and show a $=_{\mathrm{ciu}}$-theorem.

For reasons of space we omit most of the details of this development; they can be found in the accompanying technical report [9]. The main results of interest here are that contextual equivalence can be described by an inductively defined

relation Δ, that this relation models the equational theory of PILL_Y and that it satisfies a substitution property which turns out to correspond to the identity extension schema in our model.

The language without unit and tensor is given in [3]. In LILY with tensor the raw terms and types are defined by:

$$\begin{array}{rl} \tau & ::= \alpha \mid \tau \multimap \tau' \mid \forall\alpha.\tau \mid !\tau \mid I \mid \tau \otimes \tau \\ M & ::= a \mid x \mid \lambda a:\tau.M \mid M_1 M_2 \mid \Lambda\alpha.M \mid M\tau \mid \\ & \quad !(x = M : \tau) \mid \text{let } !x \text{ be } M_1 \text{ in } M_2 \mid * \mid \\ & \quad \text{let } * \text{ be } M_1 \text{ in } M_2 \mid M_1 \otimes M_2 \mid \\ & \quad \text{let } a_1 \otimes a_1 \text{ be } M_1 \text{ in } M_2 \end{array}$$

The typing judgments from [3] have been extended with standard rules for $*$ and $\otimes$. The full type system can be found in Appendix A.

As in [3], we have two dynamic semantics. One gives a call-by-value semantics ($\Downarrow_s$) and one gives a call-by-name semantics ($\Downarrow_n$).

Definition 1. *Let $\Downarrow$ denote either $\Downarrow_s$ or $\Downarrow_n$. We then define $\Downarrow_s$ and $\Downarrow_n$ as* evaluation *relations given by:*

$$v_1: \frac{}{\lambda a:\tau.M \Downarrow \lambda a:\tau.M} \qquad v_2: \frac{}{\Lambda\alpha.M \Downarrow \Lambda\alpha.M}$$

$$v_3: \frac{}{!(x = M : \tau) \Downarrow !(x = M : \tau)}$$

$$v_4: \frac{}{* \Downarrow *} \qquad v_5: \frac{}{M_1 \otimes M_2 \Downarrow M_1 \otimes M_2}$$

$$app_t: \frac{M \Downarrow \Lambda\alpha.M' \qquad M'[\tau/\alpha] \Downarrow V}{M\tau \Downarrow V}$$

$$rec: \frac{\begin{array}{c} M_1 \Downarrow !(x = M : \tau) \\ M_2[(\text{let } !x \text{ be } !(x = M : \tau) \text{ in } M)/y] \Downarrow V \end{array}}{\text{let } !y \text{ be } M_1 \text{ in } M_2 \Downarrow V}$$

$$c_{val}: \frac{M_1 \Downarrow_s \lambda a:\tau.M \qquad M_2 \Downarrow_s V' \qquad M[V'/a] \Downarrow_s V}{M_1 M_2 \Downarrow_s V}$$

$$c_{name}: \frac{M_1 \Downarrow_n \lambda a:\tau.M \qquad M[M_2/a] \Downarrow_n V}{M_1 M_2 \Downarrow_n V}$$

$$unit: \frac{M_1 \Downarrow * \qquad M_2 \Downarrow V}{\text{let } * \text{ be } M_1 \text{ in } M_2 \Downarrow V}$$

$$tensor: \frac{M_1 \Downarrow N_1 \otimes N_2 \qquad M_2[N_1, N_2/a_1, a_2] \Downarrow V}{\text{let } a_1 \otimes a_2 \text{ be } M_1 \text{ in } M_2 \Downarrow V}$$

Again, the operational semantics has been extended to accommodate $\otimes$. The tensor of two arbitrary terms constitute a value. We write $E \Downarrow_n$ to mean $\exists v.E \Downarrow_n v$ and let Typ denote the set of closed LILY types.

We will abbreviate let $!x$ be $!(x = M : \tau)$ in M to fix $x : \tau.M$ as the evaluation of this term is defined recursively and behaves like a fixed point. We will also abbreviate a non-recursive thunk $!(x = M : \tau)$, $x \notin fiv(M)$ to $!M$. These constructs then both look and behave like constructs of PILL_Y.

Definition 2 (Ground contextual equivalence). *Given terms M, M' such that $\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M : \tau$ and $\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M' : \tau$ we define $\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M =_{\mathrm{gnd}} M' : \tau$ if and only if for all closed types $\tau' \in \mathsf{Typ}$ and all contexts C such that $\emptyset; \emptyset \vdash_{\emptyset} C[M] :!\tau'$ and $\emptyset; \emptyset \vdash_{\emptyset} C[M'] :!\tau'$, we have $C[M] \Downarrow_n \Leftrightarrow C[M'] \Downarrow_n$.*

We now recall the notion of $\top\top$-closed relation [3].

Definition 3. *For all closed types $\tau \in \mathsf{Typ}$, let $Test(\tau)$ denote the set $\{\lambda a : \tau.M | \exists \tau' \in \mathsf{Typ}. \emptyset; \emptyset \vdash_{\emptyset} \lambda a : \tau.M : \tau \multimap !\tau'\}$. For all closed types $\tau, \tau' \in \mathsf{Typ}$, let $Rel(\tau, \tau') =_{\mathrm{def}} \{r \,|\, r \subseteq Term(\tau) \times Term(\tau')\}$, and let $Rel^*(\tau, \tau') =_{\mathrm{def}} \{s \,|\, s \subseteq Test(\tau) \times Test(\tau')\}$.*

Given $r \in Rel(\tau, \tau')$, let $r^\top \in Rel^*(\tau, \tau')$ be given by

$$r^\top =_{\mathrm{def}} \{(V, V') \mid \forall (M, M') \in r. VM \Downarrow_n \Leftrightarrow V'M' \Downarrow_n\}.$$

Given $s \in Rel^*(\tau, \tau')$, let $s^\top \in Rel(\tau, \tau')$ be given by

$$s^\top =_{\mathrm{def}} \{(M, M') \mid \forall (V, V') \in s. VM \Downarrow_n \Leftrightarrow V'M' \Downarrow_n\}.$$

Lemma 1. *The operations $(-)^\top$ forms a* Galois *connection between $Rel(\tau, \tau')$ and $Rel^*(\tau, \tau')$.*

As easy consequences hereof we get:

Lemma 2. *The operator $(-)^{\top\top}$ is* monotone $(-_1 \subseteq -_2 \Rightarrow (-_1)^{\top\top} \subseteq (-_2)^{\top\top})$, inflationary $((-) \subseteq (-)^{\top\top})$ *and* idempotent $\left((-)^{\top\top} = (-)^{\top\top\top\top}\right)$. *The operator $(-)^\top$ is* order reversing $\left(-_1 \subseteq -_2 \Rightarrow (-_1)^\top \supseteq (-_2)^\top\right)$.

Thus we see that $(-)^{\top\top}$ is a closure operation on relations, and we say that a relation r is $\top\top$-closed if $r = r^{\top\top}$.

We now define some operations on relations, which will be used to give a relational interpretation of the LILY types. The definitions come from [3], except for the case of $\otimes$, which, however, is as would be expected.

Given $r_1 \in Rel(\tau_1, \tau_1')$ and $r_2 \in Rel(\tau_2, \tau_2')$, let $r_1 \multimap r_2 \in Rel(\tau_1 \multimap \tau_2, \tau_1' \multimap \tau_2')$ be given by

$$\{(M, M') \mid \forall (M_1, M_1') \in r_1. (MM_1, M'M_1') \in r_2\}.$$

Given a family $(R(r) \in Rel(\tau[\sigma/\alpha], \tau'[\sigma'/\alpha']) \mid \sigma, \sigma' \in \mathsf{Typ}, r \in Rel(\sigma, \sigma'))$, we define $\forall r.R(r) \in Rel(\forall \alpha.\tau, \forall \alpha'.\tau')$ to be

$$\{(M, M') \mid \forall \sigma, \sigma' \in \mathsf{Typ}, r \in Rel(\sigma, \sigma'). (M\sigma, M'\sigma') \in R(r)\}.$$

Given $r \in Rel(\tau, \tau')$, let $!r \in Rel(!\tau, !\tau')$ be given by

$$\{(!(x = M : \tau), !(x' = M' : \tau)) \mid (\mathrm{fix}\ x : \tau.M, \mathrm{fix}\ x' : \tau'.M') \in r\}.$$

Given $r_1 \in Rel(\tau_1, \tau_1')$ and $r_2 \in Rel(\tau_2, \tau_2')$, let $r_1 \otimes r_2 \in Rel(\tau_1 \otimes \tau_2, \tau_1' \otimes \tau_2')$ be given by

$$\{(M_1 \otimes M_2), (M_1' \otimes M_2') \mid (M_1, M_1') \in r_1 \wedge (M_2, M_2') \in r_2\}$$

We now define relational interpretation Δ_τ of types τ.

Definition 4. *For all* LILY *types* τ *with free type variables* $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_n$*, for all* $\boldsymbol{\tau}, \boldsymbol{\tau'} \in \mathsf{Typ}$*,* $\boldsymbol{r}$*, such that* $r_i \in Rel(\tau_i, \tau_i')$*, let*

$$\Delta_\tau : Rel(\tau_1, \tau_1') \times \cdots \times Rel(\tau_n, \tau_n') \to Rel(\tau[\boldsymbol{\tau}/\boldsymbol{\alpha}], \tau[\boldsymbol{\tau'}/\boldsymbol{\alpha}])$$

be a function defined inductively on τ*, by*

$$\begin{aligned}
\Delta_{\alpha_i}(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} r_i \\
\Delta_{\tau_1 \multimap \tau_2}(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} \Delta_{\tau_1}(\boldsymbol{r}/\boldsymbol{\alpha}) \multimap \Delta_{\tau_2}(\boldsymbol{r}/\boldsymbol{\alpha}) \\
\Delta_{\forall \alpha.\tau}(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} \forall r. \Delta_\tau(r^{\top\top}/\alpha, \boldsymbol{r}/\boldsymbol{\alpha}) \\
\Delta_{!\tau}(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} (!\Delta_\tau(\boldsymbol{r}/\boldsymbol{\alpha}))^{\top\top} \\
\Delta_I(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} \{(*,*)\}^{\top\top} \\
\Delta_{\tau_1 \otimes \tau_2}(\boldsymbol{r}/\boldsymbol{\alpha}) &=_{\text{def}} (\Delta_{\tau_1}(\boldsymbol{r}/\boldsymbol{\alpha}) \otimes \Delta_{\tau_2}(\boldsymbol{r}/\boldsymbol{\alpha}))^{\top\top}.
\end{aligned}$$

For closed LILY *types* $\tau \in \mathsf{Typ}$ *we write* Δ_τ *instead of* $\Delta_\tau(\emptyset/\emptyset) \in Rel(\tau, \tau)$*.*

One can show that Δ_τ takes $\top\top$-closed relations to $\top\top$-closed relations.

The following proposition expresses that Δ satisfies a substitution property corresponding to identity extension, since by Proposition 2 below we see that Δ corresponds to ground contextual equivalence, that is, to our notion of equality.

Proposition 1. *For all types* $\tau, \boldsymbol{\tau'}$ *and type variables* $\boldsymbol{\alpha}, \boldsymbol{\alpha'}$ *such that* $ftv(\tau) \subseteq \{\boldsymbol{\alpha}, \boldsymbol{\alpha'}\}$ *and* $ftv(\boldsymbol{\tau'}) \subseteq \boldsymbol{\alpha}$ *we have:*

$$\Delta_{\tau[\boldsymbol{\tau'}/\boldsymbol{\alpha'}]}(\boldsymbol{r}/\boldsymbol{\alpha}) = \Delta_\tau(\boldsymbol{r}/\boldsymbol{\alpha}, \Delta_{\boldsymbol{\tau'}}(\boldsymbol{r}/\boldsymbol{\alpha})/\boldsymbol{\alpha'}).$$

Below we extend the Δ relation to terms with free variables. The definition basically says that two open terms are related if they are related when we close them by substituting related closed terms into them.

Definition 5 (Logical relation on open terms). *Suppose* $\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M : \tau$ *and* $\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M' : \tau$ *with free type variables* $\alpha_1, \ldots, \alpha_l$*, free intuitionistic variables* $x_1 : \tau_1, \ldots, x_m : \tau_m$ *and free linear variables* $a_1 : \tau_1', \ldots, a_n : \tau_n'$*. We write*

$$\Gamma; \Delta \vdash M \Delta M' : \tau \tag{4}$$

if and only if, given any $\boldsymbol{\sigma}$*,* $\boldsymbol{\sigma'}$*,* $\boldsymbol{r}$*, such that* $r_i \in Rel(\sigma_i, \sigma_i')$ *and each* r_i *is* $\top\top$*-closed, given any* $\boldsymbol{N}$*,* $\boldsymbol{N'}$ *such that* $(N_j, N_j') \in \Delta_{\tau_j}(\boldsymbol{r}/\boldsymbol{\alpha})$*, and given any* $\boldsymbol{M}$*,* $\boldsymbol{M'}$ *such that,* $(M_k, M_k') \in \Delta_{\tau_k'}(\boldsymbol{r}/\boldsymbol{\alpha})$*, we have*

$$(M[\boldsymbol{\sigma}/\boldsymbol{\alpha}, \boldsymbol{N}/\boldsymbol{x}, \boldsymbol{M}/\boldsymbol{a}], M'[\boldsymbol{\sigma'}/\boldsymbol{\alpha}, \boldsymbol{N'}/\boldsymbol{x}, \boldsymbol{M'}/\boldsymbol{a}]) \in \Delta_\tau(\boldsymbol{r}/\boldsymbol{\alpha}). \tag{5}$$

Note that the Δ *to the right denotes the linear context whereas the* Δ *to the left is the relation that we are defining.*

Proposition 2. *We have*

$$\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M =_{\text{gnd}} M' : \tau \Leftrightarrow \Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M \Delta M' : \tau$$

Using the above proposition and alternative characterizations of ground contextual equivalence one can show [9] that all the rules of the external equational theory of PILL_Y [8] hold when equality is interpreted as contextual equivalence:

Lemma 3.

$$\begin{aligned}
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} (\lambda a:\tau.M)N =_{\mathrm{gnd}} M[N/a] : \tau' \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} (\Lambda\alpha.M)\sigma =_{\mathrm{gnd}} M[\sigma/\alpha] : \tau[\sigma/\alpha] \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} \text{let } !y \text{ be } !(x = N : \tau) \text{ in } M \\
&\qquad =_{\mathrm{gnd}} M[(\text{fix } x:\tau.N)/y] : \tau' \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} \text{let } * \text{ be } * \text{ in } M =_{\mathrm{gnd}} M : \tau \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} \text{let } a_1 \otimes a_2 \text{ be } M_1 \otimes M_2 \text{ in } M_3 \\
&\qquad =_{\mathrm{gnd}} M_3[M_1, M_2/a_1, a_2] : \tau \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} \text{fix } x:\alpha.M =_{\mathrm{gnd}} M \qquad x \textit{ not free in } M \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} Y\sigma(!M) =_{\mathrm{gnd}} M!(Y\sigma(!M)) : \sigma \\
&\Gamma;\Delta \vdash_{\boldsymbol{\alpha}} \text{let } !y \text{ be } !(x = M_1 : \tau) \text{ in } M_2 \\
&\qquad =_{\mathrm{gnd}} M_2[M_1/y] : \tau' \qquad x \textit{ not free in } M_1
\end{aligned}$$

4 The LILY LAPL-structure

In this section we show how to construct an LAPL-structure from the operational semantics of LILY. Thus we define specific categories **LinType**, **Type**, functors, *etc.*, and prove that they constitute an LAPL-structure.

We write $\overline{\alpha}^n$ for $\alpha_1, \ldots, \alpha_n$ (all distinct) and let $\mathsf{Terms}(\sigma)$ denote the set of ground contextual equivalence classes of closed LILY terms of type σ.

The PILL_Y*-model* The base category **Kind** is defined as follows.

Definition 6 (Kind).

Objects: $\overline{\alpha}^n$
Morphisms: $\overline{\sigma}^m : \overline{\alpha}^n \to \overline{\alpha}^m$ *iff* $\forall i \in \{1, \ldots, m\}.\ \overline{\alpha}^n \vdash \sigma_i$

Note, that we allow $\overline{\alpha}^0$ as an object of **Kind**. The category **Kind** is clearly cartesian.

The category **LinType** is defined as follows.

Definition 7 (LinType).

Objects: $(\overline{\alpha}^n, \sigma)$ *such that* $\overline{\alpha}^n \vdash \sigma$
Morphisms: $(\overline{\omega}^m, [M]) : (\overline{\alpha}^n, \sigma) \to (\overline{\alpha}^m, \tau)$, *if* $[M]$ *is a ground contextual equivalence class of LILY-terms, such that*
- $\overline{\omega}^m : \overline{\alpha}^n \to \overline{\alpha}^m$ *in* **Kind**
- $-; x:\sigma \vdash_{\overline{\alpha}^n} M : \tau[\overline{\omega}^m/\overline{\alpha}^m]$

Note that an object $(\overline{\alpha}^n,\sigma)$ of **LinType** is a morphism $\overline{\alpha}^n \to \overline{\alpha}^1$ of **Kind**.

There is an obvious projection functor $p :$ **LinType** $\to$ **Kind** given by $(\overline{\alpha}^n, \sigma) \mapsto \overline{\alpha}^n$ and $(\overline{\omega}^m, [M]) \mapsto \overline{\omega}^m$.

Note that the fibre $\mathbf{LinType}_n$ over n has objects σ the types $\overline{\alpha}^n \vdash \sigma$ with n free type variables and morphisms $[M] : \sigma \to \tau$ are equivalence classes of terms $-; x : \sigma \vdash_{\overline{\alpha}^n} M : \tau$. Composition is by substitution.

LinType $\to$ **Kind** is fibred symmetric monoidal closed, as required and expected. We write out some of the SMCC structure for the fibre $\mathbf{LinType}_n$. The tensor is given by:

$$\begin{aligned} I &= I \\ \sigma \otimes \tau &= \sigma \otimes \tau \\ [M] \otimes [N] &= [\text{let } x \otimes y \text{ be } z \text{ in } M \otimes N] \end{aligned}$$

where $-; x : \sigma \vdash_{\overline{\alpha}^n} M : \sigma'$, $-; y : \tau \vdash_{\overline{\alpha}^n} N : \tau'$ and $-; z : \sigma \otimes \tau \vdash_{\overline{\alpha}^n} \text{let } x \otimes y \text{ be } z \text{ in } M \otimes N : \sigma' \otimes \tau'$.

The closed structure, $\sigma \multimap (-)$, is given by

$$\begin{aligned} \sigma \multimap \tau &= \sigma \multimap \tau \\ \sigma \multimap [M] &= [\lambda x : \sigma.(M[(fx)/y])] \end{aligned}$$

where $-; y : \tau \vdash_{\overline{\alpha}^n} M : \tau'$ and $-; f : \sigma \multimap \tau \vdash_{\overline{\alpha}^n} \lambda x : \sigma.(M[(fx)/y]) : \sigma \multimap \tau'$.

The category **Type** is defined as follows.

Definition 8 (Type).

Objects: $(\overline{\alpha}^n, \overline{\sigma}^s)$ *such that* $\forall i \in \{1, \ldots, s\}.\ \overline{\alpha}^n \vdash \sigma_i$
Morphisms: $(\overline{\omega}^m, \overline{[M]}^r) : (\overline{\alpha}^n, \overline{\sigma}^s) \to (\overline{\alpha}^m, \overline{\tau}^r)$ *such that*
- $\overline{\omega}^m : \overline{\alpha}^n \to \overline{\alpha}^m$ *in* **Kind**
- $\forall i \in \{1, \ldots, r\}.\ \overline{x : \sigma}^s; - \vdash_{\overline{\alpha}^n} M_i : \tau_i[\overline{\omega}^m/\overline{\alpha}^m]$.

Note that objects here are sequences of types—this ensures that the fibration $q : \mathbf{Type} \to \mathbf{Kind}$ given by the obvious projection functor has fibred finite products.

The fibre $\mathbf{Type}_n$ over n has as objects sequences of types $\overline{\sigma}^s$ such that $\forall i \in \{1, \ldots, s\}.\ \overline{\alpha}^n \vdash \sigma_i$ and morphisms $\overline{[M]}^r : \overline{\sigma}^s \to \overline{\tau}^r$ are sequences of terms such that $\forall i \in \{1, \ldots, r\}.\ \overline{x : \sigma}^s; - \vdash_{\overline{\alpha}^n} M_i : \tau_i$.

The product in $\mathbf{Type}_n$ is given by concatenation, projections by intuitionistic weakening, while the exponential $\overline{\sigma}^s \to \overline{\tau}^r$ is $\overline{!\sigma}^s \multimap \tau}^r$, a simple adaptation of the usual equation, $\sigma \to \tau = !\sigma \multimap \tau$, to our case with lists of types.

We now define the adjunction $F \dashv G$. It is mostly given by the fact that **Type** is almost the coKleisli category of the !-monad on **LinType**. Thus G is almost a forgetful functor and F is almost !.

The functor $G :$ **LinType** $\to$ **Type** is defined on objects by $(\overline{\alpha}^n, \sigma) \mapsto (\overline{\alpha}^n, \sigma)$ and on morphisms by $(\overline{\omega}^m, [M]) \mapsto (\overline{\omega}^m, [M])$. Note that G is not the identity on terms, as $-; x : \sigma \vdash_{\overline{\alpha}^n} M : \tau[\overline{\omega}^m/\overline{\alpha}^m]$ is mapped to $x : \sigma; - \vdash_{\overline{\alpha}^n} M : \tau[\overline{\omega}^m/\overline{\alpha}^m]$.

The functor $F :$ **Type** $\to$ **LinType** is defined on objects by $(\overline{\alpha}^n, \overline{\sigma}^s) \mapsto (\overline{\alpha}^n, !\sigma_1 \otimes \cdots \otimes !\sigma_s)$ and on morphisms by

$$(\overline{\omega}^m, \overline{[M]}^r) \mapsto (\overline{\omega}^m, [\text{let } \otimes_i x_i' : \otimes_i !\sigma_i \text{ be } y \text{ in let } \overline{!x}^m \text{ be } \overline{x'}^m \text{ in } \otimes_i !M_i]).$$

Note that $(F \circ G)(\sigma) = !\sigma$.

If we define $\mathbf{!} = F \circ G$ and calculate the resulting functor, we get

$$\begin{aligned} \mathbf{!}(\sigma) &= !\sigma \\ \mathbf{!}([M]) &= [\text{let } !y \text{ be } x \text{ in } !M]. \end{aligned}$$

This defines a comonad on **LinType** with the following structure:

$$\delta_\sigma : !\sigma \to !!\sigma = [\text{let } !y \text{ be } x \text{ in } !!y] \qquad \epsilon_\sigma : !\sigma \to \sigma = [\text{let } !y \text{ be } x \text{ in } y].$$

Lemma 4. *The fibrations p and q together with the adjunction $F \dashv G$ constitute a PILL_Y-model.*

Pre-LAPL-structure The pre-LAPL-structure will be given by

$$\begin{array}{ccccc} & & & & \mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \\ & & & & \downarrow \\ \mathbf{LinType} & \underset{G}{\overset{F}{\rightleftarrows}} & \mathbf{Type} & \xrightarrow{\quad I \quad} & \mathbf{Fam}(\mathbf{Set}) \\ & {\scriptstyle p}\searrow & \downarrow{\scriptstyle q} & \swarrow & \\ & & \mathbf{Kind} & & \end{array} \tag{6}$$

Here the fibrations $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Kind}$ are the usual fibrations $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Set}$ reindexed along the functor $S : \mathbf{Kind} \to \mathbf{Set}$ given on objects by $\overline{\alpha}^n \mapsto \mathsf{Typ}^n$ and on morphisms by $\overline{\sigma}^m \mapsto (\overline{\tau}^n \mapsto \overline{\sigma[\overline{\tau}^n/\overline{\alpha}^n]}^m)$. This functor S tells us how to think of the objects of **Kind** in terms of sets: n free type variables are thought of as all choices of n closed types. The morphism $\overline{\sigma}^m$ is thought of as the corresponding transformation of choices. Explicitly, the fibre $\mathbf{Fam}(\mathbf{Set})_n$ has objects $(\overline{\alpha}^n, (A_i)_{i \in S(\overline{\alpha}^n)})$, where each A_i is a set, and morphism $(f_i)_{i \in S(\overline{\alpha}^n)} : (\overline{\alpha}^n, (A_i)_{i \in S(\overline{\alpha}^n)}) \to (\overline{\alpha}^n, (B_j)_{j \in S(\overline{\alpha}^n)})$ are functions $f_i : A_i \to B_{S(\overline{\sigma}^m)(i)}$.

The functor I is given by "product of sets of terms": I maps objects $(\overline{\alpha}^n, \overline{\sigma}^s)$ to $(\overline{\alpha}^n, (\mathsf{Terms}(\sigma_1[\overline{\tau}^n/\overline{\alpha}^n]) \times \cdots \times \mathsf{Terms}(\sigma_s[\overline{\tau}^n/\overline{\alpha}^n]))_{\overline{\tau}^n \in S(\overline{\alpha}^n)})$ and morphisms $(\overline{\omega}^m, \overline{[M]}^r)$ to

$$(\overline{\omega}^m, (\overline{(t : \sigma_1[\overline{\tau}^n/\overline{\alpha}^n]}^s \mapsto \overline{[M[\overline{\tau}^n/\overline{\alpha}^n][\overline{t}^s/\overline{x}^s]]}^r)_{\overline{\tau}^n \in S(\overline{\alpha}^n)}).$$

I is product-preserving by definition and that it is faithful follows from LILY extensionality (Section 3).

Lemma 5. *The composite fibration* $\mathbf{Fam}(\mathrm{Sub}(\mathbf{Set})) \to \mathbf{Fam}(\mathbf{Set}) \to \mathbf{Kind}$ *is a fibred first-order logic fibration with products with respect to projections in* **Kind**.

Proof. This follows from general results in [5] and is obvious since the logic is just the standard logic of sets. □

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

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

$$U : \mathbf{LinType} \times_{\mathbf{Kind}} \mathbf{LinType} \to \mathbf{Context} = 2^{I(-)\times I(=)}.$$

□

We write $r \subseteq \sigma \times \tau$ to denote the fact that $r \subseteq \mathsf{Terms}(\sigma) \times \mathsf{Terms}(\tau)$ while $r \subseteq_{Adm} \sigma \times \tau$ denotes the fact that $r \subseteq \sigma \times \tau$ and $r = r^{\top\top}$.

Lemma 7. *The subfunctor of U given by mapping $(\overline{\alpha}^n, \sigma, \tau)$ to $(\overline{\alpha}^n, (\{r \subseteq_{Adm} S(\sigma)(\overline{\tau}^n) \times S(\tau)(\overline{\tau}^n)\})_{\overline{\tau}^n \in S(\overline{\alpha}^n)})$ defines a notion of admissible relations for the pre-LAPL-structure (6).*

Proof. See Appendix B. □

As mentioned in the introduction, one of our aims with this paper is to show that the notion of parametric LAPL-structures is a general notion of parametric model. Lemma 7 is important in this respect, since it shows that the concrete notion of $\top\top$-closed relations of the LILY model interprets the abstract notion of admissible relations presented in the definition of parametric LAPL-structures.

Theorem 3. *The pre-LAPL-structure (6) is an LAPL-structure.*

Basically, what needs to be proved in Theorem 3 is that all types in the model have a relational interpretation. But that is, of course, provided by the Δ from Section 3. Thus the required J functor is given (explicit descriptions of the abstractly defined categories **AdmRelCtx** and **AdmRelations** can be found in appendix) by

Definition 9 ($J_{Base} :$ **Kind** $\to$ **AdmRelCtx**).

Objects: $\overline{\alpha}^n \mapsto (\overline{\alpha}^n, \overline{\alpha}^n, (\Pi_{i=1}^n \{r \subseteq_{Adm} \tau_i \times \tau_{i+n}\})_{\overline{\tau}^{2n} \in \mathsf{Typ}^{2n}})$
Morphisms: $\overline{\sigma}^m \mapsto (\overline{\sigma}^m, \overline{\sigma}^m,$
$((r_1 \subseteq_{Adm} \tau_1 \times \tau_{n+1}, \ldots, r_n \subseteq_{Adm} \tau_n \times \tau_{2n}) \mapsto$
$(\Delta_{\sigma_1}(\overline{r}^n/\overline{\alpha}^n), \ldots, \Delta_{\sigma_m}(\overline{r}^n/\overline{\alpha}^n)))_{\overline{\tau}^{2n} \in \mathsf{Typ}^{2n}})$

Definition 10 ($J_{Total} :$ **LinType** $\to$ **AdmRelations**).

Objects: $(\overline{\alpha}^n, \sigma) \mapsto (J_{Base}(\overline{\alpha}^n), J_{Base}(\sigma))$
Morphisms: $(\overline{\omega}^m, [M]) \mapsto (J_{Base}(\overline{\omega}^m), [M], [M])$

Since the relational interpretation of types in LAPL-structures is given by a map of PILL-models, the proof of Theorem 3 also checks that the definitions of Δ for the various type constructors $(\multimap, \otimes, !)$ agrees with the abstractly defined structure on **LinAdmRel** $\to$ **AdmRelCtx**. We include the proof for $\otimes$ in appendix.

Theorem 4. *The LAPL-structure (6) is a parametric LAPL-structure, i.e., satisfies identity extension, extensionality and very strong equality.*

Proof. For identity extension, note that since contextual equivalence coincides with the relation Δ, we can rewrite the required equation to

$$\Delta_{\sigma[\tau_1/\alpha_1,\ldots,\tau_n/\alpha_n]} = J(\sigma)(\Delta_{\tau_1},\ldots,\Delta_{\tau_n}) = \Delta_\sigma[\Delta_{\tau_1}/\alpha_1,\ldots,\Delta_{\tau_n}/\alpha_n],$$

which is the content of Proposition 1. Very strong equality and extensionality follows from the same properties of the subobject fibration over **Set**. □

5 Interpretation

In this section we describe the interpretation of PILL_Y into the model we have constructed out of the operational semantics of LILY.[2]

A PILL_Y kind context $\overline{\alpha}^n$ is interpreted as the object $\overline{\alpha}^n$ in **Kind**.

A PILL_Y type $\overline{\alpha}^n \vdash \sigma$ with n free type variables is modeled as an object in **LinType** in the fiber over $\overline{\alpha}^n$:

$$[\![\overline{\alpha}^n \vdash \sigma]\!] = (\overline{\alpha}^n, \sigma),$$

which we abbreviate as:

$$[\![\sigma]\!] = \sigma.$$

These definitions look deceptively simple. To verify that this is indeed the interpretation one obtains in the LILY LAPL-structure one must of course calculate the interpretation of types in the LILY LAPL-structure, and then one quickly sees that the interpretation is as shown above.

Type contexts are modeled by the tensor and comonad structure on **LinType**. A context $x_1 : \sigma_1, \ldots, x'_n : \sigma_n; a_1 : \tau_1, \ldots, a_m : \tau_m$ is modeled as

$$!{[\![\sigma_1]\!]} \otimes \ldots \otimes !{[\![\sigma_n]\!]} \otimes [\![\tau_1]\!] \otimes \ldots \otimes [\![\tau_m]\!]$$

Thus we use tensor to concatenate contexts and we use **!** to make the types in the intuitionistic context behave intuitionistically. Of course the entire context will be inside one fiber of **LinType**.

Terms with n free type variables are modeled as morphisms in **LinType** in the fiber over $\overline{\alpha}^n$.

A term $\Xi \mid \Gamma; \Delta \vdash t : \sigma$ is modeled as a morphism

$$[\![t]\!] : [\![\Gamma; \Delta]\!] \to [\![\sigma]\!]$$

[2] Note that by the results in the preceding section, we already know that we have a well-defined interpretation of PILL_Y — that is a direct consequence of the fact that we have constructed an LAPL structure. Here we merely try to provide an intuitive description of the resulting interpretation.

in $\mathbf{LinType}_n$, that is, as a ground contextual equivalence class of LILY terms, containing a representative M of the form

$$-; x : (!\llbracket\sigma_1\rrbracket \otimes \ldots \otimes !\llbracket\sigma_n\rrbracket \otimes \llbracket\tau_1\rrbracket \otimes \ldots \otimes \llbracket\tau_m\rrbracket) \vdash_\Xi M : \sigma.$$

The inductive description of $\llbracket t\rrbracket$ follows from [12, 2]. For example, $\llbracket \Xi \mid \Gamma; \Delta, \Delta' \vdash t \otimes s : \sigma \otimes \tau\rrbracket$ is the equivalence class of the LILY term

$$-; x :!\Gamma \otimes \Delta \vdash \text{ let } \gamma \otimes \delta \otimes \delta' = \text{split } x \text{ in } \llbracket t\rrbracket(\gamma \otimes \delta) \otimes \llbracket s\rrbracket(\gamma \otimes \delta'),$$

where split is a term definable in LILY such that split x is a tensor product in the obvious way.

Now consider the interpretation of formulas in context:

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

The interpretation of the above formula, abbreviated as simply $\llbracket\phi\rrbracket$, is a family of subsets, indexed by closed types $\boldsymbol{\tau}$:

$$\begin{aligned}\llbracket\phi\rrbracket \subseteq \quad & \textstyle\prod_{\sigma\in\boldsymbol{\sigma}} \mathsf{Terms}(\sigma[\boldsymbol{\tau}/\boldsymbol{\alpha}]) \\ & \times \textstyle\prod_{\sigma'\in\boldsymbol{\sigma}',\sigma''\in\boldsymbol{\sigma}''} P(\mathsf{Terms}(\sigma'[\boldsymbol{\tau}/\boldsymbol{\alpha}]) \times \mathsf{Terms}(\sigma''[\boldsymbol{\tau}/\boldsymbol{\alpha}])) \\ & \times \textstyle\prod_{\tau'\in\boldsymbol{\tau}',\tau''\in\boldsymbol{\tau}''} P^{\top\top}(\mathsf{Terms}(\tau'[\boldsymbol{\tau}/\boldsymbol{\alpha}]) \times \mathsf{Terms}(\tau''[\boldsymbol{\tau}/\boldsymbol{\alpha}])),\end{aligned}$$

where $P^{\top\top}$ denotes the function that yields the set of all $\top\top$-closed subsets. In plain words, $\llbracket\phi\rrbracket$ is a subset of ground contextual equivalence classes of terms of types σ, of relations (on ground contextual equivalence classes of terms) and of $\top\top$-closed relations (on ground contextual equivalence classes of terms). Thus it is a very natural interpretation.

The connectives and quantifiers are interpreted just as we ordinarily do in sets. For example, the interpretation of conjunction $\llbracket\phi \wedge \phi'\rrbracket$ is the intersection of $\llbracket\phi\rrbracket$ and $\llbracket\phi'\rrbracket$.

The only remaining point to note is the interpretation of substitution of terms into formulas: the interpretation of $\llbracket\phi[M/x]\rrbracket$, is obtained as follows. As explained above, the term M is interpreted as an equivalence class of LILY terms $[M]$ and hence it induces an obvious function, which works by substitution, between sets of ground contextual equivalence classes of terms $\mathsf{Terms}(\ldots) \to \mathsf{Terms}(\ldots)$ (formally, the function is obtained via the I functor and the adjunction $F \dashv G$). That function is used to reindex the interpretation of ϕ.

6 Consequences

We now consider a few consequences of Theorem 4.

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

As always, type expressions $\alpha \vdash \sigma(\alpha)$ in LILY for which α only appears positively in σ induce endofunctors on **Lily**.

Theorem 5. *All functors* **Lily** $\to$ **Lily** *induced by types* $\sigma(\alpha)$ *in* LILY *have initial algebras and final coalgebras and these are isomorphic.*

Proof. This is a simple corollary of Theorems 1 and 4 once we observe that $\mathbf{LinType}_1$ is equivalent to **Lily** and that types in PILL_Y are simply interpreted as the corresponding types in LILY. □

Likewise we have:

Theorem 6. *For all types* $\alpha \vdash \sigma(\alpha) : \mathbf{Type}$ *of* LILY *(where* α *may appear both positively and negatively), there exists a closed type* τ *of* LILY *such that* τ *and* $\sigma(\tau)$ *are isomorphic as objects of* **Lily**.

This way we get formal proofs of all the claimed isomorphisms in [3, Figure 1] (*loc. cit.* only includes a formal proof of definability of coproducts).[3] Moreover, it shows that our model can be used to prove correct program transformations based on parametricity for a language with general recursive types, an improvement over earlier work [10], which only dealt with algebraic data types.

Example 1. As an immediate corollary of the definability of $\otimes$ types in PILL_Y [20, 7], and the observation that $\mathbf{LinType}_1$ is equivalent to **Lily**, we get that in **Lily**, the object $\sigma \otimes \tau$ is isomorphic to $\prod \alpha.(\sigma \multimap \tau \multimap \alpha) \multimap \alpha$. Phrased in purely operational semantics terms, it means that in LILY there are terms f and g, with types $f : \sigma \otimes \tau \to \prod \alpha.(\sigma \multimap \tau \multimap \alpha) \multimap \alpha$ and $g : \prod \alpha.(\sigma \multimap \tau \multimap \alpha) \multimap \alpha \to \sigma \otimes \tau$ such that the LILY terms corresponding to the composition of f and g are ground contextually equivalent to the identity terms.

For general parametric LAPL-structures we may derive *reasoning principles* for the definable inductive, coinductive, and recursive types [9]. The principles look similar to the ones considered by Pitts [16] for classical domain theory. For instance, we have the following principle for recursive types (cf. Theorem 2):

Theorem 7. *Consider a type* $\sigma(\beta, \gamma)$ *where* β *occurs only negatively and* γ *occurs only positively. Suppose* $S_+ \subseteq_{Adm} \mathit{rec}\ \alpha.\sigma(\alpha,\alpha) \times \mathit{rec}\ \alpha.\sigma(\alpha,\alpha)$ *is an admissible relation and that* $S_- \subseteq \mathit{rec}\ \alpha.\sigma(\alpha,\alpha) \times \mathit{rec}\ \alpha.\sigma(\alpha,\alpha)$ *is a relation. Then the following rule holds:*

$$\frac{(f,f) : S_- \multimap \sigma(S_+, S_-) \qquad (g,g) : \sigma(S_-, S_+) \multimap S_+}{S_- \subset \mathit{rec}\ \alpha.\sigma(\alpha,\alpha) \subset S_+}$$

For the LILY LAPL-structure this theorem provides us with a mixed induction-/coinduction principles for proving contextual equivalence of elements of recursive types defined via parametric polymorphism in LILY.

Similar results were proved in operational semantics for a language with one top-level recursive type in [4]. However, we want to stress that the definability of recursive types in LAPL, Theorems 1 and 2, also works for recursive types with parameters [8], as does the reasoning principles for the resulting types.

[3] See [12] for the details of all the relevant proofs in LAPL.

Thus, without involving any form of classical or synthetic domain theory, but just relying on purely operational semantics and general properties of LAPL-structures, we have derived general reasoning principles for recursive types. We are not aware of any other work where such general principles are derived directly from operational semantics.[4]

7 Conclusion and Future Directions

We have constructed an LAPL-structure from the operational semantics of LILY and used it to establish formally definability of a wide range of types in LILY. Moreover, we have derived reasoning principles for definable inductive, coinductive, and recursive types.

In recent work, Møgelberg has investigated the application of LAPL in denotational semantics [13]. In particular, he has shown how one may use any LAPL-structure to define a denotational semantics of FPC. By studying the resulting semantics of FPC in the LILY LAPL structure constructed here, we conjecture that one may extend the semantics to a polymorphic version of FPC and show it adequate with respect to the operational semantics of polymorphic FPC. In operational terms the semantics amounts to a translation of polymorphic FPC into LILY.

Future work also includes studying the interaction of parametric polymorphism with other effects than non-termination, in particular with references.

Acknowledgments

We thank Andy Pitts and Alex Simpson for inspiring discussions.

References

1. A. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *Transactions on Programming Languages and Systems*, 23(5):657–683, 2001.
2. A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, Edinburgh University, 1997.
3. 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.
4. L. Birkedal and R. Harper. Constructing interpretations of recursive types in an operational setting. *Information and Computation*, 155:3–63, 1999.

[4] Note that the use of an LAPL-structure here does not replace conventional use of domain theory in that we do not give a denotational semantics of LILY in a LAPL-structure (and thus do not require an adequacy proof), but rather construct a concrete LAPL-structure directly from the operational semantics of LILY.

5. L. Birkedal and R. Møgelberg. Categorical models for Abadi-Plotkin's logic for parametricity. *Mathematical Structures in Computer Science*, 15:709–772, 2005.
6. L. Birkedal, R. E. Møgelberg, and R. L. Petersen. Linear Abadi & Plotkin logic. 2006. Submitted for publication.
7. L. Birkedal, R.E. Møgelberg, and R.L. Petersen. Parametric domain-theoretic models of linear Abadi & Plotkin logic. Technical Report TR-2005-57, IT University of Copenhagen, February 2005.
8. L. Birkedal, R.E. Møgelberg, and R.L. Petersen. Parametric domain-theoretic models of polymorphic intuitionistic / linear lambda calculus. In M. Escardó, A. Jung, and M. Mislove, editors, *Proceedings of Mathematical Foundations of Programming Semantics 2005*, volume 155, pages 191–217, 2005.
9. L. Birkedal, R.L. Petersen, R.E. Møgelberg, and C. Varming. Lily operational semantics and models of linear Abadi-Plotkin logic. Technical Report TR-2006-83, IT University of Copenhagen, 2006. Available at `www.itu.dk/people/birkedal/papers/lily-lapl-tr.pdf`.
10. P. Johann. On proving the correctness of program transformations based on free theorems for higher-order polymorphic calculi. *Mathematical Structures in Computer Science*, 15(2):201–229, 2005.
11. P. Mellies and J. Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *Proceedings of the 21st Conference on Logic in Computer Science*, 2005.
12. R. E. Møgelberg. *Category theoretic and domain theoretic models of parametric polymorphism*. PhD thesis, IT University of Copenhagen, 2005.
13. R. E. Møgelberg. Interpreting polymorphic FPC into domain theoretic models of parametric polymorphism. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2006. To appear.
14. R. E. Møgelberg, L. Birkedal, and R. L. Petersen. Categorical models of PILL. Technical Report TR-2005-58, IT University of Copenhagen, February 2005.
15. R.E. Møgelberg, L. Birkedal, and R. L. Petersen. Synthetic domain theory and models of linear Abadi and Plotkin logic. In *Proceedings of Mathematical Foundations of Programming Semantics 2005*, 2005.
16. A.M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.
17. A.M. Pitts. Operational semantics and program equivalence. In *Applied Semantics, International Summer School, APPSEM 2000, Caminha, Portugal, September 9-15, 2000, Advanced Lectures*, pages 378–412, London, UK, 2002. Springer-Verlag.
18. G. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Typed lambda calculi and applications (Utrecht, 1993)*, pages 361–375. Springer, Berlin, 1993.
19. G.D. Plotkin. Second order type theory and recursion. Notes for a talk at the Scott Fest, February 1993.
20. 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.
21. J.C. Reynolds. Types, abstraction, and parametric polymorphism. *Information Processing*, 83:513–523, 1983.

We include this appendix for the benefit of the reviewers.

A Type System for LILY

We say a raw term M is well-typed and has type τ if and only if there is a set of type variables $\boldsymbol{\alpha}$, an intuitionistic Γ and linear Δ type environment such that $\Gamma, \Delta, \boldsymbol{\alpha}, \tau, M$ are in the relation defined by the following rules.

$$\frac{ftv(\Gamma,\tau) \subseteq \boldsymbol{\alpha} \quad x \notin \mathrm{dom}(\Gamma)}{\Gamma, x:\tau; \emptyset \vdash_{\boldsymbol{\alpha}} x : \tau} \qquad \frac{ftv(\Gamma,\tau) \subseteq \boldsymbol{\alpha}}{\Gamma; a:\tau \vdash_{\boldsymbol{\alpha}} a : \tau}$$

$$\frac{\Gamma; \Delta, a:\tau \vdash_{\boldsymbol{\alpha}} M : \tau' \quad a \notin \mathrm{dom}(\Delta)}{\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} \lambda a:\tau.M : \tau \multimap \tau'}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\boldsymbol{\alpha}} M_1 : \tau \multimap \tau' \quad \Gamma; \Delta_2 \vdash_{\boldsymbol{\alpha}} M_2 : \tau \quad \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2) = \emptyset}{\Gamma; \Delta_1, \Delta_2 \vdash_{\boldsymbol{\alpha}} M_1 M_2 : \tau'}$$

$$\frac{\Gamma; \Delta \vdash_{\boldsymbol{\alpha},\alpha} M : \tau \quad \alpha \notin \boldsymbol{\alpha} \cup ftv(\Gamma,\Delta)}{\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} \Lambda\alpha.M : \forall\alpha.\tau} \qquad \frac{\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M : \forall\alpha.\tau \quad ftv(\tau') \subseteq \boldsymbol{\alpha}}{\Gamma; \Delta \vdash_{\boldsymbol{\alpha}} M\tau' : \tau[\tau'/\alpha]}$$

$$\frac{\Gamma, x:\tau; \emptyset \vdash_{\boldsymbol{\alpha}} M : \tau \quad x \notin \mathrm{dom}(\Gamma)}{\Gamma; \emptyset \vdash_{\boldsymbol{\alpha}} !(x = M : \tau) :! \tau}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\boldsymbol{\alpha}} M_1 :! \tau \quad \Gamma, x:\tau; \Delta_2 \vdash_{\boldsymbol{\alpha}} M_2 : \tau' \quad x \notin \mathrm{dom}(\Gamma) \quad \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2) = \emptyset}{\Gamma; \Delta_1, \Delta_2 \vdash_{\boldsymbol{\alpha}} \text{let } !x \text{ be } M_1 \text{ in } M_2 : \tau'}$$

$$\frac{}{\Gamma; \emptyset \vdash_{\boldsymbol{\alpha}} * : I}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\boldsymbol{\alpha}} M_1 : I \quad \Gamma; \Delta_2 \vdash_{\boldsymbol{\alpha}} M_2 : \tau \quad \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2) = \emptyset}{\Gamma; \Delta_1, \Delta_2 \vdash_{\boldsymbol{\alpha}} \text{let } * \text{ be } M_1 \text{ in } M_2 : \tau}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\boldsymbol{\alpha}} M_1 : \tau_1 \quad \Gamma; \Delta_2 \vdash_{\boldsymbol{\alpha}} M_2 : \tau_2 \quad \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2) = \emptyset}{\Gamma; \Delta_1, \Delta_2 \vdash_{\boldsymbol{\alpha}} M_1 \otimes M_2 : \tau_1 \otimes \tau_2}$$

$$\frac{\Gamma; \Delta_1 \vdash_{\boldsymbol{\alpha}} M_1 : \tau_1 \otimes \tau_2 \qquad \Gamma; \Delta_2, a_1:\tau_1, a_2:\tau_2 \vdash_{\boldsymbol{\alpha}} M_2 : \sigma \quad \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2) = \emptyset \quad a_1, a_2 \notin \mathrm{dom}(\Delta_1) \cup \mathrm{dom}(\Delta_2) \quad a_1 \neq a_2}{\Gamma; \Delta_1, \Delta_2 \vdash_{\boldsymbol{\alpha}} \text{let } a_1 \otimes a_2 \text{ be } M_1 \text{ in } M_2 : \sigma}$$

B Admissible Relations

To justify that $\top\top$-closed relations can serve as a notion of admissible relations, we must show that they enjoy the closure properties of Figure 4 in [8].

We have, however, since the mentioned publication been able to simplify the set of axioms somewhat, and to showcase this, we use the closure properties of Figure 4 in [6] instead. (We have shown the old set of axioms to hold.)

We refer to figure 4 in [6] and provide only a part of a formula to hint at which construction we are debating:

$R \subseteq_{Adm} \sigma \times \tau$:A free admissible relational variable is interpreted as the projection into the set of admissible relations. Thus at every point it returns the admissible relation it receives as the last component of its input.

eq_σ: Equality is simply ground contextual equivalence, which is same as Δ, which is $\top\top$-closed.

$\rho(t\ x, u\ y)$: Assume ρ is $\top\top$-closed. We now wish to show $\rho(tM, uN)^{\top\top} \Rightarrow \rho^{\top\top}(tM, uN)$. Writing out the two formulae, we get $\rho(tM, uN)^{\top\top} \Leftrightarrow$

$$\begin{array}{l}\forall f' : \sigma' \multimap !\omega_1, g' : \tau' \multimap !\omega_2. \\ (\forall z' : \sigma', w' : \tau'.\rho(t\ z', u\ w') \supset f'\ z' \Downarrow\Leftrightarrow g'\ w' \Downarrow) \\ \supset f'\ M \Downarrow\Leftrightarrow g'\ N \Downarrow\end{array}$$

and $\rho^{\top\top}(tM, uN) \Leftrightarrow$

$$\begin{array}{l}\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2. \\ (\forall z : \sigma, w : \tau.\rho(z, w) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow) \\ \supset f(t\ M) \Downarrow\Leftrightarrow g(u\ N) \Downarrow\end{array}$$

Assume the formula for $\rho(tM, uN)^{\top\top}$ and that $f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2$ satisfy $\forall z : \sigma, w : \tau.\rho(z, w) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow$. If we instantiate $\rho(tM, uN)^{\top\top}$ with $f' = f \circ t$, $g' = g \circ u$, we get $f(t\ M) \Downarrow\Leftrightarrow g(u\ N) \Downarrow$ as required.

$\rho(x, y) \wedge \rho'(x, y)$: Conjunction is modeled by intersection, so the following neat argument applies: Assume ρ and ρ' to be $\top\top$-closed. Since

$$\rho \cap \rho' \subset \rho \quad \text{and} \quad \rho \cap \rho' \subset \rho'$$

we get

$$\rho^{\top} \subset (\rho \cap \rho')^{\top} \quad \text{and} \quad \rho'^{\top} \subset (\rho \cap \rho')^{\top}$$

so

$$\rho^{\top} \cup \rho'^{\top} \subset (\rho \cap \rho')^{\top}$$

Thus

$$(\rho \cap \rho')^{\top\top} \subset (\rho^{\top} \cup \rho'^{\top})^{\top} \subset \rho^{\top\top} = \rho$$

and likewise $(\rho \cap \rho')^{\top\top} \subset \rho'$, so $(\rho \cap \rho')^{\top\top} \subset \rho \cap \rho'$.

$(x : \tau, y : \sigma).\rho(y, x)$: Let $\widehat{\rho}$ denote $(x : \tau, y : \sigma).\rho(y, x)$. Calculation now shows, that $\widehat{\rho}^{\top\top} = \widehat{\rho^{\top\top}}$. Thus

$$\begin{array}{rcl}\widehat{\rho}^{\top\top}(M, N) = \widehat{\rho^{\top\top}}(M, N) = \rho^{\top\top}(N, M) \\ = \quad \rho(N, M) \quad = \quad \widehat{\rho}(M, N)\end{array}$$

where we obviously use, that ρ is $\top\top$-closed.

$\top$:Writing out $\top^{\top\top}(M, N)$ we get

$$\begin{array}{l}\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2. \\ (\forall z : \sigma, w : \tau.\top(z, w) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow) \\ \supset f\ M \Downarrow\Leftrightarrow g\ N \Downarrow\end{array}$$

Since $\top(z, w)$ holds in general, this is quickly shortened to

$$\begin{array}{l}\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2. \\ (\forall z : \sigma, w : \tau.f\ z \Downarrow\Leftrightarrow g\ w \Downarrow) \supset f\ M \Downarrow\Leftrightarrow g\ N \Downarrow\end{array}$$

which again (due to the very strong requirements on f and g) can be shortened to

$$\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2.\top$$

which is of course just the same as $\top$.

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

Quantifications: All quantifications are proved the same way. We will do $(\forall x : \omega.\rho)^{\top\top} \subset \forall x : \omega.\rho^{\top\top}$. Writing out the two formulae, we get $(\forall x : \omega.\rho)^{\top\top}(M, N) \Leftrightarrow$

$$\begin{array}{l}\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2. \\ (\forall z : \sigma, w : \tau.(\forall x : \omega.\rho_x(z, w)) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow) \\ \supset f\ M \Downarrow\Leftrightarrow g\ N \Downarrow\end{array}$$

and $(\forall x : \omega.\rho^{\top\top})(M, N) \Leftrightarrow$

$$\begin{array}{l}\forall x : \omega. \\ (\forall f : \sigma \multimap !\omega_1, g : \tau \multimap !\omega_2. \\ (\forall z : \sigma, w : \tau.\rho_x(z, w) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow) \\ \supset f\ M \Downarrow\Leftrightarrow g : N \Downarrow)\end{array}$$

Take any $T : \sigma$. Assume the formula for $(\forall x : \omega.\rho)^{\top\top}(M, N)$ and that $f : \sigma \multimap !\omega, g : \tau \multimap !\omega$ satisfy $\forall z : \sigma, w : \tau.\rho_T(z, w) \supset f\ z \Downarrow\Leftrightarrow g\ w \Downarrow$. Then, as $\forall x : \omega.\rho_x(z, w)$ implies $\rho_T(z, w)$, we can plug f and g into $(\forall x : \omega.\rho)^{\top\top}(M, N)$, obtaining $f\ M \Downarrow\Leftrightarrow g : N \Downarrow$ as required.

$\rho \subseteq_{Adm} \sigma \times \tau \wedge \rho' \subseteq \sigma \times \tau \wedge \rho \equiv \rho' \Rightarrow \rho' \subseteq_{Adm} \sigma \times \tau$: We recall, that $\rho \equiv \rho'$ is short hand for $\forall x : \sigma, y : \tau.\rho(x, y) \rightleftarrows \rho'(x, y)$, which in our model translates to actual equality among the relations ρ and ρ'. The statement is then obvious.

C Relational Interpretation

If we write out the definition of **AdmRelCtx** and **AdmRelations** as prescribed in Section 2 we get

Definition 11 (AdmRelCtx).

Objects: $(\overline{\alpha}^n, \overline{\alpha}^m, (A_i)_{i \in S(\overline{\alpha}^{n+m})})$, *where each* A_i *is a set.*
Morphisms:
$(\overline{\sigma}^r, \overline{\tau}^s, (f_i)_{i \in S(\overline{\alpha}^{n+m})}) :$
$(\overline{\alpha}^n, \overline{\alpha}^m, (A_i)_{i \in S(\overline{\alpha}^{n+m})})$
$\to (\overline{\alpha}^r, \overline{\alpha}^s, (B_j)_{j \in S(\overline{\alpha}^{r+s})})$,
such that
- $\overline{\sigma}^r : \overline{\alpha}^n \to \overline{\alpha}^r$ *in* **Kind**
- $\overline{\tau}^s : \overline{\alpha}^m \to \overline{\alpha}^s$ *in* **Kind**
- $f_i : A_i \to B_{S(\overline{\sigma}^m \times \overline{\tau}^s)(i)}$

Definition 12 (AdmRelations).

Objects: $(\overline{\alpha}^n, \overline{\alpha}^m, (A_i)_{i\in S(\overline{\alpha}^{n+m})}, \sigma, \tau, (f_i)_{i\in S(\overline{\alpha}^{n+m})})$, *such that*
- $(\overline{\alpha}^n, \sigma)$ *is an object of* **LinType**
- $(\overline{\alpha}^m, \tau)$ *is an object of* **LinType**
- $(\overline{\alpha}^{n+m}, (f_i)_{i\in S(\overline{\alpha}^{n+m})}) :$
  $(\overline{\alpha}^{n+m}, (A_i)_{i\in S(\overline{\alpha}^{n+m})}) \to V(\overline{\alpha}^{n+m}, \sigma, \tau)$

  in **Context**, *i.e*
  $\forall \overline{\tau}^{n+m} \in S(\overline{\alpha}^{n+m}).\forall a \in A_{\overline{\tau}^{n+m}}.$
  $f_{\overline{\tau}^{n+m}}(a)$
  $\subseteq_{Adm} \sigma[\overline{\tau}^{n+m}/\overline{\alpha}^{n+m}] \times \tau[\overline{\tau}^{n+m}/\overline{\alpha}^{n+m}]$

Morphisms:
$(\overline{\sigma}^r, \overline{\tau}^s, (h_i)_{i\in S(\overline{\alpha}^{n+m})}, [M], [N]) :$
$(\overline{\alpha}^n, \overline{\alpha}^m, (A_i)_{i\in S(\overline{\alpha}^{n+m})}, \sigma, \tau, (f_i)_{i\in S(\overline{\alpha}^{n+m})})$
$\to (\overline{\alpha}^r, \overline{\alpha}^s, (B_i)_{i\in S(\overline{\alpha}^{r+s})}, \omega, \rho, (g_i)_{i\in S(\overline{\alpha}^{r+s})})$

such that
- $(\overline{\sigma}^r, [M]) : (\overline{\alpha}^n, \sigma) \to (\overline{\alpha}^r, \omega)$ *in* **LinType**
- $(\overline{\tau}^s, [N]) : (\overline{\alpha}^m, \tau) \to (\overline{\alpha}^s, \rho)$ *in* **LinType**
- $\forall i \in S(\overline{\alpha}^{n+m}).\ h_i : A_i \to B_{S(\overline{\sigma}^r, \overline{\tau}^s)(i)}$
- $\forall i \in S(\overline{\alpha}^{n+m}).\forall a \in A_i.$

$$f_i(a) \subseteq (V(\overline{\omega}^{r+s}, [M], [N])_j \circ g_j \circ h_i)(a)$$

as illustrated by the diagram:

$$\begin{array}{ccccc} A_i & \xrightarrow[|\cap]{f_i} & V(\sigma,\tau)_i & \subseteq 2^{I(\sigma)\times I(\tau)} & \sigma \quad \tau \\ {\scriptstyle h_i}\downarrow & & \uparrow{\scriptstyle V([M],[N])_j} & & {\scriptstyle [M]}\downarrow \quad \downarrow{\scriptstyle [N]} \\ B_j & \xrightarrow{g_j} & V(\omega,\rho)_j & \subseteq 2^{I(\omega)\times I(\rho)} & \omega \quad \rho \end{array}$$

where $\omega_1 \dots \omega_r$ *are weakened versions of* $\sigma_1 \dots \sigma_r$, $\omega_{r+1} \dots \omega_{r+s}$ *are weakened versions of* $\tau_1 \dots \tau_s$ *and* $j = S(\overline{\sigma}^r, \overline{\tau}^s)(i)$.

Note that in the definition of objects σ and τ has been weakened before V is applied to them, and in the definition of morphisms the same has happened to $[M]$ and $[N]$.

Lemma 8. *J is a map of* PILL *models.*

Proof. It is sufficient to show that J is a strong symmetric monoidal closed functor which preserves the comonad structure on the nose [14]. Then J has an extension to a map of PILL-models.

That the comonad structure is preserved on the nose is an easy consequence of the very syntactic nature of our categories. That J is a strong SMC functor

is almost as easy. Only the constructs !, $\otimes$ and I are not defined directly as J applied to their non-relational counterparts.

For brevity, we only include the proof for tensor. Given $\rho \subseteq_{Adm} \sigma \times \tau$ and $\rho' \subseteq_{Adm} \sigma' \times \tau'$, the tensor relation $\rho \otimes \rho' \subseteq_{Adm} \sigma \otimes \sigma' \times \tau \otimes \tau'$ is defined as

$$\begin{array}{l}(x : \sigma \otimes \sigma', y : \tau \otimes \tau').\forall \alpha, \beta, R \subseteq_{Adm} \alpha \times \beta. \\ \forall t : \sigma \multimap \tau \multimap \alpha, t' : \sigma' \multimap \tau' \multimap \beta.(\rho \multimap \rho' \multimap R)(t, t') \supset \\ R(\text{let } x' \otimes x'' \text{ be } x \text{ in } tx'x'', \text{let } y' \otimes y'' \text{ be } y \text{ in } t'y'y'').\end{array}$$

Thus, for given types σ and τ (in the same fiber) and given terms $M : \sigma \otimes \tau$ and $N : \sigma \otimes \tau$, we wish to compare the statement $M\ J(\sigma \otimes \tau)\ N$ with $M\ J(\sigma) \otimes J(\tau)\ N$. The former is given by

$$M\ \Delta_{\sigma \otimes \tau}\ N$$

while the latter is given by

$$\begin{array}{l}\forall \alpha, \beta, R \subseteq_{Adm} \alpha \times \beta. \\ \forall t : \sigma \multimap \tau \multimap \alpha, t' : \sigma' \multimap \tau' \multimap \beta.(\rho \multimap \rho' \multimap R)(t, t') \supset \\ R(\text{let } x' \otimes x'' \text{ be } M \text{ in } tx'x'', \text{let } y' \otimes y'' \text{ be } N \text{ in } t'y'y''),\end{array}$$

where $\sigma = \sigma'$ because $J(\sigma)$ is a relation on σ. Likewise $\tau = \tau'$. If we assume $M\ \Delta_{\sigma \otimes \tau}\ N$ and introduce the following names

$$\begin{array}{l}r_1 = \Delta_\sigma, \quad r_2 = \Delta_\tau, \quad r_3 = R \\ M_1 = M, \quad M_1' = N \\ M_3 = -; a_1 : \sigma, a_2 : \tau \vdash t\ a_1\ a_2 \\ M_3' = -; a_1' : \sigma, a_2' : \tau \vdash t'\ a_1'\ a_2',\end{array}$$

we know that $(M_1, M_1') \in (r_1 \otimes r_2)^{\top\top}$ and that

$$\begin{array}{rcll}(A, A') \in r_1 & \wedge & (B, B') \in r_2 & \Leftrightarrow \\ A\ \Delta_\sigma A' & \wedge & B\ \Delta_\tau B' & \Rightarrow \\ R(t\ A\ B, t'\ A'\ B') & & & \Leftrightarrow \\ (M_3[A/a_1, B/a_2], M_3'[A'/a_1', B'/a_2']) \in r_3. & & &\end{array}$$

Thus a lemma in [9] applies, telling us that

$$\begin{array}{c}(\text{let } a_1 \otimes a_2 \text{ be } M_1 \text{ in } M_3, \text{let } a_1' \otimes a_2' \text{ be } M_1' \text{ in } M_3') \in r_3 \\ \Updownarrow \\ R(\text{let } x' \otimes x'' \text{ be } M \text{ in } t\ x'\ x'', \text{let } y' \otimes y'' \text{ be } N \text{ in } t'\ y'\ y'')\end{array}$$

Thus $M\ J(\sigma) \otimes J(\tau)\ N$.

For the converse direction choose

$$\begin{array}{l}\alpha = \sigma \otimes \tau, \quad \beta = \sigma \otimes \tau \\ t = t' = \lambda x : \sigma.\lambda y : \tau.x \otimes y \\ R = \Delta_{\sigma \otimes \tau}.\end{array}$$

This gives

$$\text{let } x' \otimes x'' \text{ be } M \text{ in } x' \otimes x''\ \Delta_{\sigma \otimes \tau}\ \text{let } y' \otimes y'' \text{ be } N \text{ in } y' \otimes y'',$$

which quickly reduces to the required $M\ \Delta_{\sigma \otimes \tau}\ N$. □