

A LINEAR CATEGORY OF POLYNOMIAL FUNCTORS (EXTENSIONAL PART)

PIERRE HYVERNAT

Laboratoire de Mathématiques, CNRS UMR 5126 – Université de Savoie, 73376 Le Bourget-du-Lac
Cedex, France

e-mail address: pierre.hyvernats@univ-savoie.fr

URL: <http://lama.univ-savoie.fr/~hyvernats/>

ABSTRACT. We construct a symmetric monoidal closed category of *polynomial endofunctors* (as objects) and *simulation cells* (as morphisms). This structure is defined using universal properties without reference to representing polynomial diagrams and is reminiscent of Day’s convolution on presheaves. We then make this category into a model for intuitionistic linear logic by defining an additive and exponential structure.

INTRODUCTION

Polynomial functors are (generalizations of) functors $X \mapsto \sum_k C_k X^{E_k}$ in the category of sets and functions. Both the “coefficients” C_k and the “exponents” E_k are sets; and sums, products and exponentiations are to be interpreted as disjoint unions, cartesian products and function spaces. All the natural parametrized algebraic datatypes arising in programming can be expressed in this way. For example, the following datatypes are polynomial:

- $X \mapsto \text{List}(X)$ for lists of elements of X , whose polynomial is $\text{List}(X) = \sum_{n \in \mathbb{N}} X^{[n]}$ where $[n] = \{0, \dots, n-1\}$;
- $X \mapsto \text{LBin}(X)$ for “left-leaning” binary trees with nodes in X , whose polynomial can be written as $\text{LBin}(X) = \sum_{t \in T} X^{N(t)}$, where T is the set of *unlabeled* left-leaning trees and $N(t)$ is the set of nodes of t ;
- $X \mapsto \text{Term}_S(X)$ for well-formed terms built from a first-order multi-sorted signature S with variables of sort τ taken in X_τ .

In the last example X is a family of sets indexed by sorts rather than a single set, and expressing it as a polynomial requires “indexed” or “multi-variables” polynomial functors.

Because of this, those functors have recently received a lot of attention from a computer science point of view. In this context, they are often called *containers* [AAG05, MA09] and

Received by the editors May 13, 2014.

2012 ACM CCS: [Theory of computation]: Semantics and reasoning—Program semantics—Categorical Semantics; Logic—Linear logic.

Key words and phrases: polynomial functors; linear logic; Day convolution.

This work was partially funded by the French ANR project récré ANR-11-BS02-0010.

coefficients and exponents are called *shapes* and *positions*. An early use of them (with yet another terminology) goes back to Petersson and Synek [PS89]: *tree-sets* are a generalization of so called W-types from dependent type theory. They are inductively generated and are related to the free monads of arbitrary polynomial functors. In the presence of extensional equality, they can be encoded using usual W-types [GK09].

Polynomial functors form the objects of a category with a very rich structure. The objects (i.e., polynomial functors), the morphisms (called *simulations*) and many operations (coproduct, tensor, composition, etc.) can be interpreted using a “games” intuition (refer to [HH06, Hyv14] for more details):

- a polynomial is a two-players game, where moves of the first player are given by its “coefficients” and counter-moves of the opponent are given by its “exponents”,
- a simulation between two such games is a witness for a kind of back-and-forth property between them.

This category has enough structure to model intuitionistic linear logic [Hyv14]. The simplest way to define this structure is to use representations of polynomial functors, called *polynomial diagrams* as the objects. The fact that such representations give rise to functors isn’t relevant! This paper gives a functorial counterpart: a model for intuitionistic linear logic where formulas are interpreted by polynomial *functors*, without reference to their representations.

The difference between the two approaches is subtle. Differentiation of plain polynomials over the real numbers provides a useful analogy. It can be defined in two radically different ways:

(1) on *representations* of polynomials with the formula:

$$P = \sum_{k=0}^n a_k X^k \quad \mapsto \quad P' = \sum_{k=1}^n k a_k X^{k-1} ,$$

(2) on polynomials *as functions* with the formula:

$$F \quad \mapsto \quad F' = x \mapsto \lim_{\varepsilon \rightarrow 0} \frac{F(x + \varepsilon) - F(x)}{\varepsilon} .$$

Both definitions are useful and neither is obviously reducible to the other. The first one is easier to work with for concrete polynomials but the second one applies to a larger class of functions. The model of intuitionistic linear logic previously defined [Hyv14] was “intensional”: just like point (1), it used representations of polynomial functors. The model described here is “extensional”: just like point (2), it applies to arbitrary functors, even though it needs not be defined for non polynomial ones.

More precisely, the category defined in [Hyv14] is equivalent (Proposition 2.6) to a subcategory of a category of arbitrary functors with simulations, where the tensor and its adjoint are not always defined.

We first (Section 1) recall some notions about polynomial functors. We then define the category of polynomial functors with simulations and show that it is symmetric monoidal closed (Section 2). We relate this structure to a generalized version of Day’s convolution product on presheaves in Section 2.5. We finish by showing how the additive and exponential structure from [Hyv14] can be recovered.

Related Works. The starting point of this work is the characterization of strong natural transformations between polynomial functors due to N. Gambino and J. Kock [GK09]. However, our notion of morphism between polynomial functors is more general than what appears in [Koc09, GK09] or [AAG05, MA09] (where polynomial functors are called *indexed containers*). In particular, there can be morphisms between polynomial functors that do not share their domains and codomains.

Another inspiration is the model of “predicate transformers” from [Hyv04]. This is a model for *classical* linear logic where formulas are interpreted by monotonic operators on subsets and proofs are interpreted by “simulations”. We show here that “monotonic operator on $\mathcal{P}(I)$ ” can be replaced by “functor on \mathbf{Set}/I ” with the following restrictions:

- we give up classical logic;
- we only consider *polynomial* functors.

1. PRELIMINARIES: POLYNOMIAL FUNCTORS

Basic knowledge about locally cartesian closed categories and their internal language (extensional dependent type theory) is assumed throughout the paper. Refer to Appendix A and B for the relevant definitions and notation. The first half of [GK09] is of crucial importance and we start by recalling some results, referring to the original article for details. From now on, \mathbb{C} will always denote a locally cartesian closed category.

1.1. Polynomials and Polynomial Functors. A polynomial is, in the usual sense, a function of several variables that can be written as a sum of monomials, where each monomial is a product of a (constant) coefficient and several variables. A polynomial functor is similar, with the following differences

- it acts on sets rather than numbers;
- sums and products are the corresponding set-theoretic operations;
- it may have arbitrarily many arguments: instead of a tuple (X_1, \dots, X_n) of variables, it acts on *families* $(X_i)_{i \in I}$, for a given set I ;
- the sum of monomial isn’t necessarily finite and can be indexed by any set;
- because $C \times Y \cong \sum_{c \in C} Y$, we don’t need constant coefficients in monomials;
- each monomial is an indexed product of variables X_i .

A single polynomial functor is thus made up from the following data:

- a set I indexing the variables,
 - a set A indexing the sum of monomials,
 - an A -indexed family of sets $(D_v)_{v \in A}$ where for each $v \in A$, i.e., for each monomial appearing in the sum, the set D_v indexes the variables composing the monomial,
 - for each $v \in A$, a function $D_v \rightarrow I$ giving the indices of the variables of the monomial.
- For example, if the monomial uses a single variable, this function is constant...

The corresponding functor is given by

$$(X_i)_{i \in I} \mapsto \sum_{v \in A} \prod_{u \in D_v} X_{n(u)}.$$

Categorically speaking, a K -indexed family of sets can be represented by a function with codomain K . If $\gamma : S \rightarrow K$, the associated family Γ will be given by $\Gamma_k = \gamma^{-1}(k)$. A polynomial functor can thus be represented with

- a set I indexing the variables;
- a set A indexing the sum;
- a function $d : D \rightarrow A$ giving an A -indexed family of sets indexing the products of variables;
- a function $n : D \rightarrow I$ giving, for each variable in each product, its index.

Written in full, the corresponding functor is

$$(X_i)_{i \in I} \mapsto \sum_{v \in A} \prod_{u \in d^{-1}(v)} X_{n(u)} .$$

In order to compose such functors, we need to consider functors acting on families of sets, and giving families of sets as a result. We can thus have polynomial functors taking I -indexed families and giving J -indexed families of sets. Such a functor is simply a J -indexed family of functors in the above sense and is thus of the form:

$$(X_i)_{i \in I} \mapsto \left(\sum_{v \in A_j} \prod_{u \in D_v} X_{n(u)} \right)_{j \in J} . \quad (1.1)$$

The only difference is that instead of having a set A indexing the sum, we have a J -indexed family of sets, each A_j indexing the sum of the j component of the functor. Categorically speaking, it amounts to replacing the set A by a function $\alpha : A \rightarrow J$ representing a J -indexed family of sets. The following definition now makes sense in any category:

Definition 1.1. If I and J are objects of \mathbb{C} , a *polynomial diagram* from I to J is a diagram P in \mathbb{C} of the shape

$$P : I \xleftarrow{n} D \xrightarrow{d} A \xrightarrow{a} J .$$

We write $\text{PolyDiag}_{\mathbb{C}}[I, J]$ for the collection of polynomial diagrams from I to J .

Definition 1.2. For each $P \in \text{PolyDiag}_{\mathbb{C}}[I, J]$, there is an associated functor $\llbracket P \rrbracket$ from \mathbb{C}/I to \mathbb{C}/J :

$$\llbracket P \rrbracket : \mathbb{C}/I \xrightarrow{\Delta_n} \mathbb{C}/D \xrightarrow{\Pi_d} \mathbb{C}/A \xrightarrow{\Sigma_a} \mathbb{C}/J .$$

$\llbracket P \rrbracket$ is called the *extension* of P and P is called a *representation* of $\llbracket P \rrbracket$. Any functor naturally isomorphic to some $\llbracket P \rrbracket$ is called a *polynomial functor*. We write $\text{PolyFun}_{\mathbb{C}}[I, J]$ for the collection of such functors.

In the locally cartesian closed category of sets and functions, the operations Δ_n , Π_d and Σ_a are given by:

- $\Delta_n : (X_i)_{i \in I} \mapsto (X_{n(d)})_{d \in D}$
- $\Pi_d : (Y_u)_{u \in D} \mapsto (\prod_{u \in d^{-1}(a)} Y_u)_{a \in A}$
- $\Sigma_a : (Z_v)_{v \in A} \mapsto (\sum_{v \in a^{-1}(j)} Z_v)_{j \in J}$

and the extension of a polynomial functor corresponds exactly to the formula (1.1). We have

Proposition 1.3. *The composition of two polynomial functors is a polynomial functor.*

Sketch of proof. The composition $Q \circ P$ of $P \in \text{PolyDiag}_{\mathbb{C}}[I, J]$ and $Q \in \text{PolyDiag}_{\mathbb{C}}[J, K]$ uses no less than four pullbacks:

$$Q \circ P \stackrel{\text{def}}{=} \begin{array}{c} \bullet \longrightarrow \bullet \longrightarrow \bullet \\ \swarrow \quad \downarrow \quad \searrow \\ \bullet \longrightarrow \bullet \longrightarrow \bullet \\ \swarrow \quad \downarrow \quad \searrow \\ D \longrightarrow A \longrightarrow E \longrightarrow B \\ \swarrow \quad \downarrow \quad \searrow \\ I \quad \quad J \quad \quad K \end{array} \quad (1.2)$$

where square (i) is a distributivity square (diagram (A.1) in Appendix A). One can then show that

$$[[Q] \circ [P]] \cong [[Q \circ P]]$$

by a sequence of Beck-Chevalley and distributivity isomorphisms (see Appendix A). \square

The identity functor from \mathbb{C}/I to itself is trivially the extension of the polynomial

$$I \xleftarrow{1} I \xrightarrow{1} I \xrightarrow{1} I \quad .$$

We obtain a *bicategory* $\text{PolyDiag}_{\mathbb{C}}$ where objects are objects of \mathbb{C} and morphisms are polynomial diagrams; and a *category* $\text{PolyFun}_{\mathbb{C}}$ where objects are slice categories and morphisms are polynomial functors.

1.2. Strong Natural Transformations. Each polynomial functor P from \mathbb{C}/I to \mathbb{C}/J is equipped with a *strength*:

$$\tau_{A,x} \quad : \quad A \odot P(x) \rightarrow P(A \odot x) \quad ,$$

naturally in $A \in \mathbb{C}$ and $x \in \mathbb{C}/I$, where $A \odot x \stackrel{\text{def}}{=} \Sigma_x \Delta_x \Delta_I(A)$. A natural transformation between two strong functors is itself *strong* when it is compatible with their strengths. This gives the category $\text{PolyFun}_{\mathbb{C}}$ a 2-category structure: objects are slice categories, morphisms are polynomial functors and 2-cells are strong natural transformations. Note that when the base category \mathbb{C} is Set , all endofunctors are strong, and so are all natural transformations.

The following proposition is crucial as it allows to represent strong natural transformations between polynomial functors by diagrams inside the category \mathbb{C} [GK09].

Proposition 1.4. *Every strong natural transformation $\rho : P_1 \Rightarrow P_2$ between polynomial functors can be uniquely represented (up-to a choice of pullbacks) by a diagram*

$$P_1 : \begin{array}{c} I \xleftarrow{n_1} D_1 \xrightarrow{d_1} A_1 \xrightarrow{a_1} J \\ \parallel \quad \uparrow \beta \quad \parallel \quad \parallel \\ \quad \quad X \xrightarrow{f} A_1 \\ \quad \quad \downarrow g \quad \downarrow \alpha \\ I \xleftarrow{n_2} D_2 \xrightarrow{d_2} A_2 \xrightarrow{a_2} J \end{array} \quad . \quad (1.3)$$

The strong natural transformation associated to such a diagram is defined as

$$\begin{aligned}
\Sigma_{a_1} \Pi_{d_1} \Delta_{n_1} &\Longrightarrow \Sigma_{a_1} \Pi_{d_1} \Pi_{\beta} \Delta_{\beta} \Delta_{n_1} && (\text{unit of } \Delta_{\beta} \dashv \Pi_{\beta}) \\
&\cong \Sigma_{a_1} \Pi_f \Delta_g \Delta_{n_2} && (n_1 \beta = n_2 g \text{ and } d_1 \beta = f) \\
&\cong \Sigma_{a_1} \Delta_{\alpha} \Pi_{d_2} \Delta_{n_2} && (\text{Beck-Chevalley isomorphism}) \\
&\cong \Sigma_{a_2} \Sigma_{\alpha} \Delta_{\alpha} \Pi_{d_2} \Delta_{n_2} && (a = b\alpha) \\
&\Longrightarrow \Sigma_{a_2} \Pi_{d_2} \Delta_{n_2} && (\text{counit of } \Sigma_{\alpha} \dashv \Delta_{\alpha}) .
\end{aligned}$$

A corollary to Proposition 1.4 is

Corollary 1.5. *If $\llbracket P_1 \rrbracket \cong \llbracket P_2 \rrbracket$ in $\text{PolyFun}_{\mathbb{C}}$, then P_1 and P_2 are related by*

$$\begin{array}{c}
P_1 : \quad I \xleftarrow{n_1} D_1 \xrightarrow{d_1} A_1 \xrightarrow{a_1} J \\
\quad \quad \quad \parallel \quad \quad \quad \uparrow \quad \quad \quad \downarrow \quad \quad \quad \parallel \\
P_2 : \quad I \xleftarrow{n_2} D_2 \xrightarrow{d_2} A_2 \xrightarrow{a_2} J \quad .
\end{array}$$

This is particularly important as it means that instead of working on polynomial functors up-to strong natural isomorphism, we can work on their representing polynomials.

1.3. Spans and Polynomial Functors. There are two ways to lift a span to a polynomial:

Definition 1.6. Given a span $R = \langle f, g \rangle : I \leftarrow X \rightarrow J$, we define two polynomials in $\text{PolyDiag}_{\mathbb{C}}[I, J]$:

$$\langle R \rangle \stackrel{\text{def}}{=} I \xleftarrow{f} X \xrightarrow{1} X \xrightarrow{g} J$$

and

$$[R] \stackrel{\text{def}}{=} I \xleftarrow{f} X \xrightarrow{g} J \xrightarrow{1} J \quad .$$

Any functor of the form $\llbracket \langle R \rangle \rrbracket$ is called a *linear* polynomial functor.

The terminology “linear” comes from the fact that $\llbracket \langle R \rangle \rrbracket$ commutes with arbitrary colimits. More precisely, we have

Lemma 1.7. *If one writes R^{\sim} for the span R with its “legs” reversed, we have an adjunction*

$$\llbracket \langle R \rangle \rrbracket \dashv \llbracket [R^{\sim}] \rrbracket \quad .$$

Proof. If R is $\langle f, g \rangle$:

$$\begin{array}{ccc}
& X & \\
f \swarrow & & \searrow g \\
I & & J
\end{array} \quad ,$$

then the extension of $\langle R \rangle$ is $\Delta_f \Sigma_g$ and the extension of $[R^{\sim}]$ is $\Delta_g \Pi_f$. The result follows from the two adjunctions $\Delta_f \dashv \Pi_f$ and $\Sigma_g \dashv \Delta_g$. \square

Composition of spans via pullbacks and composition of polynomials are compatible:

Lemma 1.8. *The operations $\langle _ \rangle$ and $[_]$ from $\text{Span}_{\mathbb{C}}$ to $\text{PolyDiag}_{\mathbb{C}}$ are functorial, in a “bicategorical” sense.*

2. SYMMETRIC MONOIDAL CLOSED STRUCTURE

2.1. SMCC Structure for Polynomial *Diagrams*. We start by recalling the main definition and result from [Hyv14].

Definition 2.1. The category $\text{PDSim}_{\mathbb{C}}$ has:

- “endo” polynomial diagrams $I \leftarrow D \rightarrow A \rightarrow I$ as objects
- equivalence classes of “simulation diagrams” as morphisms, where a simulation diagram from $P_1 = I_1 \leftarrow D_1 \rightarrow A_1 \rightarrow I_1$ to $P_2 = I_2 \leftarrow D_2 \rightarrow A_2 \rightarrow I_2$ is given by a diagram like

$$\begin{array}{ccccccc}
 P_1 : & & I_1 & \longleftarrow & D_1 & \longrightarrow & A_1 & \longrightarrow & I_1 \\
 & & \uparrow r_1 & & \uparrow \beta & & \uparrow & \lrcorner & \uparrow r_1 \\
 & & R & \xleftarrow{\gamma} & \cdots & \xrightarrow{\quad} & R & & \\
 & & \downarrow r_2 & & \downarrow & \lrcorner & \downarrow \alpha & & \downarrow r_2 \\
 P_2 : & & I_2 & \longleftarrow & D_2 & \longrightarrow & A_2 & \longrightarrow & I_2
 \end{array}
 .$$

The equivalence relation between such diagrams is detailed in [Hyv14] and corresponds to the equivalence between spans that form the sides of simulations.

We have ([Hyv14]):

Proposition 2.2. *The operation \otimes that acts on objects in a pointwise manner:*

$$P_1 \otimes P_2 \stackrel{\text{def}}{=} I_1 \times I_2 \xleftarrow{n_1 \times n_2} D_1 \times D_2 \xrightarrow{d_1 \times d_2} A_1 \times A_2 \xrightarrow{a_1 \times a_2} J_1 \times J_2$$

is a tensor product. It gives the category $\text{PDSim}_{\mathbb{C}}$ a symmetric monoidal closed structure: there is a functor $_ \multimap _ : \text{PDSim}_{\mathbb{C}}^{\text{op}} \times \text{PDSim}_{\mathbb{C}} \rightarrow \text{PDSim}_{\mathbb{C}}$ and an isomorphism

$$\text{PDSim}_{\mathbb{C}}[P_1 \otimes P_2, P_3] \cong \text{PDSim}_{\mathbb{C}}[P_1, P_2 \multimap P_3],$$

natural in P_1 and P_3 .

Seen from the angle of “games semantics” hinted at in the introduction, this operation is a kind of synchronous, “lockstep” parallel composition: a move in the tensor of two games *must be* a move in each of the games, and a counter-move / response from the opponent *must be* a response for each move, in each of the two games.

2.2. Simulations. We start by characterizing simulations between polynomials as special 2-cells involving their extensions.

Proposition 2.3. *If P_1 and P_2 are polynomial diagrams and R is a span, any 2-cell*

$$\begin{array}{ccc}
 \mathbb{C}/I_1 & \xrightarrow{[[P_1]]} & \mathbb{C}/I_1 \\
 \downarrow [\langle R \rangle] & \swarrow \rho & \downarrow [\langle R \rangle] \\
 \mathbb{C}/I_2 & \xrightarrow{[[P_2]]} & \mathbb{C}/I_2
 \end{array}$$

(where ρ is a strong natural transformation) is uniquely represented (up-to a choice of pullbacks) by a diagram of the shape

$$\begin{array}{ccccc}
 P_1 : & I_1 & \xleftarrow{n_1} & D_1 & \xrightarrow{d_1} & A_1 & \xrightarrow{a_1} & I_1 \\
 & \uparrow r_1 & & \uparrow \beta & & \uparrow & & \uparrow r_1 \\
 & R & \xleftarrow{\gamma} & R \cdot D_2 & \dashrightarrow & R \cdot A_1 & \dashrightarrow & R \\
 & \downarrow r_2 & & \downarrow & & \downarrow \alpha & & \downarrow r_2 \\
 P_2 : & I_2 & \xleftarrow{n_2} & D_2 & \xrightarrow{d_2} & A_2 & \xrightarrow{a_2} & I_2
 \end{array} \quad . \quad (2.1)$$

Proof. As compositions of polynomial functors, both $[[\langle R \rangle]][[P_1]]$ and $[[P_2]][[\langle R \rangle]]$ are polynomial (Proposition 1.3). We can use Proposition 1.4 to represent the strong natural transformation ρ by the following diagram:

$$\begin{array}{ccccccc}
 I_1 & \xleftarrow{n_1} & D_1 & \xrightarrow{d_1} & A_1 & \xrightarrow{a_1} & I_1 \\
 \parallel & & \uparrow & \lrcorner & \uparrow & \lrcorner & \uparrow r_1 \\
 & & X & \longrightarrow & R \cdot A_1 & \longrightarrow & R \\
 & & \uparrow & & \parallel & & \downarrow r_2 \\
 I_1 & & U & \dashrightarrow & R \cdot A_1 & & I_2 \\
 \uparrow r_1 & & \downarrow & \lrcorner & \downarrow & & \parallel \\
 R & \xleftarrow{\quad} & Y & \xleftarrow{\epsilon} & \bullet & \longrightarrow & \bullet \\
 \downarrow r_2 & & \downarrow & \lrcorner & \downarrow & & \downarrow \\
 I_2 & \xleftarrow{n_2} & D_2 & \xrightarrow{d_2} & A_2 & \xrightarrow{a_2} & I_2
 \end{array} \quad (2.2)$$

where (i) is a distributivity square. The plain arrows represent the polynomials $\langle R \rangle P_1$ and $P_2 \langle R \rangle$, whose extensions are $[[\langle R \rangle]][[P_1]]$ and $[[P_2]][[\langle R \rangle]]$; and the dashed arrows represent the strong natural transformation between them, as in diagram (1.3).

Going from diagram (2.2) to diagram (2.1) is easy: just follow the arrows and use the pullback lemma to show that square $(U, R \cdot A_1, A_2, D_2)$ is a pullback and that U is indeed isomorphic to $R \cdot D_2$. The morphisms α , β and γ can be read on the diagram.

Going from diagram (2.1) to diagram (2.2) is slightly messier. We choose U to be $R \cdot D_2$ and look at the following:

$$\begin{array}{ccccccc}
 I_1 & \xleftarrow{n_1} & D_1 & \xrightarrow{d_1} & A_1 & \xrightarrow{a_1} & J_1 \\
 \parallel & & \uparrow & \lrcorner & \uparrow & & \uparrow r_1 \\
 & & X & \xrightarrow{\quad} & R \cdot A_1 & \longrightarrow & R \\
 & & \uparrow h & & \parallel & & \downarrow r_2 \\
 I_1 & \xleftarrow{\gamma} & R \cdot D_2 & \dashrightarrow & R \cdot A_1 & & J_2 \\
 \uparrow l_1 & & \downarrow g & \text{(ii)} & \downarrow f & & \parallel \\
 R & \xleftarrow{\quad} & Y & \xleftarrow{\epsilon} & \cdot & \xrightarrow{\quad} & \cdot \\
 \downarrow l_2 & & \downarrow d & \text{(i)} & \downarrow \alpha & & \parallel \\
 I_2 & \xleftarrow{n_2} & D_2 & \xrightarrow{d_2} & A_2 & \xrightarrow{a_2} & J_2
 \end{array}$$

The morphisms α , β and γ come from diagram (2.1).

To define f and h , write φ for the natural isomorphism $\mathbb{C}/U[\Sigma_{k-}, -] \implies \mathbb{C}/V[-, \Delta_{k-}]$ and ψ for the natural isomorphism $\mathbb{C}/V[\Delta_{k-}, -] \implies \mathbb{C}/U[-, \Pi_{k-}]$. In particular, ϵ is $\psi^{-1}(1)$. Then:

- f is constructed from α and γ :

$$\begin{aligned}
 \gamma &\in \mathbb{C}/I_2[\Sigma_{n_2} \Delta_{d_2}(\alpha), l_2] && \text{see diagram (2.1)} \\
 \Leftrightarrow g &\stackrel{\text{def}}{=} \varphi(\gamma) \in \mathbb{C}/D_2[\Delta_{d_2}(\alpha), \Delta_{n_2}(l_2)] \\
 \Leftrightarrow f &\stackrel{\text{def}}{=} \psi\varphi(\gamma) \in \mathbb{C}/A_2[\alpha, \Pi_{d_2} \Delta_{n_2}(l_2)].
 \end{aligned}$$

By naturality of ψ^{-1} , the following commutes:

$$\begin{array}{ccc}
 \mathbb{C}/A_2[\Pi_{d_2}(d), \Pi_{d_2}(d)] & \xrightarrow{\psi^{-1}} & \mathbb{C}/D_2[\Delta_{d_2} \Pi_{d_2}(d), d] \\
 \downarrow - \circ f & & \downarrow - \circ \vec{\Delta}_{d_2}(f) \\
 \mathbb{C}/A_2[\alpha, \Pi_{d_2}(d)] & \xrightarrow{\psi^{-1}} & \mathbb{C}/D_2[\Delta_{d_2}(\alpha), d]
 \end{array}$$

where $\vec{\Delta}_{d_2}(-)$ is the action of the functor Δ_{d_2} on morphisms. Starting from the identity, we get $\psi^{-1}(1) \circ \vec{\Delta}_{d_2}(f) = \psi^{-1}(f)$, i.e., $\epsilon \circ \vec{\Delta}_{d_2}(f) = g$. This shows that the triangle (ii) commutes.

- We can then construct h from β by using the fact that X is a pullback.

The only remaining thing to check is that the upper left rectangle commutes. It follows from $r_1 \gamma = n_1 \beta$ in diagram (2.1) and the construction. \square

Proposition 2.3 makes it natural to define simulations for arbitrary endofunctors:

Definition 2.4. If F_1 and F_2 are two endofunctors over \mathbb{C}/I_1 and \mathbb{C}/I_2 , a *simulation from F_1 to F_2* is given by a span $I_1 \leftarrow R \rightarrow I_2$ and a 2-cell of the form

$$\begin{array}{ccc} \mathbb{C}/I_1 & \xrightarrow{F_1} & \mathbb{C}/I_1 \\ \downarrow & \swarrow \rho & \downarrow \\ \llbracket \langle R \rangle \rrbracket & & \llbracket \langle R \rangle \rrbracket \\ \downarrow & \swarrow & \downarrow \\ \mathbb{C}/I_2 & \xrightarrow{F_2} & \mathbb{C}/I_2 \end{array}$$

where ρ is a natural transformation. A simulation (R, ρ) is *equivalent* to a simulation (R', ρ') iff $F_2 \varepsilon \circ \rho = \rho' \circ \varepsilon F_1$ for some natural isomorphism $\varepsilon : \llbracket \langle R \rangle \rrbracket \rightarrow \llbracket \langle R' \rangle \rrbracket$.

The category $\text{FSim}_{\mathbb{C}}$ is defined with:

- endofunctors over slices of \mathbb{C} as objects,
- equivalence classes of simulations as morphisms.

It is routine to check that this gives a category. (Recall that the composition of two linear functors is again linear by Lemma 1.8). Note that the notion of equivalence of simulations is inherited from $\text{Span}_{\mathbb{C}}$: by Corollary 1.5 an isomorphism between $\llbracket \langle R \rangle \rrbracket$ and $\llbracket \langle R' \rangle \rrbracket$ amounts to a span isomorphism between R and R' . As a particular subcategory, we have

Definition 2.5. The category $\text{PFSim}_{\mathbb{C}}$ is the subcategory of $\text{FSim}_{\mathbb{C}}$ with

- *polynomial* endofunctors over slices of \mathbb{C} as objects,
- equivalence classes of *strong* simulations as morphisms

where a simulation (R, ρ) is strong if and only if the natural transformation ρ is strong.

There is a functor from $\text{PDSim}_{\mathbb{C}}$ to $\text{PFSim}_{\mathbb{C}}$ that sends a polynomial diagram to its corresponding polynomial functor and a simulation diagram to its simulation cell. This functor is

- surjective on objects by the definition of polynomial functor,
- full and faithful by Proposition 2.3.

This implies that

Proposition 2.6. *The category $\text{PDSim}_{\mathbb{C}}$ and the category $\text{PFSim}_{\mathbb{C}}$ are equivalent.*

2.3. Tensor Product. Recall that the tensor of two polynomials is the “pointwise cartesian product”:

$$P_1 \otimes P_2 \stackrel{\text{def}}{=} I_1 \times I_2 \xleftarrow{n_1 \times n_2} D_1 \times D_2 \xrightarrow{d_1 \times d_2} A_1 \times A_2 \xrightarrow{a_1 \times a_2} I_1 \times I_2 \quad .$$

This gives rise to an operation on polynomial functors: $\llbracket P_1 \rrbracket \otimes \llbracket P_2 \rrbracket \stackrel{\text{def}}{=} \llbracket P_1 \otimes P_2 \rrbracket$. However, this definition is intensional because it acts on polynomial diagrams, i.e., on representations of polynomial functors. In particular, it doesn’t even make sense for functors that are not polynomial. We will now show that it is possible to characterize $\llbracket P_1 \otimes P_2 \rrbracket$ by a universal property relying only on $\llbracket P_1 \rrbracket$ and $\llbracket P_2 \rrbracket$, thus giving an extensional definition of the tensor of polynomial functors.

To avoid confusion, we will write $\square : \mathbb{C}/A \times \mathbb{C}/B \rightarrow \mathbb{C}/_{A \times B}$ for the functor sending (x, y) in $\mathbb{C}/A \times \mathbb{C}/B$ to $x \times y$ in $\mathbb{C}/_{A \times B}$ and (f, g) in $\mathbb{C}/_A[x, x'] \times \mathbb{C}/_B[y, y']$ to $f \times g$ in $\mathbb{C}/_{A \times B}[x \times x', y \times y']$, we have:

Proposition 2.7. *Let P_1 and P_2 be polynomial functors, the polynomial functor $P_1 \otimes P_2$ is a left Kan-extension along \square : it is universal s.t.*

$$\begin{array}{ccc} \mathbb{C}/_{I_1} \times \mathbb{C}/_{I_2} & \xrightarrow{\square} & \mathbb{C}/_{I_1 \times I_2} \\ P_1 \times P_2 \downarrow & \nearrow \varepsilon & \downarrow P_1 \otimes P_2 \\ \mathbb{C}/_{I_1} \times \mathbb{C}/_{I_2} & \xrightarrow{\square} & \mathbb{C}/_{I_1 \times I_2} \end{array} .$$

More precisely, $P_1 \otimes P_2 = \text{Lan}_{\square} (P_1(-) \square P_2(-))$ in the category of endofunctors with natural transformations.

Corollary 2.8. *If \mathbb{C} has copowers, denoted by \odot , we can express left Kan-extensions using coends. We then have*

$$P_1 \otimes P_2(r) = \int^{x, y} \mathbb{C}/_{I_1 \times I_2}[x \square y, r] \odot (P_1(x) \square P_2(y)) .$$

This definition is reminiscent of the tensor of predicate transformers (Definition 7 in [Hyv04]): if $P_1 : \mathcal{P}(S_1) \rightarrow \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \rightarrow \mathcal{P}(S_2)$ are monotonic operators on subsets, then

$$\begin{array}{ccc} P_1 \otimes P_2 : \mathcal{P}(S_1 \times S_2) & \rightarrow & \mathcal{P}(S_1 \times S_2) \\ r & \mapsto & \bigcup_{x \times y \subseteq r} P_1(x) \times P_2(y) \end{array} .$$

The proof of proposition 2.7 makes heavy use of the internal language of locally cartesian closed categories. First note that a polynomial $I \leftarrow D \rightarrow A \rightarrow I$ can be described by the following judgments:

- (1) “ $\dashv I$ type” for the object I (slice over $\mathbf{1}$),
- (2) “ $i : I \vdash A(i)$ type” for the slice $a : A \rightarrow I$ in $\mathbb{C}/_I$,
- (3) “ $i : I, a : A(i) \vdash D(i, a)$ type” for the slice $d : D \rightarrow A$ in $\mathbb{C}/_A$,
- (4) “ $i : I, a : A(i), d : D(i, a) \vdash n(i, a, d) : I$ ” for the morphism $n : D \rightarrow I$.

Proof of proposition 2.7. Because $\Sigma_{a_1 \times a_2}(- \square -) = \Sigma_{a_1}(-) \square \Sigma_{a_2}(-)$ is a left-adjoint, it commutes with all colimits, including left Kan-extensions. We thus have

$$\begin{aligned} \text{Lan}_{\square} (P_1(-) \square P_2(-)) &= \text{Lan}_{\square} \left(\Sigma_{a_1} \Pi_{d_1} \Delta_{n_1}(-) \square \Sigma_{a_2} \Pi_{d_2} \Delta_{n_2}(-) \right) \\ &= \text{Lan}_{\square} \left((\Sigma_{a_1} \square \Sigma_{a_2}) (\Pi_{d_1} \Delta_{n_1}(-) \square \Pi_{d_2} \Delta_{n_2}(-)) \right) \\ &= \text{Lan}_{\square} \left(\Sigma_{a_1 \times a_2} (\Pi_{d_1} \Delta_{n_1}(-) \square \Pi_{d_2} \Delta_{n_2}(-)) \right) \\ &= \Sigma_{a_1 \times a_2} \text{Lan}_{\square} \left(\Pi_{d_1} \Delta_{n_1}(-) \square \Pi_{d_2} \Delta_{n_2}(-) \right) . \end{aligned}$$

To save some parenthesis, we will write $n_1 \cdot d_1$ instead of $n_1(i_1, a_1, d_1)$ and similarly for $n_2 \cdot d_2$. We write F_1 and F_2 for $\Pi_{d_1} \Delta_{n_1}$ and $\Pi_{d_2} \Delta_{n_2}$. Internally, F_1 is thus $X \mapsto \prod_{d_1} X(n_1 \cdot d_1)$ and $F_1 \otimes F_2$ is $R \mapsto \prod_{d_1, d_2} R(n_1 \cdot d_1, n_2 \cdot d_2)$.

To reduce verbosity, we will ignore the dependency on A_1 and A_2 , i.e., we'll “pretend” both are equal to $\mathbf{1}$. To correct that, one simply needs to add “ $a_1:A_1(i_1), a_2:A_2(i_2)$ ” to all contexts and make the constructions depend on a_1 and a_2 . Recall that if X and Y are families indexed by U and V , $X \square Y$ is the family “ $\lambda\langle u, v \rangle. X(u) \times Y(v)$ ” indexed by $U \times V$. We use the same notation for functions: $f \square g$ stands for $\lambda\langle u, v \rangle. \langle f(u), g(v) \rangle$. We will

- (1) construct a natural transformation $\varepsilon : F_1(_) \square F_2(_) \Longrightarrow F_1 \otimes F_2(_ \square _)$,
- (2) show that ε is universal: if $\rho : F_1(_) \square F_2(_) \Longrightarrow F(_ \square _)$, we construct a unique transformation $\Theta : F_1 \otimes F_2(_) \Longrightarrow F(_)$ such that $\Theta\varepsilon = \rho$.

In the internal language, if F and G are two functors from \mathbb{C}/U to \mathbb{C}/V , a natural transformation α from F to G , takes the form of

- families of functions $v : V \vdash \alpha_X(v) : F(X)(v) \rightarrow G(X)(v)$ for any U -indexed type X ,
- subject to naturality: if $f : X \rightarrow Y$, then $\alpha_Y F_f = G_f \alpha_X$, i.e.,

$$\alpha_Y(v)(F_f(v)(y)) = G_f(v)(\alpha_X(v)(y))$$

whenever $u : U \vdash X(u)$ and $\vdash v : V$.

The transformation $\varepsilon : \prod_{d_1} \Delta_{n_1}(_) \square \prod_{d_2} \Delta_{n_2}(_) \Longrightarrow \prod_{d_1 \times d_2} \Delta_{n_1 \times n_2}(_ \square _)$ is defined as follows: for families X and Y over I_1 and I_2 and $\langle h_1, h_2 \rangle : \prod_{d_1} X(n_1 \cdot d_1) \times \prod_{d_2} Y(n_2 \cdot d_2)$, we put

$$\vdash \varepsilon_{X,Y} \langle h_1, h_2 \rangle \stackrel{\text{def}}{=} \lambda \langle d_1, d_2 \rangle. \langle h_1(d_1), h_2(d_2) \rangle : \prod_{d_1, d_2} X(n_1 \cdot d_1) \times Y(n_2 \cdot d_2).$$

It is easy to check that this is natural. (Categorically speaking, $\varepsilon_{X,Y} \langle h_1, h_2 \rangle$ is just $h_1 \times h_2$.)

To check universality, let $\rho : \prod_{d_1} \Delta_{n_1}(_) \square \prod_{d_2} \Delta_{n_2}(_) \Longrightarrow F(_ \square _)$ for some functor F . We define the natural transformation $\Theta : \prod_{d_1 \times d_2} \Delta_{n_1 \times n_2}(_) \Longrightarrow F(_)$ in several steps:

- (1) define the type E_1 indexed by I_1 as “ $i : I_1 \vdash E_1(i) \stackrel{\text{def}}{=} \sum_{d_1 : D_1} \text{Id}(n_1 \cdot d_1, i)$ ” and similarly for “ $i : I_2 \vdash E_2(i)$ ”.¹
- (2) Thus, $\vdash \rho_{E_1, E_2} : \prod_{d_1} E_1(n_1 \cdot d_1) \square \prod_{d_2} E_2(n_2 \cdot d_2) \rightarrow F(E_1 \square E_2)$.
- (3) Moreover, we have

$$\vdash f_1 \stackrel{\text{def}}{=} \lambda d_1. \langle d_1, \text{refl}(n_1 \cdot d_1) \rangle : \prod_{d_1} E_1(n_1 \cdot d_1)$$

and similarly for “ $\vdash f_2$ ”.

- (4) Given h of type $\prod_{d_1, d_2} R(n_1 \cdot d_1, n_2 \cdot d_2)$, we construct

$$h, i_1, i_2 \vdash \bar{h}(i_1, i_2) \stackrel{\text{def}}{=} \lambda \langle \langle d_1, e_1 \rangle, \langle d_2, e_2 \rangle \rangle. h \langle d_1, d_2 \rangle$$

of type $\prod_{i_1, i_2} E_1(i_1) \times E_2(i_2) \rightarrow R(i_1, i_2)$, i.e., of type $E_1 \square E_2 \rightarrow R$. It works because $h \langle d_1, d_2 \rangle$ is of type $R(n_1 \cdot d_1, n_2 \cdot d_2)$ and since $e_k : \text{Id}(n_k \cdot d_k, i_k)$ ($k = 1, 2$), we can substitute $n_1 \cdot d_1$ and $n_2 \cdot d_2$ for i_1 and i_2 .²

- (5) We can now define Θ_R :

$$h \vdash \Theta_R(h) \stackrel{\text{def}}{=} F_{\bar{h}}(\rho_{E_1, E_2} \langle f_1, f_2 \rangle).$$

This is well typed because $\rho_{E_1, E_2} \langle f_1, f_2 \rangle$ is of type $F(E_1 \square E_2)$ by points 2, 3 and because $F_{\bar{h}}$ is of type $F(E_1 \square E_2) \rightarrow F(R)$.

¹ $\text{Id}(_, _)$ is the extensional identity type. Its introduction rule is “ $a : A \vdash \text{refl}(a) : \text{Id}(a, a)$ ”.

²Strictly speaking, we need to compose with an isomorphism as substitution works only “up-to canonical isomorphisms”, see [Hof95].

We have $\Theta\varepsilon = \rho$ because:

$$\begin{aligned} \Theta_{X \times Y}(\varepsilon_{X,Y} \langle h_1, h_2 \rangle) &= F_{\overline{h_1 \times h_2}}(\rho_{E_1, E_2} \langle f_1, f_2 \rangle) \\ &= \rho_{X,Y} \langle h_1, h_2 \rangle \end{aligned}$$

where the first equality is the definition of Θ and the second follows from naturality of ρ :

$$\begin{array}{ccc} F_1(E_1) \square F_2(E_2) & \xrightarrow{F_1 \overline{h_1} \square F_2 \overline{h_2}} & F_1(X) \square F_2(Y) \\ \rho_{E_1, E_2} \downarrow & & \downarrow \rho_{X, Y} \\ F(E_1 \square E_2) & \xrightarrow{F_{\overline{h_1 \times h_2}}} & F(X \square Y) \end{array} .$$

It works because the action of F_1 on morphisms is composition: $F_1 f(h) = f \circ h$, where f is of type $\prod_i X(i) \rightarrow Y(i)$ and $h : \prod_{d_1} X(n_1 \cdot d_1)$. With that in mind, we find that

$$\begin{aligned} F_1 \overline{h_1} \square F_2 \overline{h_2} \langle f_1, f_2 \rangle &= \langle \overline{h_1} \circ f_1, \overline{h_2} \circ f_2 \rangle \\ &= \langle \lambda d_1. \overline{h_1} \langle d_1, \text{refl}(n_1 \cdot d_1) \rangle, \dots \rangle \\ &= \langle \lambda d_1. h_1(d_1), \dots \rangle \\ &= \langle h_1, h_2 \rangle . \end{aligned}$$

We now need to show that Θ is unique with this property. It follows from the fact that Θ is determined by its values on “rectangles” $X \square Y$:

$$\begin{aligned} \Theta_R(h) &= F_{\overline{h}}(\Theta_{E_1 \square E_2}(f_1 \times f_2)) \\ &= F_{\overline{h}}(\rho_{E_1 \square E_2} \langle f_1, f_2 \rangle) . \end{aligned}$$

The second equality comes from $\Theta\varepsilon = \rho$ and the first one follows from the naturality square

$$\begin{array}{ccc} F_1 \otimes F_2(E_1 \square E_2) & \xrightarrow{(F_1 \otimes F_2) \overline{h}} & F_1 \otimes F_2(R) \\ \Theta_{E_1 \square E_2} \downarrow & & \downarrow \Theta_R \\ F(E_1 \square E_2) & \xrightarrow{F_{\overline{h}}} & F(R) \end{array}$$

where like above, we have $(F_1 \otimes F_2) \overline{h}(f_1 \square f_2) = h$.

This concludes the proof that $F_1 \otimes F_2 = \text{Lan}_{\square}(F_1(-) \square F_2(-))$ and thus the proof that $P_1 \otimes P_2 = \text{Lan}_{\square}(P_1(-) \square P_2(-))$. \square

This operation is a tensor product. This follows for example from the fact that it is functorial in $\text{PDSim}_{\mathbb{C}}$ and that $\text{PFSim}_{\mathbb{C}}$ is equivalent to it (Proposition 2.6), but a direct proof is also possible.

2.4. SMCC Structure. From Propositions 2.2, 2.6 and 2.7, we can deduce that

Proposition 2.9. *The category $\text{PFSim}_{\mathbb{C}}$ with \otimes is symmetric monoidal closed, i.e., there is a functor $_ \multimap _$ from $\text{PFSim}_{\mathbb{C}}^{\text{op}} \times \text{PFSim}_{\mathbb{C}}$ to $\text{PFSim}_{\mathbb{C}}$ with an adjunction*

$$\text{PFSim}_{\mathbb{C}}[P_1 \otimes P_2, P_3] \cong \text{PFSim}_{\mathbb{C}}[P_1, P_2 \multimap P_3],$$

natural in P_1 and P_3 .

The concrete intensional definition of $P_2 \multimap P_3$, either in its type theory version or its diagrammatic version is rather verbose (Definition 3.7 or Lemma 3.8 in [Hyv14]) and won't be needed here. However just as with the tensor, it is possible to define $P_2 \multimap P_3$ without referring to the representing polynomial diagrams. Not surprisingly, it takes the form a right Kan-extension. To simplify the proof, we only state the result for the case $\mathbb{C} = \text{Set}$:

Proposition 2.10. *Given two polynomial endofunctors P_2 and P_3 respectively on Set/I_2 and Set/I_3 , the polynomial endofunctor $P_2 \multimap P_3$ on $\text{Set}/I_2 \times I_3$ is a right Kan-extension along \triangleright : it is universal such that*

$$\begin{array}{ccc} \text{Set}/I_2 \times \text{Set}/I_3 & \xrightarrow{\triangleright} & \text{Set}/I_2 \times I_3 \\ P_2 \times P_3 \downarrow & \swarrow \eta & \downarrow P_2 \multimap P_3 \\ \text{Set}/I_2 \times \text{Set}/I_3 & \xrightarrow{\triangleright} & \text{Set}/I_2 \times I_3 \end{array}$$

where \triangleright is defined as $f \triangleright g \stackrel{\text{def}}{=} \Pi_{f \times 1}(1 \times g)$, or equivalently, $f \triangleright g \stackrel{\text{def}}{=} \Pi_{f \times 1} \Delta_{\tau_2}(g)$. More precisely, we have $P_2 \multimap P_3 = \text{Ran}_{\triangleright}(P_2(-) \triangleright P_3(-))$ in the category of endofunctors with natural transformations.

Corollary 2.11. *We have*

$$P_2 \multimap P_3(r) = \int_{y,z} \text{Set}/I_2 \times I_3[r, y \triangleright z] \pitchfork (P_2(y) \triangleright P_3(z))$$

where \pitchfork is the ‘‘power’’ operation.

Note that in the internal language, $Y \triangleright Z$ is ‘‘ $i_2 : I_2, i_3 : I_3 \vdash Y(i_2) \rightarrow Z(i_3)$ ’’. Before proving Proposition 2.10, we show:

Lemma 2.12. *There is a natural isomorphism*

$$\mathbb{C}/I_3[\langle r \rangle(y), z] \cong \mathbb{C}/I_2 \times I_3[r, y \triangleright z].$$

Proof. In the internal language, those homsets correspond to the types

- $\prod_{i_3} (\sum_{i_2} R(i_2, i_3) \times Y(i_2)) \rightarrow Z(i_3)$
- and $\prod_{i_2, i_3} R(i_2, i_3) \rightarrow (Y(i_2) \rightarrow Z(i_3))$,

which are indeed isomorphic. □

Proof of Proposition 2.10. We will show, using the formulas for Kan extensions and the calculus of ends and coends [Lan98], that the adjoint to $P_1 \otimes _$ (as given by proposition 2.7) is necessarily the above right Kan-extension.

Suppose P_1, P_2 and P_3 are polynomial functors with domains I_1, I_2 and I_3 . Let R be a span between $I_1 \times I_2$ and I_3 . We write R' for the corresponding span between I_1 and $I_2 \times I_3$. Besides the previous lemmas and propositions, we will use:

- if $K_1 \dashv K_2$, there is a natural isomorphism $\text{Nat}(FK_2, G) \cong \text{Nat}(F, GK_1)$ for all functors F and G (note the inversion of left and right);
- the power $X \pitchfork _$ is right-adjoint to the copower $X \odot _$;
- if $K_1 \dashv K_2$, then $[X \odot A, K_2(B)] \cong [X \odot K_1(A), B]$, and similarly for \pitchfork ;
- $\langle \langle R'(x) \rangle \rangle (y) \cong \langle R \rangle (x \square y)$: in the internal language, they are respectively

$$i_3 \vdash \sum_{i_2} \left(\sum_{i_1} R(i_1, i_2, i_3) \times X(i_1) \right) \times Y(i_2)$$

and

$$i_3 \vdash \sum_{i_1, i_2} R(i_1, i_2, i_3) \times (X(i_1) \times Y(i_2))$$

which are naturally isomorphic.

We have:

$$\begin{aligned} & \text{Nat} \left(\langle R' \rangle P_1, \text{Ran}_{\triangleright} (P_2(-) \triangleright P_3(-)) \langle R' \rangle \right) \\ & \cong \int_x \text{Set}_{/I_2 \times I_3} \left[\langle R' \rangle P_1(x), \text{Ran}_{\triangleright} (P_2(-) \triangleright P_3(-)) \langle R' \rangle (x) \right] \\ & \cong \int_x \text{Set}_{/I_2 \times I_3} \left[\langle R' \rangle P_1(x), \int_{y,z} \text{Set}_{/I_2 \times I_3} [\langle R' \rangle (x), y \triangleright z] \pitchfork (P_2(y) \triangleright P_3(z)) \right] \\ & \cong \int_{x,y,z} \text{Set}_{/I_2 \times I_3} \left[\langle R' \rangle P_1(x), \text{Set}_{/I_2 \times I_3} [\langle R' \rangle (x), y \triangleright z] \pitchfork (P_2(y) \triangleright P_3(z)) \right] \\ & \cong \int_{x,y,z} \text{Set}_{/I_2 \times I_3} \left[\text{Set}_{/I_2 \times I_3} [\langle R' \rangle (x), y \triangleright z] \odot \langle R' \rangle P_1(x), P_2(y) \triangleright P_3(z) \right] \\ & \cong \int_{x,y,z} \text{Set}_{/I_3} \left[\text{Set}_{/I_3} [\langle \langle R' \rangle (x) \rangle (y), z] \odot \langle \langle R' \rangle P_1(x) \rangle P_2(y), P_3(z) \right] \\ & \cong \int_{x,y,z} \text{Set}_{/I_3} \left[\text{Set}_{/I_3} [\langle R \rangle (x \square y), z] \odot \langle R \rangle (P_1(x) \square P_2(y)), P_3(z) \right] \\ & \cong \int_{x,y,z} \text{Set}_{/I_1 \times I_2} \left[\text{Set}_{/I_1 \times I_2} [x \square y, [R^\sim](z)] \odot (P_1(x) \square P_2(y)), [R^\sim] P_3(z) \right] \\ & \cong \int_z \text{Set}_{/I_1 \times I_2} \left[\int^{x,y} \text{Set}_{/I_1 \times I_2} [x \square y, [R^\sim](z)] \odot (P_1(x) \square P_2(y)), [R^\sim] P_3(z) \right] \\ & \cong \int_z \text{Set}_{/I_1 \times I_2} \left[P_1 \otimes P_2([R^\sim](z)), [R^\sim] P_3(z) \right] \\ & \cong \text{Nat} \left(P_1 \otimes P_2 [R^\sim], [R^\sim] P_3 \right) \\ & \cong \text{Nat} \left(\langle R \rangle P_1 \otimes P_2 [R^\sim], P_3 \right) \end{aligned}$$

$$\cong \text{Nat}\left(\langle R \rangle P_1 \otimes P_2, P_3 \langle R \rangle\right).$$

Because in Set , all natural transformations are strong, these calculations show that there is a natural isomorphism between $\text{PFSim}_{\text{Set}}[P_1 \otimes P_2, P_3]$ and $\text{PFSim}_{\text{Set}}[P_1, \text{Ran}_{\triangleright}(P_2(-) \triangleright P_3(-))]$. Note that because adjoints are unique up-to isomorphisms, the functor we just defined is necessarily isomorphic to the one defined on polynomial diagrams in [Hyv14]. This implies that $P_2 \multimap P_2$ is indeed well defined and that the Kan extension exists. \square

The previous proof relied on the fact that \mathbb{C} is Set in two ways:

- strong natural transformations and natural transformations are the same thing, so that strong natural transformations can be expressed as an end,
- Set has powers and copowers, so that we can use the end / coend formulas for $_ \multimap _$ and $_ \otimes _$.

Proposition 2.10 holds for arbitrary \mathbb{C} , but the sequence of computations needs to be rewritten to use only the universal properties of left and right Kan extensions, and we need to check that all the natural isomorphisms respect the strength.

2.5. Special Case: Polynomial Presheaves and Day's Convolution. When M is a small monoidal category, presheaves over M have a monoidal structure using Day's convolution product:

$$F \otimes G \stackrel{\text{def}}{=} \int^{a,b \in M} M[_, a \otimes b] \times F(a) \times G(b)$$

whenever $F, G : M^{\text{op}} \rightarrow \text{Set}$. Moreover, this tensor has a right adjoint making presheaves a symmetric monoidal *closed* category. The category $M = \text{Set}^{\text{op}}$ is monoidal but not small. For $F, G : \text{Set} \rightarrow \text{Set}$, the formula for Day's convolution becomes

$$F \otimes G \stackrel{\text{def}}{=} \int^{a,b \in \text{Set}} \text{Set}[a \times b, _] \times F(a) \times G(b).$$

This coend needs not exist as it is indexed by a large category. However, when F and G are polynomial, this is just a special case of Corollary 2.8.

Definition 2.13. Call a presheaf $P : \text{Set} \rightarrow \text{Set}$ polynomial if it is equivalent to

$$X \mapsto \sum_{a \in A} X^{D(a)}$$

for some set A and family $D : A \rightarrow \text{Set}$.

Corollary 2.8 implies that polynomial presheaves are closed under Day's convolution and we have the explicit formula:

$$\left(\sum_{a_1 \in A_1} X^{D_1(a_1)} \right) \otimes \left(\sum_{a_2 \in A_2} X^{D_2(a_2)} \right) \cong \sum_{(a_1, a_2) \in A_1 \times A_2} X^{D_1(a_1) \times D_2(a_2)}. \quad (2.3)$$

Moreover, the right-adjoint is also polynomial:

Proposition 2.14. *The category of polynomial endofunctors on Set with Day's convolution product is symmetric monoidal closed.*

Proof. The category of polynomial endofunctors on \mathbf{Set} with natural transformations between them is a (non full) subcategory of $\mathbf{PFSim}_{\mathbf{Set}}$. It is thus closed under $_ \otimes _$ and $_ \dashv _$.

To show that \otimes and \dashv are still adjoint in this category, we can rewrite the same proof as for Proposition 2.10 and replace R everywhere by the trivial span $\langle 1, 1 \rangle$. The proof carries through. \square

There is an explicit formula for the right-adjoint $_ \dashv _$, but it is much less elegant than the formula for the tensor:

$$\left(\sum_{a_1 \in A_1} X^{D_1(a_1)} \right) \dashv \left(\sum_{a_2 \in A_2} X^{D_2(a_2)} \right) \cong \sum_{c \in C} X^{E(c)} \quad (2.4)$$

where

$$C \stackrel{\text{def}}{=} \sum_{f \in A_1 \rightarrow A_2} \prod_{a_1 \in A_1} D_1(a_1)^{D_2(f(a_1))} \quad \text{and} \quad E((f, \phi)) \stackrel{\text{def}}{=} \sum_{a_1 \in A_1} D_2(f(a_1)) .$$

(See [Hyv14].)

3. ADDITIVE AND EXPONENTIAL STRUCTURE

3.1. Additive Structure. In [Hyv14], it was shown that the category of polynomial diagrams with simulations also has a cartesian / cocartesian structure whenever \mathbb{C} has a well behaved coproduct. The coproduct of two diagrams, which is also their product is defined as:

$$P_1 \oplus P_2 \stackrel{\text{def}}{=} I_1 + I_2 \xleftarrow{n_1+n_2} D_1 + D_2 \xrightarrow{d_1+d_2} A_1 + A_2 \xrightarrow{a_1+a_2} J_1 + J_2 \quad .$$

A category \mathbb{C} is *extensive* if the canonical functor $\mathbb{C}/I_1 \times \mathbb{C}/I_2 \rightarrow \mathbb{C}/I_1+I_2$ sending (f_1, f_2) to $f_1 + f_2$ is an equivalence of category. It implies in particular the following:

Lemma 3.1. *If \mathbb{C} is locally cartesian closed and extensive, we have*

- $\Sigma_{f+g}(k+l) \cong \Sigma_f(k) + \Sigma_g(l)$,
- $\Delta_{f+g}(k+l) \cong \Delta_f(k) + \Delta_g(l)$,
- $\Pi_{f+g}(k+l) \cong \Pi_f(k) + \Pi_g(l)$

whenever the expressions make sense.

With that in mind, we have directly that $\llbracket P_1 \oplus P_2 \rrbracket(x+y) \cong \llbracket P_1 \rrbracket(x) + \llbracket P_2 \rrbracket(y)$ whenever $x \in \mathbb{C}/I_1$ and $y \in \mathbb{C}/I_2$. We can thus express the additive structure on polynomial functors without referring to the underlying polynomials. We have

Lemma 3.2. *If \mathbb{C} is extensive with an initial object $\mathbf{0}$, then:*

- (1) *the unique functor from $\mathbb{C}/\mathbf{0}$ to itself is a zero object in $\mathbf{FSim}_{\mathbb{C}}$,*
- (2) *if we define $F_1 \oplus F_2$ on $\mathbb{C}/I_1+I_2 \cong \mathbb{C}/I_1 \times \mathbb{C}/I_2$ with*

$$F_1 \oplus F_2(x+y) \stackrel{\text{def}}{=} F_1(x) + F_2(y) ,$$

then $_ \oplus _$ is a product as well as a coproduct in the category $\mathbf{FSim}_{\mathbb{C}}$.

- (3) *$\mathbf{0}$ and $_ \oplus _$ are a zero object and a product/coproduct in $\mathbf{PFSim}_{\mathbb{C}}$ as well.*

Proof. The first point is direct. The second point boils down to the following: because \mathbb{C} is extensive, any span $I_1 + I_2 \leftarrow R \rightarrow J$ is isomorphic to a span $I_1 + I_2 \leftarrow R_1 + R_2 \rightarrow J$ where the left leg is $r_1 + r_2$ with $r_k : I_k \rightarrow R_k$ and the right leg is $[s_1, s_2]$, with $s_k : R_k \rightarrow J$. Let's write R_k for the obvious span $I_k \leftarrow R_k \rightarrow J$: its legs are r_k and s_k . Extensivity of \mathbb{C} implies that:

- $[[\langle R \rangle]](x + y) = [[\langle R_1 \rangle]](x), [[\langle R_2 \rangle]](y) : \mathbb{C}/J$ for any $x + y : \mathbb{C}/_{I_1+I_2}$;
- $[[[R^\sim]]](z) = [[[R_1^\sim]]](z) + [[[R_2^\sim]]](z) : \mathbb{C}/_{I_1+I_2}$ for any $z : \mathbb{C}/J$. (Recall that $[[[R^\sim]]]$ is the right adjoint of $[[\langle R \rangle]]$ as per Lemma 1.7.)

We have:

$$\begin{aligned}
& \text{Nat}\left(\langle R \rangle F_1 \oplus F_2, G \langle R \rangle\right) \\
& \cong \text{Nat}\left(\langle R \rangle F_1 \oplus F_2 [R^\sim], G\right) \\
& \cong \int_{z \in \mathbb{C}/J} \mathbb{C}/J \left[\langle R \rangle F_1 \oplus F_2 [R^\sim](z), G(z) \right] \\
& \cong \int_z \mathbb{C}/J \left[\langle R \rangle F_1 \oplus F_2 \left([R_1^\sim](z) + [R_2^\sim](z) \right), G(z) \right] \\
& \cong \int_z \mathbb{C}/J \left[\langle R \rangle \left(F_1 [R_1^\sim](z) + F_2 [R_2^\sim](z) \right), G(z) \right] \\
& \cong \int_z \mathbb{C}/J \left[\langle R_1 \rangle F_1 [R_1^\sim](z) + \langle R_2 \rangle F_2 [R_2^\sim](z), G(z) \right] \\
& \cong \int_z \mathbb{C}/J \left[\langle R_1 \rangle F_1 [R_1^\sim](z), G(z) \right] \times \mathbb{C}/J \left[\langle R_2 \rangle F_2 [R_2^\sim](z), G(z) \right] \\
& \cong \int_z \mathbb{C}/J \left[\langle R_1 \rangle F_1 [R_1^\sim](z), G(z) \right] \times \int_z \mathbb{C}/J \left[\langle R_2 \rangle F_2 [R_2^\sim](z), G(z) \right] \\
& \cong \text{Nat}\left(\langle R_1 \rangle F_1 [R_1^\sim], G\right) \times \text{Nat}\left(\langle R_2 \rangle F_2 [R_2^\sim], G\right) \\
& \cong \text{Nat}\left(\langle R_1 \rangle F_1, G \langle R_1 \rangle\right) \times \text{Nat}\left(\langle R_2 \rangle F_2, G \langle R_2 \rangle\right).
\end{aligned}$$

This shows that \oplus is indeed the coproduct in $\text{FSim}_{\mathbb{C}}$. Note that this proof doesn't rely on the functors F_1 and F_2 being polynomial. The proof that it is also a product is similar.

To get the last point, i.e., that $_ \oplus _$ is also a coproduct in $\text{PFSim}_{\mathbb{C}}$, one needs to show that the natural isomorphisms given preserves the strengths of natural transformations. This is left as an exercise... Another way to prove the last point is simply to use Lemma 3.1 and the fact that \oplus is the product and coproduct in $\text{PDSim}_{\mathbb{C}}$ [Hyv14]. \square

3.2. Exponential Structure. As hinted in [Hyv14], the category of $\text{PDSim}_{\text{Set}}$ has free commutative \otimes -comonoids. For a set I , we write $\mathcal{M}_f(I)$ for the collection of finite multisets of elements of I . The free commutative \otimes -comonoid for $I \leftarrow D \rightarrow A \rightarrow I$ is given by

$$\mathcal{M}_f(I) \xleftarrow{c_I \circ n^*} D^* \xrightarrow{d^*} A^* \xrightarrow{c_I \circ a^*} \mathcal{M}_f(I) \tag{3.1}$$

where

- $_*$: $\mathbf{Set} \rightarrow \mathbf{Set}$ is the “list functor” sending a set X to the collection of finite sequences of elements in X ,
- $c_I : I^* \rightarrow \mathcal{M}_f(I)$ sends a sequence to its equivalence class under permutations (multiset).

Conjecture 3.3. In $\mathbf{PFSim}_{\mathbf{Set}}$, the free commutative \otimes -comonoid over F is given by

$$!F \stackrel{\text{def}}{=} \Sigma_{c_I} \circ F^* \circ \Delta_{c_I} .$$

This conjecture is a strengthening of the following lemma:

Lemma 3.4. *If we write $\mathbf{PFSim}_{\mathbf{Set}\sim}$ for the category of polynomial functors and simulations, where two simulations (R, ρ) and (R', ρ') are identified when $R \cong R'$, then:*

- $\mathbf{PFSim}_{\mathbf{Set}\sim}$ with \otimes and \multimap is symmetric monoidal closed,
- $\mathbf{PFSim}_{\mathbf{Set}\sim}$ with $\mathbf{0}$ and \oplus is cartesian and cocartesian,
- $\mathbf{PFSim}_{\mathbf{Set}\sim}$ has free commutative \otimes -comonoids given by

$$!F \stackrel{\text{def}}{=} \Sigma_{c_I} \circ F^* \circ \Delta_{c_I} .$$

Proof. The first two points follow from earlier results in the paper, and the third one is a consequence of the fact that (3.1) is the free commutative \otimes -comonoid in $\mathbf{PDSim}_{\mathbf{Set}\sim}$ [Hyv14]. It gives the extensional definition of the lemma since it implies that $[[!P]] = \Sigma_{c_I} \circ [[P]]^* \circ \Delta_{c_I}$. \square

It doesn't look too difficult to extend the proof that (3.1) gives the free commutative \otimes -comonoid in $\mathbf{PDSim}_{\mathbf{Set}\sim}$ (Proposition 3.2 in [Hyv14]) to the whole of $\mathbf{PDSim}_{\mathbf{Set}}$. A complete and concise (or at least readable) proof of this fact would be most welcome and would prove the conjecture...

3.3. Failure of Classical (Linear) Logic. It is natural to ask if the resulting model for intuitionistic linear logic can be extended to a model for classical linear logic, i.e., if the monoidal closed structure of the category $\mathbf{PFSim}_{\mathbf{Set}}$ can be extended to a $*$ -autonomous structure. The answer is, perhaps unsurprisingly, no.

The full proof isn't very enlightening but let's look at what happens with the “natural” choice of $\perp \stackrel{\text{def}}{=} 1 \leftarrow 1 \rightarrow 1 \rightarrow 1$ as a potential dualizing object. Take arbitrary objects A and B in \mathbb{C} and define the polynomial functor $P_{A,B}(X) = A \times X^B$. As in the case of presheaves on sets (page 17), there is a simple explicit formula for $P_{A,B}^\perp = P_{A,B} \multimap \perp$: we have $P_{A,B}^\perp(X) = B^A \times X^A$ and

$$P_{A,B}^{\perp\perp}(X) = A^{B^A} X^{B^A} .$$

Asking that the canonical simulation from $P_{A,B}$ to $P_{A,B}^{\perp\perp}$ is an isomorphism would imply (by a variant of Corollary 1.5) that the canonical map from A to A^{B^A} in \mathbb{C} is an isomorphism. This is not possible in general:

Lemma 3.5. *Any cartesian closed category \mathbb{C} in which the canonical natural transformation from A to A^{B^A} is an isomorphism is posetal.*

If \mathbb{C} is also cocartesian, then it becomes a (possibly large) “pre boolean algebra”.

Proof. Because \mathbb{C} is cartesian closed, we may use simply typed λ -calculus as its internal language. We'll write A^B as $B \Rightarrow A$ as is natural in type theory. Taking both A and B to be $C \times C$ above, we have the following isomorphism:

$$\begin{aligned} \mathbb{C}[1, C \times C] &\cong \mathbb{C}[1, (C \times C \Rightarrow C \times C) \Rightarrow C \times C] \\ &\cong \mathbb{C}[C \times C \Rightarrow C \times C, C \times C] \end{aligned}$$

where each pair $\langle x, y \rangle$ in $\mathbb{C}[1, C \times C]$ is sent to $\lambda f. \langle x, y \rangle$ in $\mathbb{C}[C \times C \Rightarrow C \times C, C \times C]$.

Take c_1, c_2 in $\mathbb{C}[1, C]$, we have $\varphi = \lambda f. f \langle c_1, c_2 \rangle$ in $\mathbb{C}[C \times C \Rightarrow C \times C, C \times C]$ which satisfies $\varphi \text{ id} = \langle c_1, c_2 \rangle$ and $\varphi \text{ tw} = \langle c_2, c_1 \rangle$, where tw in $\mathbb{C}[C \times C, C \times C]$ exchanges the left and right components of a pair. However, Because of the isomorphism above, φ must be of the form $\lambda f. \langle x, y \rangle$ for some x and y in $\mathbb{C}[1, C]$. This implies that $\varphi \text{ id} = \varphi \text{ tw}$, and thus, that $c_1 = c_2$. Because \mathbb{C} is cartesian closed, any f_1, f_2 in $\mathbb{C}[A, B]$ correspond precisely to constants $\lceil f_1 \rceil$ and $\lceil f_2 \rceil$ in $\mathbb{C}[1, A \Rightarrow B]$ and are thus equal. The category \mathbb{C} is thus posetal.

Because \mathbb{C} is cartesian closed, it is thus a (possibly large) Heyting semi-lattice; and if \mathbb{C} is also cocartesian, it becomes a Heyting algebra. Now, the existence of a transformation from $(A \Rightarrow B) \Rightarrow A$ to A amounts to saying the law of Peirce is satisfied. This makes the Heyting algebra boolean... □

Note however that the model of predicate transformers from [Hyv04] can be seen as a “proof irrelevant” variant of $\text{PFSim}_{\text{Set}}$:

- we collapse the categories Set/I into preorders, making each Set/I equivalent to the algebra of subsets of I ,
- we collapse the categories of spans into preorders, making each $\text{Span}[I, J]$ equivalent to the algebra of relations between I and J ,
- we identify simulations (R, α) and (R, β) .

This gives a non-trivial $*$ -autonomous category:

- polynomial functors on Set/I become monotonic transformations on $\mathcal{P}(I)$, all of which are in fact “polynomial”,
- simulations become relations satisfying a closure property,
- duality gives $P^\perp(x) = \overline{P(\bar{x})}$ whenever $P : \mathcal{P}(I) \rightarrow \mathcal{P}(I)$, where \bar{y} is the complement of y with respect to I .
- the dualizing object is $\mathbf{1}$, the unit of the tensor, i.e., $\mathbf{1} = \perp$.
- However, the category is not compact closed as $_ \otimes _$ is different from its dual.

3.4. Future Work. The first question that comes to mind is what happens if we consider analytic functors on Set , i.e., those of the form

$$X \mapsto \sum_{a \in A} X^{D(a)} /_{G(a)}$$

where each $G(a)$ is a subgroup of the automorphisms of $D(a)$, acting in an obvious way on $X^{D(a)}$. Formula 2.3 from Section 2.5 has a natural generalization to this context. Does it work as an intensional formula for Day’s convolution? What about the analogous to formula 2.4?

4. THANKS

The author would like to thank both Neil Ghani and Paul-André Mellies for helpful discussions on early versions of this work.

REFERENCES

- [AAG05] Michael Abott, Thorsten Altenkirch, and Neil Ghani, *Containers - constructing strictly positive types*, Theoretical Computer Science **342** (2005), 3–27.
- [GK09] Nicola Gambino and Joachim Kock, *Polynomial functors and polynomial monads*, To appear in Mathematical Proceedings of the Cambridge Philosophical Society, [arXiv:0906.4931v2](https://arxiv.org/abs/0906.4931v2), 2009.
- [HH06] Peter Hancock and Pierre Hyvernat, *Programming interfaces and basic topology*, Annals of Pure and Applied Logic **137** (2006), no. 1-3, 189–239. MR MR2182103
- [Hof95] Martin Hofmann, *On the interpretation of type theory in locally cartesian closed categories*, CSL '94: Selected Papers from the 8th International Workshop on Computer Science Logic (London, UK), Springer-Verlag, 1995, pp. 427–441.
- [Hyv04] Pierre Hyvernat, *Predicate transformers and linear logic: yet another denotational model*, 18th International Workshop CSL 2004 (Jerzy Marcinkowski and Andrzej Tarlecki, eds.), LNCS, vol. 3210, Springer-Verlag, September 2004, pp. 115–129.
- [Hyv14] Pierre Hyvernat, *A Linear Category of Polynomial Diagrams*, Mathematical Structures in Computer Science **24** (2014), no. 1.
- [Koc09] Joachim Kock, *Notes on polynomial functors*, preliminary draft, 2009.
- [Lan98] Saunders Mac Lane, *Categories for the working mathematician*, second ed., Graduate Texts in Mathematics, vol. 5, Springer-Verlag, New York, 1998. MR MR1712872 (2001j:18001)
- [MA09] Peter Morris and Thorsten Altenkirch, *Indexed containers*, Twenty-Fourth IEEE Symposium in Logic in Computer Science (LICS 2009), 2009.
- [ML84] Per Martin-Löf, *Intuitionistic type theory*, Bibliopolis, Naples, 1984, Notes by Giovanni Sambin. MR 86j:03005
- [PS89] Kent Petersson and Dan Synek, *A set constructor for inductive sets in Martin-Löf's type theory*, Proceedings of the 1989 Conference on Category Theory and Computer Science, Manchester, UK, LNCS, vol. 389, Springer Verlag, 1989.
- [See84] Robert A. G. Seely, *Locally cartesian closed categories and type theory*, Mathematical Proceedings of the Cambridge Philosophical Society **95** (1984), no. 1, 33–48. MR MR727078 (86b:18008)

APPENDIX A. LOCALLY CARTESIAN CLOSED CATEGORIES

For a category \mathbb{C} with finite limits, we write “ $\mathbf{1}$ ” for its terminal object and “ $A \times B$ ” for the cartesian product of A and B . The “pairing” of $f : C \rightarrow A$ and $g : C \rightarrow B$ is written $\langle f, g \rangle : C \rightarrow A \times B$.

If $f : A \rightarrow B$ is a morphism, it induces a pullback functor Δ_f from slices over B to slices over A . This functor has a left adjoint Σ_f which is “pre-composition by f ”. When all the Δ_f s also have a right adjoint, we say that \mathbb{C} is *locally cartesian closed*. The right adjoint is written Π_f . We thus have

$$\Sigma_f \dashv \Delta_f \dashv \Pi_f .$$

Besides the isomorphisms coming from the adjunctions, slices enjoy two fundamental properties:

- *the Beck-Chevalley isomorphisms:*

$$\Pi_g \Delta_l \cong \Delta_k \Pi_f \quad \text{and} \quad \Sigma_g \Delta_l \cong \Delta_k \Sigma_f$$

whenever

$$\begin{array}{ccc} \cdot & \xrightarrow{g} & \cdot \\ \downarrow l & \lrcorner & \downarrow k \\ \cdot & \xrightarrow{f} & \cdot \end{array}$$

is a pullback,

- *distributivity:* when $b : C \rightarrow B$ and $a : B \rightarrow A$, we have a commuting diagram

$$\begin{array}{ccc} \cdot & \xrightarrow{a'} & \cdot \\ \swarrow \epsilon & \lrcorner & \downarrow u \stackrel{\text{def}}{=} \Pi_a(b) \\ C & \xrightarrow{u'} & B \\ \searrow b & & \xrightarrow{a} A \end{array} \quad (\text{A.1})$$

where ϵ is the co-unit of $\Delta_a \dashv \Pi_a$. For such a diagram, we have

$$\Pi_a \Sigma_b \cong \Sigma_u \Pi_{a'} \Delta_\epsilon .$$

APPENDIX B. DEPENDENT TYPE THEORY

In [See84], Seely showed how an extensional version of Martin L of’s theory of dependent types [ML84] could be regarded as the internal language for locally cartesian closed categories. A little later, Hofmann showed in [Hof95] that Seely’s interpretation works only “up-to canonical isomorphisms” and proposed a solution.

A type A in context Γ , written “ $\Gamma \vdash A$ type” is interpreted as a morphism $a : \Gamma_A \rightarrow \Gamma$, that is as an object in the slice over (the interpretation of) Γ . Then, a term of type A in context Γ , written “ $\Gamma \vdash t : A$ ” is interpreted as a morphism $u : \Gamma \rightarrow \Gamma_A$ such that $au = 1$, i.e., a section of (the interpretation of) its type. When A is a type in context Γ , we usually write $A(\gamma)$ to emphasize the dependency on the context and we silently omit irrelevant

parameters. If we write $\llbracket \Gamma \vdash A \rrbracket = a$ to mean that the interpretation of type $\Gamma \vdash A$ is a , the main points of the Seely semantics are:

$$\frac{\llbracket \Gamma \vdash A \rrbracket = a \quad \llbracket \Gamma, x : A \vdash B(x) \rrbracket = b}{\llbracket \Gamma \vdash \prod_{x:A} B(x) \rrbracket = \Pi_a(b)} \quad \text{product ,}$$

$$\frac{\llbracket \Gamma \vdash A \rrbracket = a \quad \llbracket \Gamma, x : A \vdash B(x) \rrbracket = b}{\llbracket \Gamma \vdash \sum_{x:A} B(x) \rrbracket = \Sigma_a(b)} \quad \text{sum ,}$$

$$\frac{\llbracket \Gamma \vdash \vec{u} : \Delta \rrbracket = f \quad \llbracket \Delta \vdash A(\vec{x}) \rrbracket = u}{\llbracket \Gamma \vdash A(\vec{u}) \rrbracket = \Delta_f(u)} \quad \text{substitution .}$$

Of particular importance is the distributivity condition (A.1) whose type theoretic version is an intensional version of the axiom of choice:

$$\Gamma \vdash \prod_{x:A} \sum_{y:B(x)} U(x, y) \quad \cong \quad \Gamma \vdash \sum_{f:\prod_{x:A} B(x)} \prod_{x:A} U(x, f(x)) . \quad (\text{B.1})$$

Extensional type theory has a special type for equality “proofs”. Its formation and introduction rules are as follows:

$$\frac{\Gamma \vdash u_1 : A \quad \Gamma \vdash u_2 : A}{\Gamma \vdash \text{Id}_A(u_1, u_2)} \quad \frac{\Gamma \vdash u : A}{\Gamma \vdash \text{refl}_A(u) : \text{Id}_A(u, u)} .$$

This type for equality is “extensional” because having an inhabitant of $\text{Id}_X(u, v)$ implies that u and v are definitionally (extensionally) equal. This is reflected in its interpretation:

$$\frac{\llbracket \Gamma \vdash u_1 : A \rrbracket = f_1 \quad \llbracket \Gamma \vdash u_2 : A \rrbracket = f_2}{\llbracket \Gamma \vdash \text{Id}_X(x_1, x_2) \rrbracket = \text{eq}(f_1, f_2)} \quad \text{identity type}$$

where $\text{eq}(f_1, f_2)$ is the equalizer of f_1 and f_2 . (This is indeed a slice over the interpretation of Γ .)