A LINEAR CATEGORY OF POLYNOMIAL DIAGRAMS

PIERRE HYVERNAT

Abstract. We present a categorical model for intuitionistic linear logic where objects are polynomial diagrams and morphisms are *simulation diagrams*. The multiplicative structure (tensor product and its adjoint) can be defined in any locally cartesian closed category, whereas the additive (product and coproduct) and exponential ($\otimes$-comonoid comonad) structures require additional properties and are only developed in the category Set, where the objects and morphisms have natural interpretations in terms of games, simulation and strategies.

Introduction

Categories of games abound in the literature of denotational semantics of linear logic. We present a category based on a notion of game that differs from traditional games semantics. Objects are a kind of two players game and following a well established tradition [Joy77], morphisms from G_1 to G_2 amount to strategies (for the first player) in a game called $G_1 \multimap G_2$. Composition of strategies is a "relational composition", or more precisely, "span composition". In particular, interaction is irrelevant to the definition of composition. The theory is developed in the abstract setting of locally cartesian closed categories and polynomial diagrams [GK09, Koc09].

The paper is organized as follows: after some preliminaries (section 1), we construct a symmetric monoidal closed category around polynomial diagrams over any locally cartesian closed category (section 2). We then restrict to the case where the base category is the category of sets and functions and extend the SMCC structure to a denotational model for full intuitionistic linear logic by adding a biproduct (cartesian and cocartesian structure) and constructing an exponential comonad for the free commutative $\otimes$-comonoid (section 3). The proof that the exponential structure is indeed the free commutative $\otimes$-comonoid involves a lot of tedious checking and, for simplicity's sake, is only spelt out in a simpler category.

Related Works. Part of this work is implicitly present in [Hyv05, HH06] where polynomial endofunctors are called "interaction systems" and everything was done with dependant type theory. The focus was on representing formal topological spaces with dependent types. Categorically speaking, it amount to the following: it is possible to show that the free monad construction on polynomial endofunctors described in the second part of [GK09] gives a monad on the category of polynomial endofunctors and simulations (Section 2.1). The category introduced in [HH06] is "simply" the Kleisli category of the this monad.

The same kind of polynomial functors are also used by Altenkirch and Morris (together with Ghani, Hancock and McBride) in [MA09] under the name "indexed containers" in order to give a semantics to a large family of indexed, strictly positive datatypes. The unindexed version was developed earlier in [AAG05].

Date: September 2012.
Key words and phrases. polynomial functors; linear logic.

Games. The games we consider are non-initialized, state-based, 2-players, alternating games. More precisely, a game is given by the following data:

- a set I of *states*,
- for each state $i \in I$, a set of *moves* $A(i)$ for Alfred,
- for each move $a \in A(i)$, another set of *counter moves* $D(a)$ for Dominic,
- a function n going from counter moves to states, giving the new state after each possible choice of counter move from Dominic. We usually write $i[a/d]$ instead of $n(d)$ whenever $a \in A(i)$ and $d \in D(a)$.

As is customary in categories, we represent a family of sets indexed by I by an object of the slice category $\mathsf{Set}/_I$. A game is thus simply given by a diagram of the form

$$I \xleftarrow{\ n\ } D \xrightarrow{\ d\ } A \xrightarrow{\ a\ } I \quad .$$

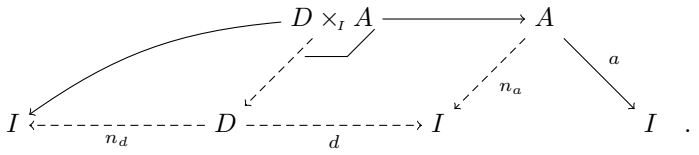
Other names for the players would be "Player" and "Opponent" (games semantics), "Angel" and "Demon" (process calculus), "Alice" and "Bob" (cryptography) etc.

Those games are slightly asymmetrical in that there is no actual state between A-moves and D-moves. In particular, it is not obvious what form the familiar A/D duality should take. Many of the usual board games such as chess or go are more symmetric: given a state, both players could make a move. Such games are more appropriately represented by two spans over the same set:

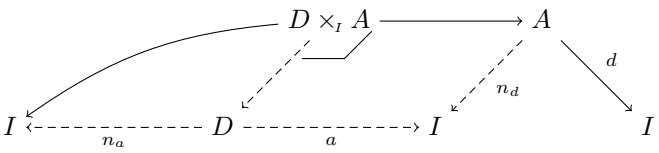
$$I \xleftarrow{\ d\ } D \xrightarrow{\ n_d\ } I \qquad\qquad I \xleftarrow{\ a\ } A \xrightarrow{\ n_a\ } I \quad .$$

- I represents the set of states,
- for each state $i \in I$, the fiber $A(i)$ gives the available A-moves and the function n_A gives the new state after such a move,
- for each state $i \in I$, the fiber $D(i)$ gives the available D-moves and the function n_D gives the new state after such a move.

We get an asymmetric alternating version as above with the plain arrows from the following construction:



This kind of games could prove an interesting starting point for representing Conway games [Joy77]. For those games, the opposite (or negation for games semantics) amounts to interchanging the players by considering



which is *not* the dual of section 2.3.

Another possibility suggested by Martin Hyland would be to consider games where the two players play simultaneously. Those games are represented by a slice $a : A \to I$ for A-moves and a slice $d : D \to I$ for D-moves, together with a function $n : A \times_I D \to I$ to get the next state. As far as our games are concerned,

it amounts to considering diagrams of the form

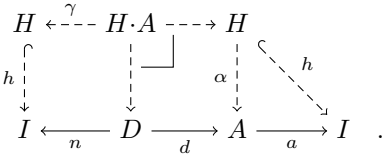
$$I \xleftarrow{\ n\ } A \times_I D \xrightarrow{\ \Delta_d(a)\ } D \xrightarrow{\ a\ } I \quad .$$

In such games, the sets of counter moves $D(a)$ depend only on $i \in I$ and not on the actual $a \in A(i)$. Here again, there is a natural notion of duality, different from the one from section 2.3.

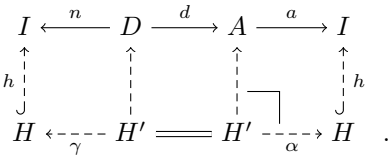
Strategies and Simulations. For a game $I \leftarrow D \rightarrow A \rightarrow I$ as above, a "non-losing strategy for Alfred" consists of:

- a subset of "good" states $H \subset I$,
- a function α choosing an A-move for each $i \in H$,
- such that whenever $i \in H$ and $d \in D\big(\alpha(i)\big)$, we have $i[\alpha(i)/d] \in H$.

This means that, as long as the games start in H, Alfred always has a move to play. Each game will either go on infinitely or stop when Dominic has no counter-move available. Categorically speaking, such a strategy is described by a diagram



Dually, a "non-losing strategy for Dominic" is given by a diagram



The *simulations* we will consider generalize both kind of strategies and satisfy a property similar to what appears in the theory of labeled transition systems, but with an additional layer of quantifiers to account for the counter moves. To make a relation $R \subseteq I_1 \times I_2$ into a simulation between games $I_1 \leftarrow D_1 \rightarrow A_1 \rightarrow I_1$ and $I_2 \leftarrow D_2 \rightarrow A_2 \rightarrow I_2$ as above, we need two functions α and β satisfying:

- whenever $(i_1, i_2) \in R$ and $a_1 \in A_1(i_1)$, there is a move $a_2 \overset{\mathsf{def}}{=} \alpha(a_1) \in A_2(i_2)$ which simulates a_1 in the following sense;
- whenever $d_2 \in D_2(a_2)$ is a response to a_2, there is a response $d_1 \overset{\mathsf{def}}{=} \beta(d_2) \in D_1(a_1)$, such that $\big(i_1[a_1/d_1]\,,\, i_2[a_2/d_2]\big) \in R$.

Strategies for Alfred or Dominic are obtained by instantiating $I_1 \leftarrow D_1 \rightarrow A_1 \rightarrow I_1$ or $I_2 \leftarrow D_2 \rightarrow A_2 \rightarrow I_2$ to the trivial game $\mathbf{1} \leftarrow \mathbf{1} \rightarrow \mathbf{1} \rightarrow \mathbf{1}$, where $\mathbf{1} = \{\star\}$ is the terminal object of Set. The symmetric monoidal closed structure will in particular imply that a simulation from G_1 to G_2 is simply a strategy (for Alfred) for a game called $G_1 \multimap G_2$.

Note that the actual definition of simulation will be slightly more general in that it considers arbitrary spans instead of relations (monic spans). For strategies, considering some $H \rightarrow I$ instead of a subset makes it possible for the function α to choose move depending on more than just a state.

1. Preliminaries, Polynomials and Polynomial Functors

1.1. **Locally Cartesian Closed Categories.** Some basic knowledge about locally cartesian closed categories and their internal language (extensional dependent type theory) is assumed throughout the paper. Here is a review of the notions (and

notations) needed in the rest of the paper. 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 $f : C \to A$ and $g : C \to B$ is written $\langle f, g\rangle : C \to A \times B$.

If $f : A \to B$ 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 simply "pre-composition by f". When all the Δ_fs also have a right adjoint, we say that $\mathbb{C}$ is *locally cartesian closed*. The right adjoint is written Π_f. In all the sequel, $\mathbb{C}$ stands for a category which is (at least) locally cartesian closed. We thus have

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

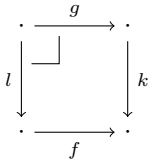
Additional requirements will be explicitly stated.

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

- *the Beck-Chevalley isomorphisms:*

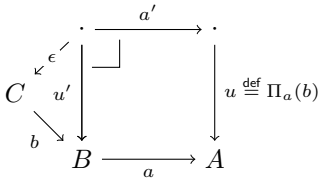
  $$\Pi_g \, \Delta_l \quad \cong \quad \Delta_k \, \Pi_f \qquad \text{and} \qquad \Sigma_g \, \Delta_l \quad \cong \quad \Delta_k \, \Sigma_f$$

  whenever

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

(1)



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

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

Moreover, any slice category $\mathbb{C}/_I$ is canonically enriched over $\mathbb{C}$ by putting

$$\mathrm{Hom}(x, y) \quad \stackrel{\mathsf{def}}{=} \quad \Pi_I \Pi_x \Delta_x(y)$$

whenever $x, y \in \mathbb{C}/_I$. (Here, I also stands for the unique map from I to $\mathbf{1}$.) Any $\mathbb{C}/_I$ is also canonically tensored over $\mathbb{C}$ by using the left adjoint of $\mathrm{Hom}(x, \_)$:

$$A \odot x \quad \stackrel{\mathsf{def}}{=} \quad \Sigma_x \Delta_x \Delta_I(A)$$

for any $x \in \mathbb{C}/_I$ and object A.

1.2. **Dependent Type Theory.** In [See84], Seely showed how an extensional version of Martin Löf'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. Some of the proofs in this paper rely on the use of this internal language and we use Seely's original interpretation. Strictly speaking, some of those morphisms constructed with type theory should be composed with canonical "substitution" isomorphisms.

A type A in context Γ, written $\Gamma \vdash A$ is interpreted as a morphism $a : \Gamma_A \to \Gamma$, 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 $u : \Gamma \to \Gamma_A$ 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 $[\![\Gamma \vdash A]\!] = a$ to mean that the interpretation of type $\Gamma \vdash A$ is a, the main points of the Seely semantics are:

$$\frac{[\![\Gamma \vdash A]\!] = a \qquad [\![\Gamma, x : A \vdash B(x)]\!] = b}{[\![\Gamma \vdash \textstyle\prod_{x:A} B(x)]\!] = \Pi_a(b)} \text{ product },$$

$$\frac{[\![\Gamma \vdash A]\!] = a \qquad [\![\Gamma, x : A \vdash B(x)]\!] = b}{[\![\Gamma \vdash \textstyle\sum_{x:A} B(x)]\!] = \Sigma_a(b)} \text{ sum },$$

$$\frac{[\![\Gamma \vdash \vec{u} : \Delta]\!] = f \qquad [\![\Delta \vdash A(\vec{x})]\!] = u}{[\![\Gamma \vdash A(\vec{u})]\!] = \Delta_f(u)} \text{ substitution }.$$

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

$$(2) \qquad \Gamma \vdash \prod_{x:A} \sum_{y:B(x)} U(x,y) \qquad \cong \qquad \Gamma \vdash \sum_{f:\prod_{x:A} B(x)} \prod_{x:A} U\big(x, f(x)\big) \ .$$

1.3. **Polynomials and Polynomial Functors.** We now recall some definitions and results from [GK09] and refer to the original article for historical notes, details about proofs and additional comments.

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

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

We write $\mathsf{Poly}_{\mathbb{C}}[I, J]$ for the collection of such polynomials from I to J.

Definition 1.2. *For each $P \in \mathsf{Poly}_{\mathbb{C}}[I, J]$ as in Definition 1.1, there is an associated functor from $\mathbb{C}/_I$ to $\mathbb{C}/_J$ called the* extension *of P. It is also denoted by P and is defined as the following composition*

$$P \quad = \quad \mathbb{C}/_I \xrightarrow{\ \Delta_n\ } \mathbb{C}/_D \xrightarrow{\ \Pi_d\ } \mathbb{C}/_A \xrightarrow{\ \Sigma_a\ } \mathbb{C}/_J \quad .$$

Any functor isomorphic to the extension of a polynomial is called a polynomial functor. *We write $\mathsf{PolyFun}_{\mathbb{C}}[I, J]$ for the collection of polynomial functors.*

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

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

and it can be shown that polynomial functors compose, see [GK09] for example. We thus obtain a category $\mathsf{PolyFun}_{\mathbb{C}}$ where objects are slice categories and morphisms are polynomial functors. We also obtain a *bicategory* $\mathsf{Poly}_{\mathbb{C}}$ of polynomials: composition of polynomials is associative only up-to canonical isomorphisms.

Proposition 1.1. *Polynomial functors commute with connected limits.*

The simplest example of functor which is not polynomial is the *finite multiset functor*, seen as a functor from $\mathsf{Set}/_{\mathbf{1}} \cong \mathsf{Set}$ to itself: this functor doesn't commute with connected limits. When $\mathbb{C}$ is Set, the converse of proposition 1.1 also holds, giving a more "extensional" characterization of polynomial functors:

Proposition 1.2. *A functor $P : \mathsf{Set}/_I \to \mathsf{Set}/_J$ is polynomial iff it commutes with all connected limits.*

There are several other characterizations of polynomial functors on Set, all nicely summarized in [GK09].

2. Polynomials and Simulations: SMCC Structure

We will now construct a category where polynomials play the rôle of objects. More precisely, we will consider "endo-polynomials", i.e., diagrams of the form

$$I \xleftarrow{n} D \xrightarrow{d} A \xrightarrow{a} I \quad .$$

We simply call such a diagram a *polynomial over I* and think of them as games, as described on page 2.

2.1. Simulations.

The morphisms between two such polynomials over I and J will be spans between I and J, with some additional structure.

Definition 2.1. *If P_1 and P_2 are two polynomial functors over I_1 and I_2 respectively, a* simulation from P_1 to P_2 *is a diagram*

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

We can internalize the notion of simulation using the language of dependent types:

Proposition 2.1. *A simulation from P_1 to P_2 is given by: (refer to Definition 2.1)*

1. $i_1 : I_1, i_2 : I_2 \vdash R(i_1, i_2)$ *for the span,*
2. $i_1, i_2, r : R(i_1, i_2), a_1 : A_1(i_1) \vdash \alpha(i_1, i_2, r, a_1) : A_2(i_2)$ *for the morphism α,*
3. $i_1, i_2, r, a_1, d_2 : D_2\big(i_2, \alpha(...)\big) \vdash \beta(i_1, i_2, r, a_1, d_2) : D_1(i_1, a_1)$ *for the morphism β,*
4. $i_1, i_2, r, a_1, d_2 : D_2\big(i_2, \alpha(...)\big) \vdash \gamma(..., d_2) : R\big(i_2[a_1/\beta(...)], i_2[\alpha(...)/d_2]\big)$ *for the morphism γ,*

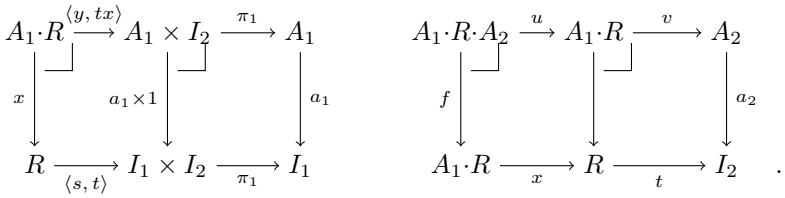
Putting all this together and rewriting it more concisely, a simulation is given by:

- $i_1 : I_1, i_2 : I_2 \vdash R$ *for the span,*
- $i_1, i_2 \vdash \pi : \prod_{a_1} \sum_{a_2} \prod_{d_2} \sum_{d_1} R\big(i_1[a_1/d_1], i_2[a_2/d_2]\big)$, *where for $k = 1, 2$ the types are $a_k : A_k(i_k)$ and $d_k : D_k(i_k, a_k)$.*

Proof. We'll only show the beginning in order to give a taste of the manipulations involved. Let's first fix some notation: the simulation is given by the diagram

$$\begin{array}{lccccccc}
 & I_1 & \xleftarrow{n_1} & D_1 & \xrightarrow{d_1} & A_1 & \xrightarrow{a_1} & I_1 \\
 & \uparrow{\scriptstyle s} & & \uparrow{\scriptstyle \beta} & & \uparrow{\scriptstyle y} & & \uparrow{\scriptstyle s} \\
 & R & \xleftarrow{\gamma} & \cdot & \to & A_1{\cdot}R & \dashrightarrow & R \\
 & \downarrow{\scriptstyle t} & & \downarrow & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle t} \\
 & I_2 & \xleftarrow[n_2]{} & D_2 & \xrightarrow[d_2]{} & A_2 & \xrightarrow[a_2]{} & I_2
\end{array} \quad .$$

The following pullbacks will be useful in the sequel:

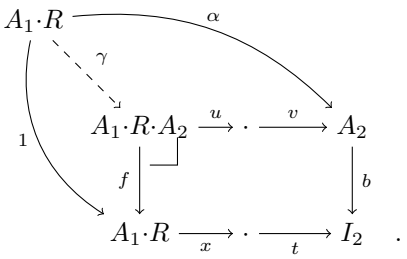


Interpreting the above types in $\mathbb{C}$ gives:

(1) $\vdash I_1$ is "I_1",
(2) $i_1 : I_1 \vdash I_2$ is "$\pi_1 : I_1 \times I_2 \to I_1$",
(3) $i_1, i_2 : I_2 \vdash R(i_1, i_2)$ is "$\langle s, t\rangle : R \to I_1 \times I_2$",
(4) $i_1, i_2, r : R(i_1, i_2) \vdash A_1(i_1)$ is "$x : A_1{\cdot}R \to R$",
(5) $i_1, i_2, r, a_1 : A_1(i_1) \vdash A_2(i_2)$ is "$f : A_1{\cdot}R{\cdot}A_2 \to A_1{\cdot}R$".

The term $i_1, i_2, r, a_1 \vdash \alpha(i_1, i_2, r, a_1) : A_2(i_2)$ thus corresponds to a section φ of f.

The sections of f are in 1-1 correspondence with the morphisms $\alpha : A_1{\cdot}R \to A_2$ s.t. $tx = b\alpha$: given such an α, construct γ as the mediating arrow in

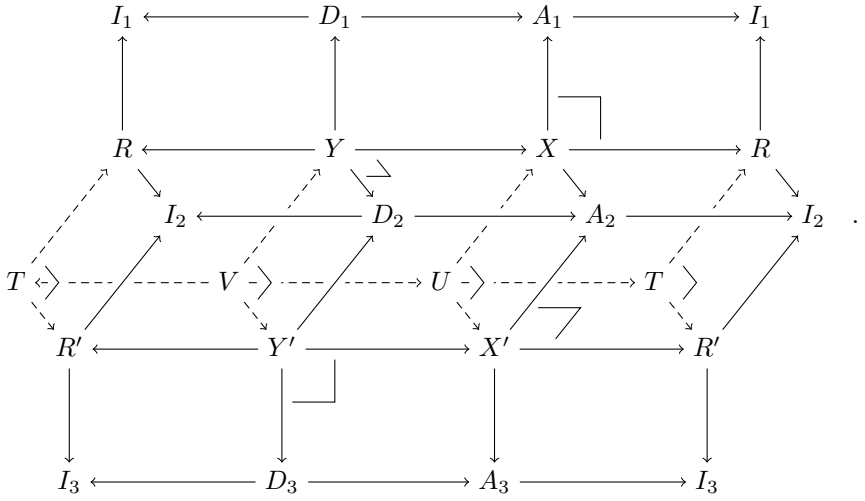


The lower triangle shows that this γ is a section.

The inverse of this construction is given by $\gamma \mapsto vu\gamma$. Because of the upper triangle above, this is indeed a left inverse of the "mediating arrow" transformation. This is also a right inverse: let γ be s.t. $f\gamma = \mathrm{id}$. We have $bvu\gamma = tx f\gamma = tx$ and so, if we use $\alpha \overset{\text{def}}{=} vu\gamma$ in the above diagram, the mediating arrow will necessarily be γ.

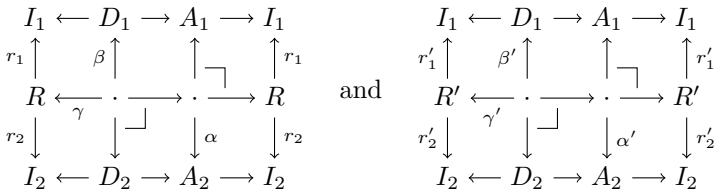
The rest is similar. □

Defining the composition of simulations using type theory isn't too difficult, but here is the diagrammatic representation of such a composition:

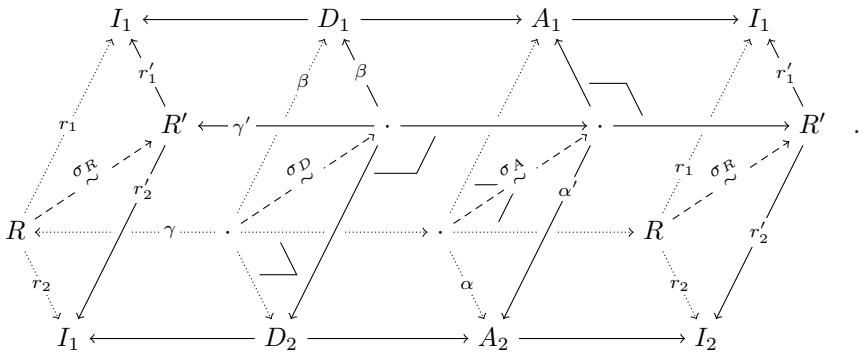


The plain arrows show the polynomials P_1, P_2 and P_3 and the initial two simulation diagrams. The dashed diagonal arrows are computed from the two simulations by pullbacks and the dashed horizontal arrows are mediating morphisms. To show that the "background layer" forms a simulation from P_1 to P_3, we need to show that the squares (U, A_1, I_1, T) and (V, U, A_3, D_3) are pullbacks. For (U, A_1, I_1, T), we know by the pullback lemma that (U, X, I_2, R') is a pullback by pasting (U, X, A_2, X') and (X', A_2, I_2, R'). A second application of the pullback lemma on (U, X, R, T) and $(T, R, I_2, R,)$ shows that (U, X, R, T) is also a pullback. Finally, a third application shows that (U, A_1, I_1, T) is a pullback, as expected. The same reasoning shows that the square (V, U, A_3, D_3) is also a pullback.

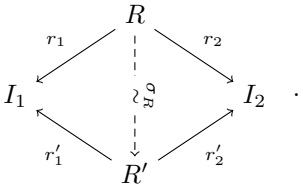
As in the case of spans, composition of simulations is only associative up to isomorphism. We will thus need to consider equivalence classes of such simulations. Two simulations



are equivalent if there are isomorphisms making the following diagram commute:



Such isomorphisms are in fact induced by a single span isomorphism



We can now define:

Definition 2.2. $\mathsf{PSim}_{\mathbb{C}}$ *is the category of polynomial endofunctors diagrams (simply called polynomials) and equivalence classes of simulation diagrams as in Definition 2.1.*

2.2. **Tensor Product and SMCC Structure.** The tensor is just a "pointwise cartesian product":

Definition 2.3. *The polynomial $P_1 \otimes P_2$ is defined as*

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

In terms of games, Alfred and Dominic play synchronously in the two games P_1 and P_2 at the same time. Composition and simulation diagrams lift pointwise, making it straightforward to check that

Lemma 2.1. *$\_ \otimes \_$ is a bifunctor in the category $\mathsf{PSim}_{\mathbb{C}}$. It has a neutral element given by $\mathbf{1} \leftarrow \mathbf{1} \rightarrow \mathbf{1} \rightarrow \mathbf{1}$ where $\mathbf{1}$ is the terminal element of $\mathbb{C}$.*

We will now prove that:

Proposition 2.2. *The category $\mathsf{PSim}_{\mathbb{C}}$ with $\otimes$ is symmetric monoidal closed, i.e., there is a functor $\_ \multimap \_$ from $\mathsf{PSim}_{\mathbb{C}}^{\mathrm{op}} \times \mathsf{PSim}_{\mathbb{C}}$ to $\mathsf{PSim}_{\mathbb{C}}$ with an adjunction*

$$\mathsf{PSim}_{\mathbb{C}}[P_1 \otimes P_2 \ , \ P_3] \quad \cong \quad \mathsf{PSim}_{\mathbb{C}}[P_1 \ , \ P_2 \multimap P_3] \ ,$$

natural in P_1 and P_3.

Let's start by giving a definition of $P_2 \multimap P_3$ using the internal language of LCCC. A purely diagrammatic definition of $P_2 \multimap P_3$ will follow.

Definition 2.4. *The polynomial $P_2 \multimap P_3$ is defined as:*

(1) $\vdash I_2 \times I_3$,
(2) $i_2, i_3 \vdash \sum_{f:A_2(i_2)\to A_3(i_3)} \prod_{a_2:A_2(i_2)} D_3\big(i_3, f(a_2)\big) \to D_2(i_2, a_2)$,
(3) $i_2, i_3 \ , \ f, \varphi \vdash \sum_{a_2:A_2(i_2)} D_3\big(i_3, f(a_2)\big)$,
(4) $i_2, i_3 \ , \ f, \varphi \ , \ a_2, d_3 \vdash \big(i_2[a_2/\varphi(a_2)(d_3)], i_3[f(a_2)/d_3]\big) : I_2 \times I_3$,

where the types of variables are as follows: f is of type $A_2(i_2) \to A_3(i_3)$, φ is of type $\prod_{a_2:A_2(i_2)} D_3\big(i_3, f(a_2)\big) \to D_2(i_2, a_2)$, a_2 is of type $A_2(i_2)$ and d_3 is of type $D_3\big(i_3, f(a_2)\big)$.

Proof of proposition 2.2. There is a canonical natural isomorphism

$$\mathsf{Span}_{\mathbb{C}}[I_1, I_2 \times I_3] \quad \cong \quad \mathsf{Span}_{\mathbb{C}}[I_1 \times I_2, I_3] \quad \cong \quad \mathbb{C}/_{I_1 \times I_2 \times I_3}$$

and we use it implicitly. In order to show the adjunction, we need to find a natural isomorphism between

$$\prod_{a_1, a_2} \sum_{a_3} \prod_{d_3} \sum_{d_1, d_2} R\big(i_1[a_1/d_1] \ , \ i_2[a_2/d_2] \ , \ i_3[a_3/d_3]\big)$$

meaning that R is a simulation from $P_1 \otimes P_2$ to P_3 and

$$\prod_{a_1} \sum_{f, \varphi} \prod_{a_2, d_3} \sum_{d_1} R\big(i_1[a_1/d_1] \ , \ i_2[a_2/\varphi(a_2)(d_3)] \ , \ i_3[f(a_2)/d_3]\big) \ ,$$

meaning that R is a simulation from P_1 to $P_2 \multimap P_3$. The types are as follows:

- $a_k : A_k(i_k)$ for $k = 1, 2, 3$,
- $d_k : D_k(a_k)$ for $k = 1, 2, 3$,
- $f : A_2(i_2) \to A_3(i_3)$,
- $\varphi : \prod_{a_2} D_3\big(f(a_2)\big) \to D_2(a_2)$.

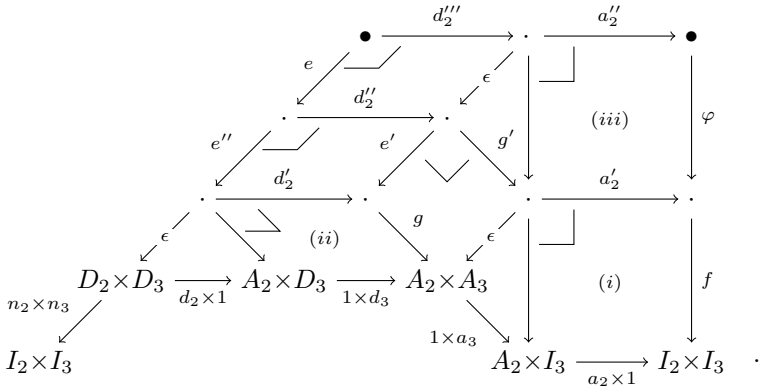
This is just a sequence of "distributivity" (type theoretic axiom of choice, page 5) and obvious isomorphisms changing the order of independent variables:

(1) from $\prod_{a_2} \sum_{a_3}$ to $\sum_f \prod_{a_2}$, to get $\prod_{a_1} \sum_f \prod_{a_2} \prod_{d_3} \sum_{d_1, d_2} \ \cdots$
(2) from $\prod_{d_3} \sum_{d_2}$ to $\sum_g \prod_{d_3}$, to get $\prod_{a_1} \sum_f \prod_{a_2} \sum_g \prod_{d_3} \sum_{d_1} \ \cdots$
(3) from $\prod_{a_2} \sum_g$ to $\sum_\varphi \prod_{a_2}$, to get $\prod_{a_1} \sum_f \sum_\varphi \prod_{a_2} \prod_{d_3} \sum_{d_1} \ \cdots$

The "…" use exactly the appropriate substitutions to make the last line into what was needed: $a_3 \stackrel{\mathsf{def}}{=} f(a_2)$, $d_2 \stackrel{\mathsf{def}}{=} g(d_3)$ and $g \stackrel{\mathsf{def}}{=} \varphi(a_2)$. □

For completeness, here is the diagrammatic definition of $P_2 \multimap P_3$:

Lemma 2.2. *The polynomial $P_2 \multimap P_3$ is given by*



where the 1*s are appropriate identities and squares* (i)*,* (ii) *and* (iii) *are distributivity squares as in diagram* (1)*, i.e., the* ϵ*s are appropriate counits of the adjunction* $\Delta_{\_} \dashv \Pi_{\_}$*.*

It is not too difficult to show that this corresponds to the type theoretic definition. However, an elegant and direct proof that this is indeed the right adjoint for $\_ \otimes P_2$ is still to be found. One easy thing is the following (compare it with the second part of Proposition 2.1)

Lemma 2.3. *The extension of* $P_2 \multimap P_3$ *is*

$$\begin{aligned} P_2 \multimap P_3 \quad &= \quad \Pi_{a_2 \times 1}\, \Sigma_{1 \times a_3}\, \Pi_{1 \times d_3}\, \Sigma_{d_2 \times 1}\, \Delta_{n_2 \times n_3} \\ &= \quad \Pi_{a_2 \times 1}\, \mathbf{1} \otimes P_3\, \Sigma_{d_2 \times 1}\, \Delta_{n_2 \times 1} \ . \end{aligned}$$

Proof. This is just a rewriting of the definition using Beck-Chevalley and distributivity as appropriate. Using the notation from Lemma 2.2:

$$\begin{aligned} P_2 \multimap P_3 \quad &= \quad \Sigma_f\, \Sigma_\varphi\, \Pi_{a_2''}\, \underline{\Pi_{d_2'''}\, \Delta_e}\, \Delta_{e''}\, \Delta_\epsilon\, \Delta_{n_2 \times n_3} \\ &= \quad \Sigma_f\, \underline{\Sigma_\varphi\, \Pi_{a_2''}\, \Delta_\epsilon}\, \Pi_{d_2''}\, \Delta_{e''}\, \Delta_\epsilon\, \Delta_{n_2 \times n_3} \\ &= \quad \Sigma_f\, \Pi_{a_2'}\, \Sigma_{g'}\, \underline{\Pi_{d_2''}\, \Delta_{e''}}\, \Delta_\epsilon\, \Delta_{n_2 \times n_3} \\ &= \quad \Sigma_f\, \Pi_{a_2'}\, \underline{\Sigma_{g'}\, \Delta_{e'}}\, \Pi_{d_2'}\, \Delta_\epsilon\, \Delta_{n_2 \times n_3} \\ &= \quad \underline{\Sigma_f\, \Pi_{a_2'}\, \Delta_\epsilon}\, \underline{\Sigma_g\, \Pi_{d_2'}\, \Delta_\epsilon}\, \Delta_{n_2 \times n_3} \\ &= \quad \Pi_{a_2 \times 1}\, \Sigma_{1 \times a_3}\, \Pi_{1 \times d_3}\, \Sigma_{d_2 \times 1}\, \Delta_{n_2 \times n_3} \ . \end{aligned}$$

The second equality follows from a Beck-Chevalley isomorphism:

$$\Sigma_{d_2 \times 1} \Delta_{1 \times n_3} \quad = \quad \Delta_{1 \times n_3} \Sigma_{d_2 \times 1} \ .$$

□

2.3. **Linear Negation.** Of particular interest is the dual of P: $P^\perp \stackrel{\text{def}}{=} P \multimap \mathbf{1}$ where $\mathbf{1}$ is the neutral element for $\otimes$.

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

In the denotational model for intuitionistic linear logic, this polynomial is the most natural choice for the "$\perp$" object. The polynomial $P^\perp$ is thus the "linear negation" of P. By simplifying definition 2.4 in this case, we obtain:

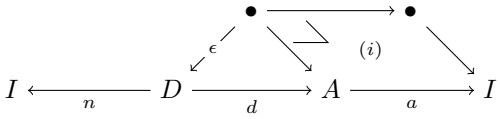
Definition 2.5. *Given a polynomial $P = (I \leftarrow D \rightarrow A \rightarrow I)$, the polynomial $P^\perp$ is defined by:*

(1) $\vdash I$,
(2) $i : I \vdash \prod_{a:A(i)} D(i,a)$,
(3) $i : I, f : \prod_{a:A(i)} D(i,a) \vdash A(i)$,
(4) $i : I, f : \prod_{a:A(i)} D(i,a), a : A(i) \vdash n\big(i, a, f(a)\big) : I$.

Note that this negation isn't involutive.

The dual $G^\perp$ of a game G is rather different from the usual operation consisting of interchanging the players as is done in games semantics. The main property is that a strategy (for Alfred) in $G^\perp$ is exactly a strategy for Dominic in G.

Simplifying the diagrammatic definition of $\multimap$ (Lemma 2.2), we find that $P^\perp$ is

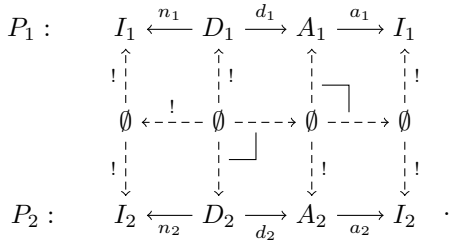


where square (i) is a distributivity square. The middle arrow is thus of the form $\Delta_{\_}a$. It is worth noting that for any polynomial P, the polynomial $P^\perp$ has the form of "simultaneous games" described on page 3.

3. Additive and Exponential Structure

To go further, we will need more structure from $\mathbb{C}$. To avoid spelling out the exact requirements (extensivity, existence of certain colimits etc.), we now work in the category of sets and functions $\mathbb{C} = \mathsf{Set}$, where we certainly have all we need.

3.1. **Enriched Structure.** Because $\emptyset$ is initial in Set, there is an "empty" simulation between any two polynomial functors. It is given by the diagram



This is true for any $\mathbb{C}$ having an initial object $\mathbf{0}$ because in an LCCC, we necessarily have $\mathbf{0} \times_I X \cong \mathbf{0}$.

Moreover, each $\mathsf{Set}/_X$ is a cocomplete category and span composition is continuous on both sides, making the $\mathsf{Span}_{\mathsf{Set}}$ enriched over (large) sup-monoids. Because a colimit of simulations is easily made into a simulation, this implies that $\mathsf{PSim}_{\mathsf{Set}}$ is also enriched over sup-monoids.

Proposition 3.1. $\mathsf{PSim}_{\mathsf{Set}}$ *is enriched over large sup-monoids.*

3.2. **Additive Structure.** The category Set is extensive. This means in particular that a slice $f \in \mathsf{Set}/_{B+C}$ can be uniquely (up-to isomorphism) written as $f_B + f_C$ for some $f_B \in \mathsf{Set}/_B$ and $f_C \in \mathsf{Set}/_C$. In other words, $+$ is an equivalence of categories $\mathsf{Set}/_{B+C} \cong \mathsf{Set}/_B \times \mathsf{Set}/_C$. Extensivity implies that in $\mathsf{Span}_{\mathsf{Set}}$:

- $\emptyset$ is a zero object,
- the coproduct of X and Y is given by $X + Y$ (disjoint union),
- $X + Y$ is also the product of X and Y,

Moreover, the infinite coproduct from Set $\sum_{k\in K} X_k$ lifts to the infinite coproduct and product in $\mathsf{Span}_{\mathsf{Set}}$. We have:

Lemma 3.1. *The forgetful functor $U : \mathsf{PSim}_{\mathsf{Set}} \to \mathsf{Span}_{\mathsf{Set}}$ sending a polynomial to its domain and a simulation to its underlying span has a left and a right adjoint.*

Proof. The object part of the adjoints $L \dashv U \dashv R$ for set I are given by

$$L(I) \stackrel{\text{def}}{=} I \xleftarrow{!} \emptyset \xrightarrow{!} \emptyset \xrightarrow{!} I \qquad \text{and} \qquad R(I) \stackrel{\text{def}}{=} I \xleftarrow{!} \emptyset \xrightarrow{!} I \xrightarrow{1} I \quad .$$

The rest is simple verification. □

Because the forgetful functor U has a left adjoint, it must preserve the product (and so, the coproduct as well). The product of P_1 and P_2 is thus a polynomial over $I_1 + I_2$.

Definition 3.1. *If P_1 and P_2 are polynomials over I_1 and I_2, we write $P_1 \oplus P_2$ for*

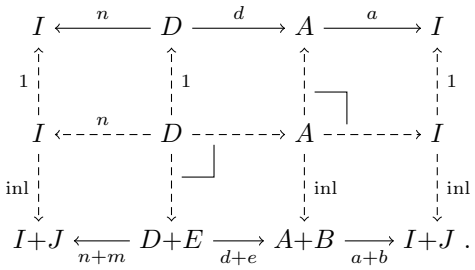
$$P_1 \oplus P_2 \quad \stackrel{\text{def}}{=} \quad I_1 + I_2 \xleftarrow{n_1+n_2} D_1 + D_2 \xrightarrow{d_1+d_2} A_1 + A_2 \xrightarrow{a_1+a_2} I_1 + I_2 \quad .$$

The polynomial $\mathbf{0}$ is the unique polynomial with domain and codomain $\emptyset$.

We have

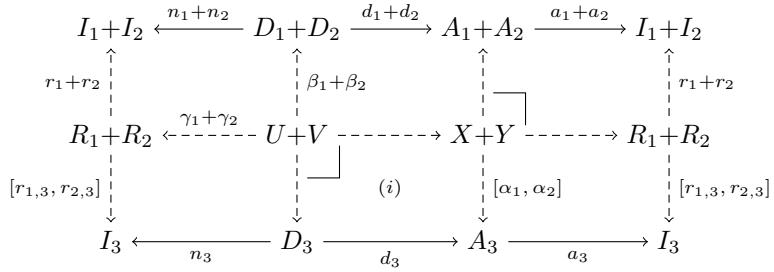
Lemma 3.2. *The bifunctor $\oplus$ is both a product and a coproduct in $\mathsf{PSim}_{\mathsf{Set}}$. The polynomial $\mathbf{0}$ is a zero object.*

Proof. The "injections" are given by



and similarly for the "right" injection. Because all the six squares are in fact pullbacks, this also defines the projections by mirroring everything horizontally.

Now, because Set is extensive, any simulation from $P_1 \oplus P_2$ to P_3 is of the form:



where $[\_,\_]$ is the "copairing" and all the remaining morphisms are either of the form $[f, g]$ or $f + g$.[1] It is easy to split this into two simulations: one from P_1 to P_3 and the other from P_2 to P_3. Checking that those are simulations is straightforward. Conversely, we can construct a simulation as above from any two simulations. The constructions are inverse to each other (up-to isomorphism).

[1] A small lemma is necessary for square (i).

This shows that $\oplus$ is a coproduct. Because $\mathsf{PSim}_{\mathsf{Set}}$ is enriched over (large) commutative monoids coproduct is also a product.

The proof that the polynomial $\mathbf{0}$ is a zero object is left to the reader. □

This proof also extends to infinite coproducts:

Lemma 3.3. *For a set K and polynomials P_k for $k \in K$, the polynomial*

$$\bigoplus_{k\in K} P_k \quad \stackrel{\mathsf{def}}{=} \quad \coprod_k I_k \xleftarrow{\coprod_k n_k} \coprod_k D_k \xrightarrow{\coprod_k d_k} \coprod_k A_k \xrightarrow{\coprod_k a_k} \coprod_k I_k$$

is both the cartesian product and coproduct of the polynomials P_k in $\mathsf{PSim}_{\mathsf{Set}}$.

3.3. **Exponentials.** Whenever the infinite coproduct distributes over a binary tensor $\odot$:

$$\coprod_{k\geq 0}(X \odot I_k) \quad \cong \quad X \odot \coprod_{k\geq 0} I_k$$

the free $\odot$-monoid over I and free commutative $\odot$-monoid over I are given by

$$\coprod_{k\geq 0} I^{\odot k} \quad \text{and} \quad \coprod_{k\geq 0} S_k(I)$$

where $S_k(I)$ is the coequalizer of the $k!$ symmetries on $I^{\odot k}$.

For the cartesian product $\times$ on the category Set, we obtain finite words and finite multisets, i.e. equivalence classes of words under permutations:

$$I^* \quad \stackrel{\mathsf{def}}{=} \quad \coprod_{k\geq 0} I^k$$

and

$$\mathcal{M}_f(I) \quad \stackrel{\mathsf{def}}{=} \quad \coprod_{k\geq 0} \mathcal{M}_f^k(I) \quad \stackrel{\mathsf{def}}{=} \quad \coprod_{k\geq 0} I^k/\mathfrak{S}_k$$

where $\mathfrak{S}_k$ is the group of permutations of $\{1,\dots,k\}$, acting in an obvious way on I^k.

The next lemma is probably folklore among the right people, but I could find no proof in the literature. A proof is given in appendix on page 19.

Lemma 3.4. *The operation $\mathcal{M}_f(\_)$ is the object part of a monad on $\mathsf{Span}_{\mathsf{Set}}$. This monad gives the free commutative $\times$-monoid in $\mathsf{Span}_{\mathsf{Set}}$. Because $\mathsf{Span}_{\mathsf{Set}}$ is self-dual, this is also the free commutative $\times$-comonoid comonad.*

The unit and multiplication are inherited from Set:

$$w_A \stackrel{\mathsf{def}}{=} \begin{array}{ccc} & \mathbf{1} & \\ {\scriptstyle 1}\swarrow & & \searrow{\scriptstyle \varepsilon} \\ \mathbf{1} & & \mathcal{M}_f(A) \end{array} \qquad c_A \stackrel{\mathsf{def}}{=} \begin{array}{ccc} & \mathcal{M}_f(A)\times\mathcal{M}_f(A) & \\ {\scriptstyle 1}\swarrow & & \searrow{\scriptstyle \uplus} \\ \mathcal{M}_f(A)\times\mathcal{M}_f(A) & & \mathcal{M}_f(A) \end{array}$$

where ε picks the empty multiset and $\uplus$ is the union of multisets.

Just as in $\mathsf{Span}_{\mathsf{Set}}$, the infinite product (which is also the coproduct) distributes over the binary tensor in $\mathsf{PSim}_{\mathsf{Set}}$:

$$\bigoplus_{k\geq 0} Q \otimes P_k \quad \cong \quad Q \otimes \bigoplus_{k\geq 0} P_k \ .$$

We can thus use the dual formula to get the free commutative $\otimes$-comonoid.

$$!P = \bigoplus_{k\geq 0} P^k$$

where P^k is the equalizer of all the symmetries on $P^{\otimes k}$. Because the forgetful functor U is a right adjoint, it preserves products and equalizers. Because U is monoidal, the above formula implies that it preserves the free comonoid. Thus, the polynomial $!P$ has domain $\mathcal{M}_f(I)$ whenever P has domain I.

Definition 3.2. *If P is a polynomial, define $\mathbf{!}P$ to be the following polynomial*

$$\mathbf{!}P \quad \stackrel{\mathsf{def}}{=} \quad \mathcal{M}_f(I) \xleftarrow{cn^*} D^* \xrightarrow{d^*} A^* \xrightarrow{ca^*} \mathcal{M}_f(I)$$

where c takes a word to its orbit.

Note that this is *not* a pointwise application of $\mathcal{M}_f$, which would give

$$\mathcal{M}_f(I) \xleftarrow{\mathcal{M}_f(n)} \mathcal{M}_f(D) \xrightarrow{\mathcal{M}_f(d)} \mathcal{M}_f(A) \xrightarrow{\mathcal{M}_f(a)} \mathcal{M}_f(I) \quad .$$

Proposition 3.2. *The free commutative $\otimes$-comonoid comonad on $\mathsf{Span}_{\mathsf{Set}}$ lifts to the category $\mathsf{PSim}_{\mathsf{Set}}$. Its action on objects is given by $P \mapsto !P$.*

Just as in $\mathsf{Span}_{\mathsf{Set}}$ it is sufficient to check that

$$P^k \quad \stackrel{\mathsf{def}}{=} \quad \mathcal{M}_f^k(I) \xleftarrow{cn^k} D^k \xrightarrow{d^k} A^k \xrightarrow{ca^k} \mathcal{M}_f^k(I)$$

is the equalizer of the symmetries. Both the diagrammatic proof and the type theoretic proof are possible but very tedious and we will only show that P^k is the equalizer of the symmetries in the category $\mathsf{PSim}_{\mathsf{Set}\sim}$, obtained from $\mathsf{PSim}_{\mathsf{Set}}$ by identifying any two simulations when their spans are isomorphic. This makes the forgetful functor $U : \mathsf{PSim}_{\mathsf{Set}\sim} \to \mathsf{Span}_{\mathsf{Set}}$ faithful, simplifying the argument.

First, some notation:

- tuples are denoted using the Gothic alphabet: $\mathfrak{u} \in U^k$, $\mathfrak{i} \in I^k$ etc.
- any permutation $\sigma \in \mathfrak{S}_k$ induces a natural transformation $\_^k \to \_^k$,
- $c : \_^* \to \mathcal{M}_f(\_)$ is the natural transformation sending a tuple to its orbit,
- for any set X, $s : \mathcal{M}_f(X) \rightarrowtail X^*$ is a section of $c_X : X^* \twoheadrightarrow \mathcal{M}_f(X)$;[2]

and a preliminary lemma:

Lemma 3.5. *In Set, suppose $h : U^k \to U^k$ sends any element of U^k to a permutation of itself, i.e.,*

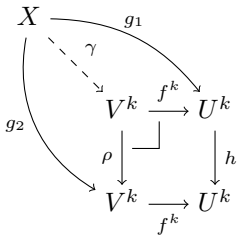
$$\begin{array}{ccc} U^k & \xrightarrow{c} & \mathcal{M}_f^k(U) \\ {\scriptstyle h}\uparrow & \nearrow_{c} & \\ U^k & & \end{array} \quad .$$

For any $g : V \to U$, we can find a $\rho : V^k \to V^k$ with the same property, i.e., with $c\rho = c$ such that:

$$\begin{array}{ccccc} \mathcal{M}_f^k(V) & \xleftarrow{c} & V^k & \xrightarrow{f^k} & U^k \\ & {\scriptstyle c}\searrow & {\scriptstyle \rho}\downarrow & & \downarrow{\scriptstyle h} \\ & & V^k & \xrightarrow[f^k]{} & U^k \end{array} \quad .$$

[2] this transformation cannot be made natural

Proof. Define $\rho : \mathfrak{v} \mapsto \sigma_{f^k(\mathfrak{v})}(\mathfrak{v})$, where $\sigma_{\mathfrak{u}}$ is any permutation s.t. $f^k(\mathfrak{u}) = \sigma_{\mathfrak{u}}(\mathfrak{u})$. This ρ makes the diagram commute. To show that the square is a pullback, we construct mediating arrows as follows: given



we put $\gamma(x) \stackrel{\text{def}}{=} \sigma^{-1}_{g_1(x)}\big(g_2(x)\big)$. We have

$$f^k\gamma(x) = f^k\sigma^{-1}_{g_1(x)}g_2(x) = \sigma^{-1}_{g_1(x)}f^k g_2(x) = \sigma^{-1}_{g_1(x)}g_1h(x) = g_1(x)$$

where the last equality comes from $hg_1(x) = \sigma_{g_1(x)}\big(g_1(x)\big)$. For the second triangle:

$$\rho\gamma(x) = \sigma_{f^k\gamma(x)}\gamma(x) = \sigma_{g_1(x)}\gamma(x) = \sigma_{g_1(x)}\sigma^{-1}_{g_1(x)}g_2(x) = g_2(x)\ .$$

Moreover, for any other mediating γ', we must have

$$g_2(x) \quad = \quad \rho\gamma'(x) \quad = \quad \sigma_{f^k\gamma'(x)}\gamma'(x) \quad = \quad \sigma_{g_1(x)}\gamma'(x)$$

which implies that $\gamma' = \gamma$. □

Proof of Proposition 3.2. We need to show that P^k is the equalizer of the symmetries on $P^{\otimes k}$. We first need to make $\hat{c}$ into a simulation from P^k to $P^{\otimes k}$, i.e., we need to define α, β and γ filling the diagram

$$\begin{array}{lccccccc}
P^k: & \mathcal{M}^k_f(I) & \longleftarrow & D^k & \longrightarrow & A^k & \longrightarrow & \mathcal{M}^k_f(I) \\
& \uparrow{\scriptstyle c} & & \uparrow{\scriptstyle \beta} & & \uparrow & & \uparrow{\scriptstyle c} \\
& I^k & \longleftarrow{\scriptstyle \gamma} & Y & \longrightarrow & X & \longrightarrow & I^k \\
& \downarrow{\scriptstyle 1} & & \downarrow & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle 1} \\
P^{\otimes k}: & I^k & \underset{n^k}{\longleftarrow} & D^k & \underset{d^k}{\longrightarrow} & A^k & \underset{a^k}{\longrightarrow} & I^k
\end{array} \quad .$$

The set X is (isomorphic to) $\big\{(\mathfrak{i},\mathfrak{a}) \;\big|\; \mathfrak{i} \sim a^k(\mathfrak{a})\big\}$ and the function α sends $(\mathfrak{i},\mathfrak{a})$ to $\sigma_{\mathfrak{i},\mathfrak{a}}(\mathfrak{a})$ where $\sigma_{\mathfrak{i},\mathfrak{a}}$ is a permutation such that $\sigma_{\mathfrak{i},\mathfrak{a}}\big(a^k(\mathfrak{a})\big) = \mathfrak{i}$; and the set Y is (isomorphic to) $\big\{(\mathfrak{i},\mathfrak{a},\mathfrak{d}) \;\big|\; \mathfrak{i} \sim a^k(\mathfrak{a}), d^k(\mathfrak{d}) = \mathfrak{a}\big\}$. The function β sends $(\mathfrak{i},\mathfrak{a},\mathfrak{d})$ to $\sigma^{-1}_{\mathfrak{i},\mathfrak{a}}(\mathfrak{d})$ and the function γ sends $(\mathfrak{i},\mathfrak{a},\mathfrak{d})$ to $n^k(\mathfrak{d})$.

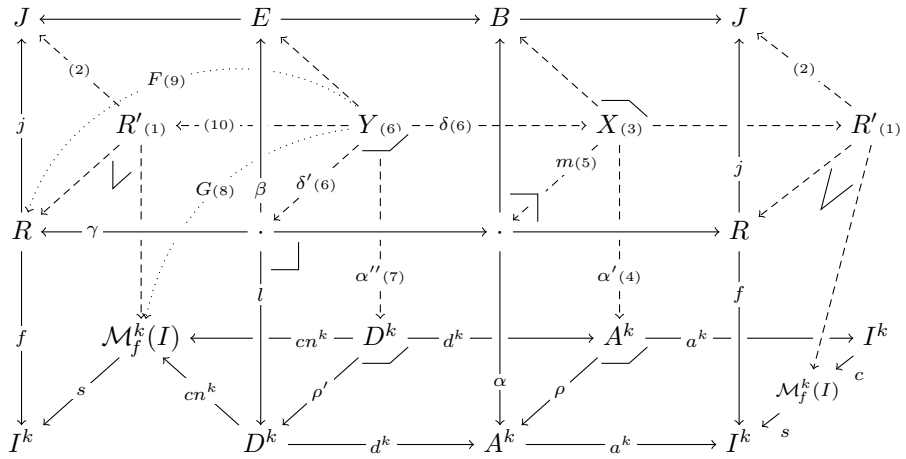
Now, given a simulation from Q to $P^{\otimes k}$

$$\begin{array}{lccccccc}
Q: & J & \longleftarrow & E & \longrightarrow & B & \longrightarrow & J \\
& \uparrow{\scriptstyle j} & & \uparrow{\scriptstyle \beta} & & \uparrow & & \uparrow{\scriptstyle j} \\
& R & \longleftarrow{\scriptstyle \gamma} & \cdot & \longrightarrow & \cdot & \longrightarrow & R \\
& \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle f} \\
P^{\otimes k}: & I^k & \underset{n^k}{\longleftarrow} & D^k & \underset{d^k}{\longrightarrow} & A^k & \underset{a^k}{\longrightarrow} & I^k
\end{array} \tag{3}$$

which equalizes the symmetries, we need to construct a simulation from Q to P^k. That (3) equalizes the symmetries implies in particular that

$$\forall \sigma \in \mathfrak{S}_k \ \exists H \qquad \begin{array}{ccccc} & & R & & \\ & \swarrow^{f} & \uparrow{\scriptstyle H} & \searrow^{j} & \\ I^k & & & & J \\ & \nwarrow_{\sigma f} & & \nearrow_{j} & \\ & & R & & \end{array} \quad . \tag{4}$$

The simulation from Q to $P^k = \mathcal{M}_f^k(I) \leftarrow D^k \to A^k \to \mathcal{M}_f^k(I)$ is constructed in several steps as indicated by the small numbers in parenthesis:



where the bottom layer comes from Lemma 3.5 and:

- (1) R' is constructed by pullback;
- (2) is obtained by composition;
- (3) X is obtained by pullback;
- (4) α' is the mediating arrow, where (3) $\to I^k$ is (3) $\to R' \to \mathcal{M}_f^k(I) \to I^k$;
- (5) m is the mediating arrow;
- (6) Y, δ and δ' are constructed by pullback;
- (7) α'' is the mediating arrow;
- (8) G sends y to $cn^k\alpha''(y)$;
- (9) F sends y to $H^{-1}_{f\gamma\delta'(y)}\big(\gamma\delta'(y)\big)$, where for $\mathfrak{i} \in I^k$, $H_{\mathfrak{i}}$ is the automorphism in diagram (4) corresponding to $\sigma_{\mathfrak{i}}$, chosen such that $\sigma_{\mathfrak{i}} = sc(\mathfrak{i})$;
- (10) is the mediating arrow corresponding to F and G.

We only need to check that (8) and (9) make the appropriate diagram commute, i.e., that $sG = fF$:

$$\begin{array}{rcl} sG(y) & = & scn^k\alpha''(y) \\ & = & scn^k\rho'\alpha''(y) \\ & = & scn^kl\delta'(y) \\ (\star) & = & scf\gamma\delta'(y) \\ & = & \sigma_{f\gamma\delta'(y)}f\gamma\delta'(y) \\ & = & fH^{-1}_{f\gamma\delta'(y)}\gamma\delta'(y) \\ & = & fF(y) \end{array}$$

where equality $(\star)$ comes from diagram (3). Because square $(\delta, \alpha', \alpha'', d^k)$ is a pullback, this gives a simulation from Q to P^k.

That this simulation is the mediating arrow for the equalizer diagram in $\mathsf{PSim}_{\mathsf{Set}\sim}$ follows from the fact that its corresponding span is indeed the mediating span in $\mathsf{Span}_{\mathsf{Set}}$, together with the fact that the forgetful functor $U : \mathsf{PSim}_{\mathsf{Set}\sim} \to \mathsf{Span}_{\mathsf{Set}}$ is faithful.

□

Concluding Remarks

Toward Differential Logic. As noted in section 3.1, the category $\mathsf{PSim}_{\mathsf{Set}}$ is enriched over large commutative monoids. Moreover, $\mathsf{PSim}_{\mathsf{Set}}$ has enough duality to make $!P$ into a commutative $\otimes$-*monoid*. These are key features when one interprets differential logic [ER03, ER06]. However, trying to lift the differential structure of Rel, the category of sets and relations to the category $\mathsf{Span}_{\mathsf{Set}}$ fails as the candidate for the deriving transformation [BCS06]:

$$\partial_X \quad : \quad X \otimes !X \to !X\partial_X \quad \stackrel{\mathsf{def}}{=} \quad \begin{array}{ccccc} & & X \times \mathcal{M}_f(X) & & \\ & \swarrow^{1} & & \searrow^{@} & \\ X \times \mathcal{M}_f(X) & & & & \mathcal{M}_f(X) \end{array}$$

is only *lax natural*. It is the only obstruction to get a differential category in the sense of Blute, Cockett and Seely [BCS06] as the four coherence conditions seem to hold both in $\mathsf{Span}_{\mathsf{Set}}$ and in $\mathsf{PSim}_{\mathsf{Set}}$, even if the full proof for the later is rather long. (As with Proposition 3.2, the proof is much simpler for the category $\mathsf{PSim}_{\mathsf{Set}\sim}$.) Whether lax naturality is enough to model differential logic is still to be investigated.

Extensional Version: Polynomial Functors and Simulation Cells. This work was very "intensional" in that it only dealt with polynomial *diagrams* and not at all with polynomial *functors*. The different notions presented here have a more "extensional" version which does not rely on knowing a particular representation of the polynomial functors. In particular, the tensor and the linear arrow of two polynomial functors can be defined by universal properties. The corresponding category $\mathsf{FSim}_{\mathbb{C}}$, has polynomial functors as objects, and morphisms (simulations) are given by cells of the form

$$\begin{array}{ccc} \mathbb{C}/_{I_1} & \xrightarrow{P_1} & \mathbb{C}/_{J_1} \\ \downarrow{\scriptstyle L} & \swarrow^{\alpha} & \downarrow{\scriptstyle L} \\ \mathbb{C}/_{I_2} & \xrightarrow[P_2]{} & \mathbb{C}/_{J_2} \end{array}$$

where L is a "linear" polynomial functor. Composition is simply obtained by pasting such cells vertically. See the upcoming [Hyv12] for details.

References

[AAG05] Michael Abott, Thorsten Altenkirch, and Neil Ghani, *Containers - constructing strictly positive types*, Theoretical Computer Science **342** (2005), 3–27.

[BCS06] Richard F. Blute, J. Robin B. Cockett, and Robert A. G. Seely, *Differential categories*, Mathematical Structures in Computer Science **16** (2006), 1049–1083.

[ER03] Thomas Ehrhard and Laurent Regnier, *The differential lambda calculus*, Theoretical Computer Science **309** (2003), no. 1, 1–41.

[ER06] ———, *Differential interaction nets*, Theoretical Computer Science **364** (2006), 166–195.

[GK09] Nicola Gambino and Joachim Kock, *Polynomial functors and polynomial monads*, To appear in Mathematical Proceedings of the Cambridge Philosophical Society, `arXiv: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.

[Hyv05] Pierre Hyvernat, *A logical investigation of interaction systems*, Thèse de doctorat, Institut mathématique de Luminy, Université Aix-Marseille II, 2005.

[Hyv12] ———, *A linear category of polynomial functors (extensional part)*, In preparation, 2012.

[Joy77] André Joyal, *Remarques sur la théorie des jeux à deux personnes*, Gazette des sciences mathématiques du Quebec **1** (1977), no. 4, 175.

[Koc09] Joachim Kock, *Notes on polynomial functors*, preliminary draft, 2009.

[MA09] Peter Morris and Thorsten Altenkirch, *Indexed containers*, Twenty-Fourth IEEE Symposium in Logic in Computer Science (LICS 2009), 2009.

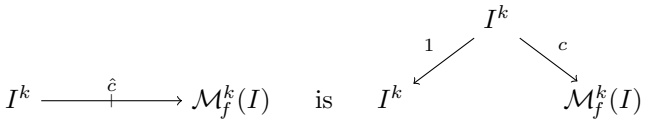
[ML84] Per Martin-Löf, *Intuitionistic type theory*, Bibliopolis, Naples, 1984, Notes by Giovanni Sambin. MR 86j:03005

[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)

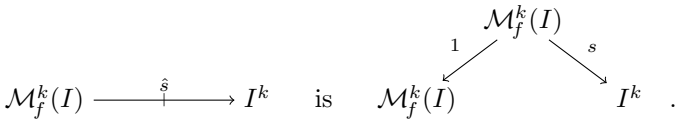
Appendix A. Free Commutative ×-Monoid in $\mathsf{Span}_{\mathsf{Set}}$

Lemma A.1. *The operation $\mathcal{M}_f(\_)$ is the object part of a monad on $\mathsf{Span}_{\mathsf{Set}}$. This monad gives the free commutative ×-monoid in $\mathsf{Span}_{\mathsf{Set}}$. Because $\mathsf{Span}_{\mathsf{Set}}$ is self-dual, this is also the free commutative ×-comonoid comonad.*

Proof. Because the coproduct distributes over ×, and because coproducts in $\mathsf{Span}_{\mathsf{Set}}$ are computed as in Set, we only need to show that $\mathcal{M}_f^k(I)$ is the coequalizer of all the symmetries on I^k. Let $c : I^* \twoheadrightarrow \mathcal{M}_f(I)$ be the function sending a word to its corresponding multiset, and let $s : \mathcal{M}_f(I) \rightarrowtail I^*$ be a section of c, i.e., a function choosing a representative for each equivalence class. This gives rise to a pair retraction/section in $\mathsf{Span}_{\mathsf{Set}}$:



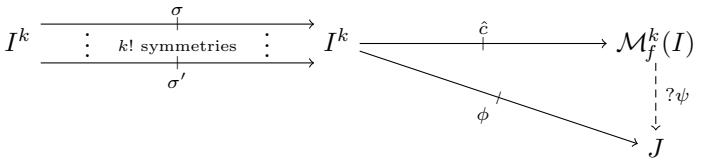
and



.

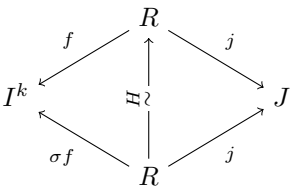
We'll show that $\hat{c}$ is the coequalizer of the symmetries: consider

(5)



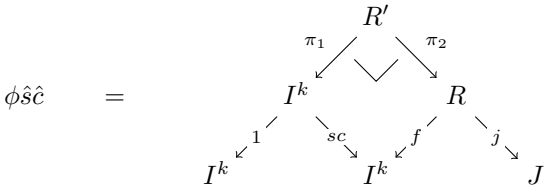
where the σs are spans with the identity for right leg and permutations for left leg. It is immediate that $\hat{c}$ coequalizes them. Suppose moreover that $\phi = I^k \leftarrow R \rightarrow J$ coequalizes them, i.e.,

(6)



.

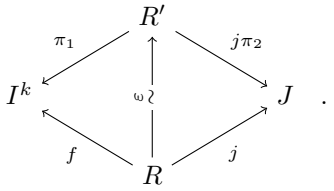
To close the triangle in (5), put $\psi \stackrel{\mathsf{def}}{=} \phi\hat{s}$. We need to check that $\psi\hat{c} = \phi$, i.e., that $\phi\hat{s}\hat{c} = \phi$. We have



where

$$R' \stackrel{\mathsf{def}}{=} \left\{ (\mathfrak{i}, r) \in I^k \times R \;\middle|\; sc(\mathfrak{i}) = f(r) \right\} .$$

To show that this span is equal to ϕ, we need to find an isomorphism between R and R' s.t.

(7)


To do that, note that for any word $\mathfrak{i} \in I^k$, the word $sc(\mathfrak{i})$ is a permutation of $\mathfrak{i}$. For any such $\mathfrak{i}$, choose some $\sigma_\mathfrak{i} \in \mathfrak{S}_k$ s.t. $sc(\mathfrak{i}) = \sigma_\mathfrak{i}(\mathfrak{i})$, and define ε to be the function

$$r \mapsto \left(f(r)\ ,\ H^{-1}_{f(r)}(r)\right)$$

where $H_\mathfrak{i}$ is the automorphism on R corresponding to the permutation $\sigma_\mathfrak{i}^{-1}$ in (6). That $\pi_1\varepsilon = f$ is trivial, and that $j\pi_2\varepsilon = j$ follows from diagram 6. The inverse of ε is the function $(\mathfrak{i}, r) \mapsto H_\mathfrak{i}(r)$:

$$r \quad\mapsto\quad \left(f(r), H^{-1}_{f(r)}(r)\right) \quad\mapsto\quad H_{f(r)}H^{-1}_{f(r)}(r) = r$$

and

$$(\mathfrak{i}, r) \quad\mapsto\quad H_\mathfrak{i}(r) \quad\mapsto\quad \left(fH_\mathfrak{i}(r), H^{-1}_{fH_\mathfrak{i}(r)}H_\mathfrak{i}(r)\right) = (\mathfrak{i}, r)$$

where the equality follows from

$$fH_\mathfrak{i}(r) \quad=\quad \sigma_\mathfrak{i}^{-1}f(r) \quad=\quad \sigma_\mathfrak{i}^{-1}sc(\mathfrak{i}) \quad=\quad \sigma_\mathfrak{i}^{-1}\sigma_\mathfrak{i}(\mathfrak{i}) \quad=\quad \mathfrak{i}\ .$$

Because $\hat{s}$ is a section of $\hat{c}$, this ψ is unique: if $\psi'\hat{c} = \phi$, we have $\psi' = \psi'\hat{c}\hat{s} = \phi\hat{s}$. This concludes the proof.

□

Laboratoire de Mathématiques, CNRS UMR 5126 – Université de Savoie, 73376 Le Bourget-du-Lac Cedex, France
E-mail address: pierre.hyvernat@univ-savoie.fr
URL: http://lama.univ-savoie.fr/~hyvernat/