

Models of linear dependent type theory

Martin Lundfall

October 17, 2017

Abstract

The role that types play in type theory can be seen as a simultaneous generalisation of the concept of a set, proposition and data type. In linear type theory, this view can be extended by interpreting linear types as a particular kind of resource that is *consumed* in the derivation. In this paper, we construct a type theory which deals with non-linear, dependent types (what we will call *cartesian*), and linear types, where both constructs are allowed to depend on terms of cartesian types. In the interplay between the cartesian and linear types we develop two new type formers, $\Box_{x:A} B$ and $\sqsubset_{x:A} B$, akin to Π and Σ , but where the dependent type B (and therefore the resulting construct) is linear. The !-exponential from linear logic is deconstructed into two modalities, M and L , which turns linear types into cartesian types and vice versa. We construct a semantic framework where the type theory is interpreted by constructing a base category of cartesian contexts equipped with two fibrations, whose fibers over a context Γ give rise to the categories of linear and cartesian types in Γ . The operators M and L give rise to a fiberwise adjunction between these.

Contents

1	TODO	3
2	Introduction	3
3	Preliminaries	3
3.1	Dependent type theory	3
3.2	Linear type theory	3
3.3	Fibrations	3
3.3.1	Grothendieck fibrations	3
3.3.2	Other types of fibrations	4
3.4	Enriched and higher categories	6
4	Syntax	8
4.1	Structural rules	8
4.2	Cartesian typing rules	9
4.3	Linear typing rules	11
4.4	Linear-Cartesian interplay	12
4.4.1	Linear dependent types	15
4.5	Universes	18
5	Semantics	19
5.1	Structural semantic core	19
5.2	Semantic type formers	22
6	Models	27
6.1	Set indexed families	27
6.1.1	Concrete examples	30
6.1.2	Syntactic enriched categories	30
6.2	Diagrams in monoidal categories	31
6.2.1	Universes in the diagrams model	34
6.3	Higher linear dependent types	34
6.3.1	Univalent linear type theory	35
7	Ideas	36
7.1	Dependent ordered type theory	36

1 TODO

New =E2. Comprehension in diagrams Linear univalence Careful about soundness

2 Introduction

3 Preliminaries

The paper is largely split into two sections; syntax and semantics. Familiarity with basic category theoretic concepts like limits, adjoints, monoidal categories is assumed.

3.1 Dependent type theory

A *type* is a syntactic construction whose meaning and function stems from the ways in which it interacts with the derivation rules of a *type theory*. Approaching type theory from the perspective of logic, we can form a first approximation of the definition of a type as a generalization of a proposition and a set.

3.2 Linear type theory

Linear logic is a substructural logic in which the rules of weakening and contraction:

$$\frac{\Gamma, \Delta \vdash B}{\Gamma, A, \Delta \vdash B} \text{ Weak} \quad \frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B} \text{ Contr}$$

are not admissible.

In other words, assumptions cannot be freely assumed or dismissed; they must be used exactly once in the conclusion. In linear logic, we are inclined to think of a sequent $A_1, A_2, \dots, A_n \vdash B$ as modelling a function or process in which the assumptions A_1, A_2, \dots, A_n are *resources used* to yield B . As a first, toy example, we consider the chemical process of burning hydrogen:

Example 3.1. Consider the following primitive derivation rule:

$$O_2 \otimes H_2 \otimes H_2 \vdash H_2O \otimes H_2O$$

stating that given an oxygen molecule and two hydrogen molecules, burning yields two water molecules. If weakening was admissible, we would be able to assume an additional hydrogen molecule without changes the antecedent, which does not make sense under the resource interpretation of linear logic.

Just as intuitionistic logic naturally extends to (dependent) type theory under the slogan of propositions as types, linear logic can be extended to a linear type theory, where sequents are decorated with proof terms:

$$x_1 : A_1, x_2 : A_2, \dots, x_n : A_n \vdash t : B$$

and here linearity implies that the free variables of t are x_1, x_2, \dots, x_n , each appearing exactly once.

Interest in linear type theory stems from disparate sources. From the perspective of (classical) computer science, it can be used for modelling state and storage, with linear variables denoting resources like pointers or files [10], or even as a theoretical description of the resource handling of a blockchain [14]. In quantum physics, linear types respect the no cloning theorem of quantum states, and are also proposed as a syntax for quantum computation [4]. From a semantic perspective, linear type theory can be interpreted in a symmetric monoidal closed categories, generalising the structure of a cartesian closed categories in which (non-dependent) type theories are interpreted.

3.3 Fibrations

A general heuristic for modelling a mathematical object E dependent over another object B is to specify a *projection* morphism $p : E \rightarrow B$ subject to certain constraints. For example, a family of sets A_i indexed by the set I might equally well be understood as the set $E = \bigsqcup_{i \in I} A_i$ together with a morphism $p : E \rightarrow I$ such that $p^{-1}(i) = A_i$. This is the idea guiding the concept of a fibration. Depending on which kind of mathematical object we are dealing with, we impose various conditions on the projection $p : E \rightarrow B$ to be able to *lift* certain structure of the base B into *fibers* $E_b = p^{-1}(b)$ for $b \in B$.

3.3.1 Grothendieck fibrations

In the context of categories, the appropriate notion is that of a **Grothendieck fibration**, often just called **fibration**. In order for a functor $p : E \rightarrow B$ to be a fibration, one needs to be able to lift arrows in the base category B to arrows in E . We do this by asking for the existence of certain *cartesian arrows* in E .

Definition 3.2 (Cartesian arrow). Given a functor $p : E \rightarrow B$, an arrow $f : e' \rightarrow e''$ of E is said to be **cartesian** with respect to p if for every $h : e \rightarrow e''$ and $\alpha : p(e) \rightarrow p(e')$ such that $p(f)\alpha = p(h)$, there is a unique arrow $\hat{\alpha} : e \rightarrow e'$ such that $p(\hat{\alpha}) = \alpha$ and $f\hat{\alpha} = h$.

The situation is illustrated in the following diagrams:

$$\begin{array}{ccc} p(e) & & \\ \downarrow \forall \alpha & \searrow p(h) & \\ p(e') & \xrightarrow{p(f)} & p(e'') \end{array} \qquad \begin{array}{ccc} e & & \\ \downarrow !\exists \hat{\alpha} & \searrow \forall h & \\ e' & \xrightarrow{f} & e'' \end{array}$$

Definition 3.3 (Grothendieck fibration). A functor $p : E \rightarrow B$ is a **Grothendieck fibration** if, for every $e \in E$, $b \in B$ and arrow $f : b \rightarrow p(e)$ in B , there exists a cartesian arrow f^* in E such that $p(f^*) = f$.

We will refer to a cartesian arrow f^* for which $p(f^*) = f$ as a **lift** of f . Although lifts are not uniquely determined, from the condition of cartesianness their domain will be determined up to unique isomorphism. A basic review of Grothendieck fibrations particularly relevant to the semantics of dependent type theory can be found in [8]. We will simply repeat the basic notions that will be used in our investigations.

For any object $b \in B$ in the base, we call the subcategory of E which is mapped to b and its identity morphism the **fiber** over b . This will be denoted E_b . A fibration $p : E \rightarrow B$ induces for every $u : b \rightarrow b'$ in B a functor $u^* : E_{b'} \rightarrow E_b$ by sending every object to the domain of the cartesian arrow associated to u . Such functors will be unique up to unique natural isomorphism. In general, for two compatible morphisms u and v in the base, a lift of a composition is not identical to the composition of a lift, only naturally isomorphic. In other words, we have $u^* \circ v^* \cong (u \circ v)^*$, but not functoriality on the nose. Those fibrations for which there are lifts of morphisms such that the equalities hold on the nose are called **Split**.

Definition 3.4 (Cartesian functor). Let $p : E \rightarrow B$ and $q : E' \rightarrow B$ be fibrations over the same base. A functor $F : E \rightarrow E'$ is **cartesian** if $qF = p$ and cartesian morphisms in E with respect to p are mapped to cartesian morphisms in E' with respect to q .

This determines a category $\mathbf{Fib}(B)$, consisting of fibrations over B and cartesian functors between them. More generally, one can construct a category \mathbf{Fib} whose morphisms from fibrations $p : E \rightarrow B$ and $q : E' \rightarrow B'$ are pairs of functors (F, G) where $F : E \rightarrow E'$ and $G : B \rightarrow B'$ such that $G \circ p = q \circ F$ and F preserves cartesian morphisms. In fact, the functor $\mathbf{Fib} \rightarrow \mathbf{Cat}$ sending a fibration to its base is itself a fibration whose fibers are $\mathbf{Fib}(B)$ for any small category B .

With the notion of a fibered natural transformation, \mathbf{Fib} forms a 2-category.

Definition 3.5 (Fibered natural transformation). For two pairs of parallel functors F, H and G, J between fibrations (E, B) and (E', B') , as illustrated in the commutative square:

$$\begin{array}{ccc} E & \begin{array}{c} \xrightarrow{F} \\ \xrightarrow{G} \end{array} & E' \\ \downarrow p & & \downarrow p' \\ B & \begin{array}{c} \xrightarrow{H} \\ \xrightarrow{J} \end{array} & B' \end{array}$$

a **Fibered natural transformation** (λ, λ') between (F, H) and (G, J) is a pair of natural transformations $\lambda : G \rightarrow F$ and $\lambda' : A \rightarrow B$ such that $p'(\lambda) = p_{\lambda'}$.

Notice that this definition does not ask for F and G to be cartesian functors. But when that is the case, fibered natural transformations between the parallel morphisms (F, A) and (G, A) in \mathbf{Fib} are the 2-morphisms of this 2-category.

An important special case of this is when $B = B'$ and $H = J = 1_B$. Then a fibered natural transformation $\lambda : F \rightarrow G$ is simply a natural transformation such that all of its components are sent to identities via p' . Such a natural transformation is sometimes called *vertical*.

Definition 3.6. Let $p : E \rightarrow B$ and $q : D \rightarrow B$ be fibrations over the same base and $F : E \rightarrow D$ and $G : D \rightarrow E$ cartesian functors with respect to these. F is called a **fibered left adjoint** of G if $F \dashv G$ in the usual way and the unit η of the adjunction is vertical. Such an adjunction will be called a **fiber adjunction**.

3.3.2 Other types of fibrations

In exploring models of linear and dependent types, fibrations of other structures will arise. Two important examples will be fibrations of groupoids and fibrations of monoidal categories.

Definition 3.7. A map $p : G \rightarrow H$ in \mathbf{Grpd} is a **fibration of groupoids** if for every $g \in G$ and $f : p(g) \rightarrow h$ in H , there exists an object g' and map $\hat{f} : g \rightarrow g'$ in G such that $p(g') = h$ and $p(\hat{f}) = f : p(g) \rightarrow p(g')$.

When considering fibrations of monoidal categories, we distinguish between the case where both the fibration and the base are monoidal categories, and when we simply want each fiber to be a monoidal category and the induced functors between these to be monoidal functors. The former notion is that of a monoidal fibration:

Definition 3.8. A **monoidal fibration** is a functor $\Phi : E \rightarrow B$ such that

- Φ is a Grothendieck fibration
- E and B are monoidal categories and Φ is a strict monoidal functor, and
- the tensor product of E preserves cartesian arrows.

In particular, when B is a cartesian monoidal category, an arrow $f : b \rightarrow p(e)$ induces a strong monoidal functor $f^* : B_e \rightarrow B_{f^*e}$ between the fibers [16].

The more general structure of a lax monoidal fibration [18], does not require neither E nor B to be monoidal. We simply want each fiber of E to carry a monoidal structure, and that the induced functors between fibers are monoidal.

Definition 3.9. A **lax monoidal fibration** is a fibration $p : E \rightarrow B$ along with

1. Two functors $\otimes : E \times_B E \rightarrow E$ and $I : B \rightarrow E$ fitting into the following diagram:

$$\begin{array}{ccccc} E \times_B E & \xrightarrow{\otimes} & E & \xleftarrow{I} & B \\ & \searrow & \downarrow p & \swarrow 1_B & \\ & & B & & \end{array}$$

2. Three fibered natural isomorphisms α, λ and ρ associated with the diagrams:

$$\begin{array}{ccc} E \times_B E \times_B E & \xrightarrow{1_E \times_B \otimes} & E \times_B E \\ \downarrow \otimes \times_B 1_E & \nearrow \alpha & \downarrow \otimes \\ E \times_B E & \xrightarrow{\otimes} & E \end{array}$$

and

$$\begin{array}{ccccc} B \times_B E & \xrightarrow{I \times_B 1_E} & E \times_B E & \xleftarrow{1_E \times I} & E \times_B B \\ & \searrow \pi_2 & \downarrow \otimes & \swarrow \pi_1 & \\ & & E & & \end{array}$$

$\lambda : \pi_2 \Rightarrow \otimes$ $\rho : \otimes \Rightarrow \pi_1$

3. such that α, λ and ρ satisfies the pentagon and triangle identities in each fiber.

4. for every $b \in B$, $\rho_{I_b} = \lambda_{I_b}^{-1} : I_b \otimes I_b \rightarrow I_b$

These conditions are sufficient for each fiber to be a monoidal category and for the induced functors between fibers to be (lax) monoidal [18].

Example 3.10. An example of a fibration that is lax monoidal but not monoidal is the fibration $\mathbf{Gph} \rightarrow \mathbf{Set}$ taking a directed graph defined by $(V, E, dom, cod : E \rightarrow V)$ to its underlying set of vertices, V . For two graphs $\mathcal{A} = (A, O, dom_A, cod_A)$ and $\mathcal{B} = (B, O, dom_B, cod_B)$ over the same fiber \mathbf{Gph}_O , we define their tensor product by:

$$\mathcal{A} \otimes \mathcal{B} = (A \times_O B, cod_A \circ \pi_1, dom_B \circ \pi_2)$$

where $A \times_O B$ is the pullback in the following diagram:

$$\begin{array}{ccc} A \times_O B & \xrightarrow{\pi_2} & B \\ \downarrow \pi_1 & & \downarrow cod_B \\ A & \xrightarrow{dom_A} & O \end{array}$$

This tensor product is only defined over graphs with the same underlying set, i.e. living in the same fiber.

3.4 Enriched and higher categories

The concept of a category can be generalized in a few different directions. In exploring semantics of linear type theory, our first encounter with such structures will be enriched categories, where hom-sets are generalized to “hom-objects” of a monoidal category \mathcal{V} .

Definition 3.11. Recall that a functor $F : \mathcal{C} \rightarrow \mathcal{D}$ is **monoidal** if there exists a natural transformation $\mu : F \times F \rightarrow F$ with components $\mu_{A,B} : F(A) \otimes_{\mathcal{D}} F(B) \rightarrow F(A \otimes_{\mathcal{C}} B)$ and a morphism $I_{\mathcal{D}} \rightarrow F(I_{\mathcal{C}})$. Satisfying the following naturality conditions:

1. **(Associativity)** For all objects $x, y, z \in \mathcal{C}$ the following diagram commutes:

$$\begin{array}{ccc}
 (F(x) \otimes_{\mathcal{D}} F(y)) \otimes_{\mathcal{D}} F(z) & \xrightarrow{a_{F(x), F(y), F(z)}^{\mathcal{D}}} & F(x) \otimes_{\mathcal{D}} (F(y) \otimes_{\mathcal{D}} F(z)) \\
 \downarrow \mu_{x,y} \otimes id & & \downarrow id \otimes \mu_{y,z} \\
 F(x \otimes_{\mathcal{C}} y) \otimes_{\mathcal{D}} F(z) & & F(x) \otimes_{\mathcal{D}} (F(y \otimes_{\mathcal{C}} z)) \\
 \downarrow \mu_{x \otimes_{\mathcal{C}} y, z} & & \downarrow \mu_{x, y \otimes_{\mathcal{C}} z} \\
 F((x \otimes_{\mathcal{C}} y) \otimes_{\mathcal{C}} z) & \xrightarrow{F(a_{x,y,z}^{\mathcal{C}})} & F(x \otimes_{\mathcal{C}} (y \otimes_{\mathcal{C}} z))
 \end{array}$$

where $a^{\mathcal{C}}$ and $a^{\mathcal{D}}$ denote the associators of the monoidal categories;

2. **(Unitality)** For all $x \in \mathcal{C}$ the following diagrams commutes:

$$\begin{array}{ccc}
 1_{\mathcal{D}} \otimes_{\mathcal{D}} F(x) & \xrightarrow{\epsilon \otimes id} & F(1_{\mathcal{C}}) \otimes_{\mathcal{D}} F(x) \\
 \downarrow \ell_{F(x)}^{\mathcal{D}} & & \downarrow \mu_{1_{\mathcal{C}}, x} \\
 F(x) & \xleftarrow{F(\ell_x^{\mathcal{C}})} & F(1 \otimes_{\mathcal{C}} x)
 \end{array}$$

and

$$\begin{array}{ccc}
 F(x) \otimes_{\mathcal{D}} 1_{\mathcal{D}} & \xrightarrow{id \otimes \epsilon} & F(x) \otimes_{\mathcal{D}} F(1_{\mathcal{C}}) \\
 \downarrow r_{F(x)}^{\mathcal{D}} & & \downarrow \mu_{x, 1_{\mathcal{C}}} \\
 F(x) & \xleftarrow{F(r_x^{\mathcal{C}})} & F(x \otimes_{\mathcal{C}} 1)
 \end{array}$$

where $\ell^{\mathcal{C}}, \ell^{\mathcal{D}}, r^{\mathcal{C}}, r^{\mathcal{D}}$ denote the left and right unitors of the two monoidal categories, respectively.

Definition 3.12. For a monoidal category \mathcal{V} , a \mathcal{V} -**enriched category** \mathcal{C} , consists of the following:

1. A set \mathcal{C}_0 of objects,
2. for each pair $a, b \in \mathcal{C}_0$, an object $V_{a,b} \in \mathcal{V}$,
3. for every $a, b, c \in \mathcal{C}_0$, a *composition law*, i.e. a functor $M_{a,b,c} : V_{b,c} \otimes V_{a,b} \rightarrow V_{a,c}$ and
4. an identity element $j_a : I \rightarrow V_{a,a}$

such that for all a, b, c, d , the following associativity and identity diagrams commute:

$$\begin{array}{ccc}
 (V_{c,d} \otimes V_{b,c}) \otimes V_{a,b} & \xrightarrow{\alpha} & V_{c,d} \otimes (V_{b,c} \otimes V_{a,b}) \\
 \downarrow M_{b,c,d} \otimes 1_{V_{a,b}} & & \downarrow 1_{V_{c,d}} \otimes M_{a,b,c} \\
 V_{b,d} \otimes V_{a,b} & & V_{c,d} \otimes V_{a,c} \\
 \searrow M_{a,b,d} & & \swarrow M_{a,c,d} \\
 & V_{a,d} & \\
 \\
 I \otimes V_{a,b} & \xrightarrow{l} & V_{a,b} \xleftarrow{r} V_{a,b} \otimes I \\
 \downarrow j_b \otimes 1_{V_{a,b}} & \nearrow M_{a,b,b} & \downarrow 1_{V_{a,b}} \otimes j_a \\
 V_{b,b} \otimes V_{a,b} & & V_{a,b} \otimes V_{a,a} \\
 & \nwarrow M_{a,a,b} & \nearrow M_{a,a,b}
 \end{array}$$

where α, l and r are the associator, and left and right unitor isomorphisms associated with the monoidal structure of \mathcal{V} .

Note that for the special case of $\mathcal{V} = \mathbf{Set}$ we get back the definition of a category. Plenty of examples of constructions in enriched categories can be found in the comprehensive introduction *Basic concepts of enriched category theory* by Max Kelly, [9].

Another generalization of a category is suggested by the direction in which categories generalize sets/classes by allowing arrows between objects. Thinking of arrows as one-dimensional objects between zero-dimensional objects, one can imagine the existence of 2-dimensional arrows between 1-dimensional ones. This leads to the notion of a 2-category. These come in two different forms, strict 2-categories (or just 2-categories) or weak 2-categories (also known as bicategories). Equipped with the notion of an enriched category, strict 2-categories can be defined concisely as:

Definition 3.13. A (strict) **2-category** is a *Cat*-enriched category.

Breaking down this definition provides a more illuminating view. A strict 2-category, \mathcal{C} , consists of

- a collection of objects \mathcal{C}_0 ,
- for any pair of objects $a, b \in \mathcal{C}_0$, a category $\mathcal{C}_{a,b}$, whose objects will be called “1-morphisms” and whose morphisms are renamed “2-morphisms”.
- for any object $a \in \mathcal{C}_0$, an “identity” functor $1_a : \mathbf{1} \rightarrow \mathcal{C}_{a,a}$,
- for any triple of objects $a, b, c \in \mathcal{C}_0$, a functor $Comp : \mathcal{C}_{b,c} \times \mathcal{C}_{a,b} \rightarrow \mathcal{C}_{a,c}$ satisfying the associativity and identity diagrams from the definition of enriched categories.

The action of $Comp$ on 1-morphisms is called “horizontal composition” and is written $gf := Comp(g, f)$ for $f \in \mathcal{C}_{a,b}$ and $g \in \mathcal{C}_{b,c}$, whereas the composition between 2-morphisms $\alpha : f \Rightarrow g$, $\alpha' : g \Rightarrow h$ for $f, g, h \in \mathcal{C}_{a,b}$ is written $\alpha' \circ \alpha$. Thanks to the functoriality of $Comp$, these satisfy the following *interchange law*:

$$(\beta' \circ \beta)(\alpha' \circ \alpha) = (\beta' \alpha') \circ (\beta \alpha)$$

The path from strict 2-categories to weak 2-categories is marched to the category theoretic slogan that it is undesirable to speak of equality between objects in a category. 2-morphisms give us a way of relating 1-morphisms up to isomorphism instead of on-the-nose equality. Thus we are inclined to loosen the restriction of the associativity and unital diagrams in the definition of a 2-category to only commute up to coherent natural isomorphism:

Definition 3.14. A **weak 2-category** \mathcal{C} is a collection of objects, 1-morphisms and 2-morphisms with composition and identity functors as before, such that for all $a, b, c, d \in \mathcal{C}_0$, there exists natural isomorphisms

$$\begin{aligned} \gamma : M_{a,b,d} \circ (M_{b,c,d} \otimes 1_{V_{a,b}}) &\Rightarrow M_{a,c,d} \circ (1_{V_{c,d}} \otimes M_{a,b,c}) \circ a \\ \lambda : l &\Rightarrow M_{a,b,b} \circ (j_b \circ 1_{V_{a,b}}) \\ \rho : r &\Rightarrow M_{a,a,b} \circ (1_{V_{a,b}} \circ j_a) \end{aligned}$$

in other words, associated with the diagrams:

$$\begin{array}{ccc} (V_{c,d} \otimes V_{b,c}) \otimes V_{a,b} & \xrightarrow{\alpha} & V_{c,d} \otimes (V_{b,c} \otimes V_{a,b}) \\ \downarrow M_{b,c,d} \otimes 1_{V_{a,b}} & & \downarrow 1_{V_{c,d}} \otimes M_{a,b,c} \\ V_{b,d} \otimes V_{a,b} & \xrightarrow{\gamma} & V_{c,d} \otimes V_{a,c} \\ \searrow M_{a,b,d} & & \swarrow M_{a,c,d} \\ & V_{a,d} & \end{array}$$

$$\begin{array}{ccccc} I \otimes V_{a,b} & \xrightarrow{l} & V_{a,b} & \xleftarrow{r} & V_{a,b} \otimes I \\ \downarrow j_b \otimes 1_{V_{a,b}} & \Downarrow \lambda & \uparrow M_{a,b,b} & \Downarrow \rho & \downarrow 1_{V_{a,b}} \otimes j_a \\ V_{b,b} \otimes V_{a,b} & & & & V_{a,b} \otimes V_{a,a} \end{array}$$

subject to the following coherence conditions:

In section 6.3, we will consider the category of small symmetric monoidal categories, which is itself a monoidal category. That is, for two symmetric monoidal categories \mathcal{A} and \mathcal{B} , there is a symmetric category $\mathcal{A} \otimes \mathcal{B}$. We will briefly outline the idea behind the construction of this category and its monoidal structure. The full details, as well as many other results about the monoidal structure of **SMCat**, can be found in [15].

Definition 3.15. For $\mathcal{A}, \mathcal{B} \in \mathbf{SMCat}$, let their **tensor product category** be the symmetric monoidal category $\mathcal{A} \otimes \mathcal{B}$ whose objects are the words generated by the constant unit symbol I and the 2-ary symbol \otimes and the language of $Ob(\mathcal{A}) \times Ob(\mathcal{B})$. In other words, $Ob(\mathcal{A} \otimes \mathcal{B})$ is generated inductively from:

- $I \in \text{Ob}(\mathcal{A} \otimes \mathcal{B})$, and for $a \in \mathcal{A}$ and $b \in \mathcal{B}$, $a \otimes b \in \mathcal{A} \otimes \mathcal{B}$
- If X and Y are in $\mathcal{A} \otimes \mathcal{B}$, then $X \otimes Y \in \mathcal{A} \otimes \mathcal{B}$.

The arrows of $\mathcal{A} \otimes \mathcal{B}$ are generated by the following:

- A natural isomorphism of associativity, with components $\text{ass}_{X,Y,Z} : (X \otimes Y) \otimes Z \rightarrow X \otimes (Y \otimes Z)$.
- Natural isomorphism l and r , with components $l_X : X \rightarrow I \otimes X$ and $r_X : X \rightarrow X \otimes I$.
- Natural isomorphism s , with components $s_{X,Y} : X \otimes Y \rightarrow X \otimes Y$, satisfying $s_{X,Y} \circ s_{Y,X} = 1_{Y,X}$.
- For each $b \in \mathcal{B}$, a morphism $\alpha_b : I \rightarrow I_{\mathcal{A}} \otimes b$, natural in b .
- For each $a \in \mathcal{A}$, a morphism $\beta_a : I \rightarrow a \otimes I_{\mathcal{B}}$, natural in a .
- For any $f : b \rightarrow b'$ in \mathcal{B} , an arrow $a \otimes f : a \otimes b \rightarrow a \otimes b'$
- For any $f : a \rightarrow a'$ in \mathcal{A} , an arrow $f \otimes a : a \otimes b \rightarrow a' \otimes b$
- For X and $f : Y \rightarrow Z$, an arrow $X \otimes f : X \otimes Y \rightarrow X \otimes Z$.

subject to a variety of coherence conditions, outlined in [15].

To provide some intuition for this construction, compare it to the construction of the tensor product of abelian groups. The natural transformation γ , is essentially a categorification of the relation:

$$\begin{aligned} (a_1, b) + (a_2, b) &\sim (a_1 + a_2, b) \\ (a, b_1) + (a, b_2) &\sim (a, b_1 + b_2) \end{aligned}$$

in the presentation of the abelian tensor product as a quotient of the direct product of groups.

The tensor product of symmetric monoidal categories extends to a 2-functor $\text{Ten} : \mathbf{SMCat} \times \mathbf{SMCat} \rightarrow \mathbf{SMCat}$.

Definition 3.16. There is a symmetric monoidal category I , consisting of words generated by one letter, \star , where the tensor product is given by word concatenation and the empty word is the unit.

There is a small caveat due to the higher dimensional nature of this construction; the functor Ten and I do not technically form a monoidal structure on \mathbf{SMCat} . The associativity, left and right unitor and symmetries are all functors, $A : (\mathcal{A} \otimes \mathcal{B}) \otimes \mathcal{C} \rightarrow \mathcal{A} \otimes (\mathcal{B} \otimes \mathcal{C})$ equipped with a corresponding functor, $A' : (\mathcal{B} \otimes \mathcal{C}) \rightarrow (\mathcal{A} \otimes \mathcal{B}) \otimes \mathcal{C}$, but $A \circ A'$ is strictly the identity functor, only up naturally isomorphic to it. One might call this a lax symmetric monoidal 2-categorical structure on \mathbf{SMCat} .

However, by identifying 1-cells connected by a 2-cell, we get the quotient category $\mathbf{SMCat}/$, which has a symmetric closed monoidal structure.

As a final note, which will become important in section 6.3, one can easily see from the definition that if \mathcal{A} and \mathcal{B} are groupoids, their tensor product category will be a groupoid as well.

4 Syntax

The particular linear dependent type theory under consideration is inspired by the work of Krishnaswami in [10] and Vákár in [17]. Types are either *cartesian*, in which case we simply write $\Gamma \vdash A$ type, or *linear*, written $\Gamma \vdash A$ linear. When making typing judgements of linear terms, contexts will be split into two parts, separated by a semicolon. The first part contains *cartesian* assumptions, for which weakening and contraction is allowed, while the second part is the *linear* part, containing ephemeral assumptions that we are also inclined to think of as resources. The derivation rules for linear types will force the linear variables to occur exactly once in the conclusion. Dependent types are restricted to only depend on terms of cartesian types.

4.1 Structural rules

We will be dealing with the following judgements:

The basic structural rules for the linear dependent type theory are given in Figure 2. Omitted are the rules concerning judgemental equality, which specify that it is an equality relation which is congruent with respect to the other structural rules.

Judgement:

$\vdash \Gamma \text{ ctxt}$
 $\vdash \Gamma; \Xi \text{ ctxt}$
 $\Gamma \vdash A \text{ type}$
 $\Gamma \vdash A \text{ linear}$
 $\Gamma \vdash M : A$
 $\Gamma; \Xi \vdash M : A$
 $\Gamma \vdash A \equiv A' \text{ type}$
 $\Gamma \vdash A \equiv A' \text{ linear}$
 $\Gamma \vdash M \equiv N : A$
 $\Gamma; \Xi \vdash x \equiv y : A$

Meaning:

Γ is a well-formed cartesian context.
 $\Gamma; \Xi$ is a well-formed mixed context
 A is a cartesian type in Γ
 A is a linear type in Γ
 M is a term of the cartesian type A in Γ
 M is a (linear) term of the linear type A in $\Gamma; \Xi$
 A and A' are equal cartesian types in Γ
 A and A' are equal linear types in $\Gamma; \Xi$
 M and N are equal cartesian terms of A in Γ
 x and y are equal linear terms of A in $\Gamma; \Xi$

Figure 1: Judgements of linear dependent type theory

$\frac{}{\vdash \cdot \text{ ctxt}} \quad \text{CI-Base}$	$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, \Gamma'; \Xi \vdash t : A'}{\Gamma, x : A, \Gamma'; \Xi \vdash t : A'} \quad \text{Weak-L}$
$\frac{}{\vdash \cdot; \cdot \text{ ctxt}} \quad \text{CM-Base}$	$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, \Gamma' \vdash \mathcal{J}}{\Gamma, x : A, \Gamma' \vdash \mathcal{J}} \quad \text{Weak-I}$
$\frac{\Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A \text{ ctxt}} \quad \text{C-int-ext}$	$\frac{\Gamma \vdash M : A \quad \Gamma, x : A, \Gamma' \vdash \mathcal{J}}{\Gamma, \Gamma'[M/x] \vdash \mathcal{J}[M/x]} \quad \text{Int-subst-1}$
$\frac{\Gamma \vdash A \text{ type} \quad \vdash \Gamma, \Delta \text{ ctxt}}{\vdash \Gamma, x : A, \Delta \text{ ctxt}} \quad \text{C-weak-1}$	$\frac{\Gamma \vdash M : A \quad \Gamma, x : A, \Gamma'; \Xi \vdash t : A'}{\Gamma, \Gamma'[M/x]; \Xi[M/x] \vdash t : A'[M/x]} \quad \text{Int-subst-2}$
$\frac{\Gamma \vdash A \text{ type} \quad \vdash \Gamma, \Delta; \Xi \text{ ctxt}}{\vdash \Gamma, x : A, \Delta; \Xi \text{ ctxt}} \quad \text{C-weak-2}$	$\frac{\Gamma; \Xi, x : A \vdash t : B \quad \Gamma; \Xi' \vdash M : A}{\Gamma; \Xi, \Xi' \vdash t[M/x] : B} \quad \text{Lin-subst}$
$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma \vdash A \text{ linear}}{\vdash \Gamma; \Xi, x : A \text{ ctxt}} \quad \text{C-lin-ext}$	$\frac{\Gamma, x : A, \Gamma' \text{ ctxt}}{\Gamma, x : A, \Gamma' \vdash x : A} \quad \text{Int-var}$
$\frac{\Gamma; \Xi, x : A, y : B, \Xi' \vdash t : A'}{\Gamma; \Xi, y : B, x : A, \Xi' \vdash t : A'} \quad \text{Lin-exch}$	$\frac{\vdash \Gamma; x : A \text{ ctxt}}{\Gamma; x : A \vdash x : A} \quad \text{Lin-var}$

Figure 2: Structural rules

\mathcal{J} denotes a judgement of the form $A \text{ type}$, $A \text{ linear}$ or $M : A$ (for a cartesian type A).

4.2 Cartesian typing rules

The cartesian types that we will use are the standard Π , Σ and Id -types. For Σ and Id we will introduce an extra elimination and computational rule for the case where the type being eliminated into is linear.

$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \Pi_{x:A} B \text{ type}} \quad \Pi\text{-F}$	$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B[M/x]}{\Gamma \vdash (M, N) : \Sigma_{x:A} B} \quad \Sigma\text{-I}$
$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x. b : \Pi_{x:A} B} \quad \Pi\text{-I}$	$\frac{\Gamma, t : \Sigma_{x:A} B \vdash C \text{ type} \quad \Gamma, x : A, y : B \vdash c : C[(x, y)/t] \quad \Gamma \vdash s : \Sigma_{x:A} B}{\Gamma \vdash \hat{c}[s] : C[s/t]} \quad \Sigma\text{-E}_1$
$\frac{\Gamma \vdash t : \Pi_{x:A} B \text{ type} \quad \Gamma \vdash M : A}{\Gamma \vdash t(M) : B[M/x]} \quad \Pi\text{-E}$	$\frac{\Gamma, t : \Sigma_{x:A} B \vdash C \text{ linear} \quad \Gamma, x : A, y : B; \Xi \vdash c : C[(x, y)/t] \quad \Gamma \vdash s : \Sigma_{x:A} B}{\Gamma; \Xi[pr_1(s)/x][pr_2(s)/y] \vdash \hat{c}[s] : C[s/t]} \quad \Sigma\text{-E}_2$
$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash M : A}{\Gamma \vdash \lambda x. b(M) \equiv b[M/x] : B[M/x]} \quad \Pi\text{-C}$	$\frac{\Gamma \vdash \hat{c}[(a, b)] : C[(a, b)/t]}{\Gamma \vdash \hat{c}[(a, b)] \equiv c[(a, b)/t] : C[(a, b)/t]} \quad \Sigma\text{-C}_1$
$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \Sigma_{x:A} B \text{ type}} \quad \Sigma\text{-F}$	$\frac{\Gamma; \Xi \vdash \hat{c}[(a, b)] : C[(a, b)/t]}{\Gamma; \Xi \vdash \hat{c}[(a, b)] \equiv c[(a, b)/t] : C[(a, b)/t]} \quad \Sigma\text{-C}_2$
$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash 1 \text{ type}} \quad 1\text{-F}$	$\frac{\Gamma \vdash \hat{c}[\star] : C[\star/x]}{\Gamma \vdash \hat{c}[\star] \equiv c[\star/x] : C[\star/x]} \quad 1\text{-C}$
$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash \star : 1} \quad 1\text{-I}$	
$\frac{\Gamma, x : 1 \vdash C \text{ type} \quad \Gamma \vdash c : C[\star/x] \quad \Gamma \vdash M : 1}{\Gamma \vdash \hat{c}[M] : C[M/x]} \quad 1\text{-E}$	

Figure 3: (Cartesian) dependent sum and product types

$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : A}{\Gamma \vdash M =_A N \text{ type}} \quad =\text{-F}$	$\frac{\vdash \Gamma, x, y : A, p : x =_A y; \Xi \text{ ctxt} \quad \Gamma, x, y : A, p : x =_A y \vdash C \text{ linear} \quad \Gamma, z : A; \Xi[z/x, z/y, \text{refl}(z)/p] \vdash c : C[z/x, z/y, \text{refl}(z)/p] \quad \Gamma \vdash M : A \quad \Gamma \vdash N : A \quad \Gamma \vdash P : M =_A N}{\Gamma; \Xi[M/z, N/z, P/z] \vdash R_{[x,y,p]}^{Id}(c, M, N, P) : C[M/x, N/y, P/p]} \quad =\text{-E}_2$
$\frac{\Gamma \vdash M : A}{\Gamma \vdash \text{refl}(M) : M =_A M} \quad =\text{-I}$	$\frac{\Gamma \vdash R_{[x,y,p]}^{Id}(c, M, M, \text{refl}(M)) : C[M/x, M/y, \text{refl}(M)/p]}{\Gamma \vdash R_{[x,y,p]}^{Id}(c, M, M, \text{refl}(M)) \equiv c[M/z] : C[M/x, M/y, \text{refl}(M)/p]} \quad =\text{-C}_1$
$\frac{\Gamma, x, y : A, p : x =_A y \vdash C \text{ type} \quad \Gamma, z : A \vdash c : C[z/x, z/y, \text{refl}(z)/p] \quad \Gamma \vdash M : A \quad \Gamma \vdash N : A \quad \Gamma \vdash P : M =_A N}{\Gamma \vdash R_{[x,y,p]}^{Id}(c, M, N, P) : C[M/x, N/y, P/p]} \quad =\text{-E}_1$	$\frac{\Gamma; \Xi[M/x, M/y, \text{refl}(M)/p] \vdash R_{[x,y,p]}^{Id}(c, M, M, \text{refl}(M)) : C[M/x, M/y, \text{refl}(M)/p]}{\Gamma; \Xi[M/x, M/y, \text{refl}(M)/p] \vdash R_{[x,y,p]}^{Id}(c, M, M, \text{refl}(M)) \equiv c[M/z] : C[M/x, M/y, \text{refl}(M)/p]} \quad =\text{-E}_3$

Figure 4: (Cartesian) identity type

4.3 Linear typing rules

Perhaps the most important linear types are the \otimes - and I -types, as they will provide an interpretation of linear contexts. Semantically, we will not distinguish between the context $\Xi \equiv x_1 : A_1, x_2 : A_2, \dots, x_n : A_n$ and $x_1 \otimes x_2 \otimes \dots \otimes x_n : A_1 \otimes A_2 \otimes \dots \otimes A_n$.

$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash A \otimes B \text{ linear}} \quad \otimes\text{-F}$	$\frac{}{\Gamma \vdash I \text{ linear}} \quad I\text{-F}$
$\frac{\Gamma; \Xi \vdash a : A \quad \Gamma; \Xi' \vdash b : B}{\Gamma; \Xi, \Xi' \vdash (a, b) : A \otimes B} \quad \otimes\text{-I}$	$\frac{}{\Gamma; \cdot \vdash * : I} \quad I\text{-I}$
$\frac{\Gamma; \Xi' \vdash t : A \otimes B \quad \Gamma; \Xi, x : A, y : B \vdash c : C}{\Gamma; \Xi, \Xi' \vdash \text{let } x, y \text{ be } t \text{ in } c : C} \quad \otimes\text{-E}$	$\frac{\Gamma; \Xi \vdash c : C \quad \Gamma; \Xi' \vdash t : I}{\Gamma; \Xi, \Xi' \vdash \text{let } t \text{ be } * \text{ in } c : C} \quad I\text{-E}$
$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (a, b) \text{ in } c : C}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (a, b) \text{ in } c \equiv c[a/x][b/y] : C} \quad \otimes\text{-C}$	$\frac{\Gamma; \Xi \vdash \text{let } * \text{ be } * \text{ in } c : C}{\Gamma; \Xi \vdash \text{let } * \text{ be } * \text{ in } c \equiv c : C} \quad I\text{-C}$

Figure 5: Linear \otimes and I type formers

The typing rules for the remaining linear rules are standard.

$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash A \multimap B \text{ linear}} \multimap\text{-F}$	$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash A \oplus B \text{ linear}} \oplus\text{-F}$
$\frac{\Gamma; \Xi, x : A \vdash b : B}{\Gamma; \Xi \vdash \lambda x. b : A \multimap B} \multimap\text{-I}$	$\frac{\Gamma; \Xi \vdash a : A}{\Gamma \Xi \vdash \text{inl}(a) : A \oplus B} \oplus\text{-I}_1$
$\frac{\Gamma; \Xi \vdash f : A \multimap B \quad \Gamma; \Xi' \vdash a : A}{\Gamma; \Xi, \Xi' \vdash f(a) : B} \multimap\text{-E}$	$\frac{\Gamma; \Xi \vdash b : B}{\Gamma \Xi \vdash \text{inr}(b) : A \oplus B} \oplus\text{-I}_2$
$\frac{\Gamma; \Xi \vdash \lambda x. b(a) : B}{\Gamma; \Xi \vdash \lambda x. b(a) \equiv b[a/x] : B} \multimap\text{-C}$	$\frac{\Gamma; \Xi, x : A \vdash c : C \quad \Gamma; \Xi, y : B \vdash d : C; \Gamma; \Xi' \vdash t : A \oplus B}{\Gamma; \Xi, \Xi' \vdash \text{case } t \text{ of } \text{inl}(x) \rightarrow c \parallel \text{inr}(y) \rightarrow d : C} \oplus\text{-E}$
$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash A \& B \text{ linear}} \&\text{-F}$	$\frac{\Gamma; \Xi \vdash \text{case } \text{inl}(a) \text{ of } \text{inl}(x) \rightarrow c \parallel \text{inr}(y) \rightarrow d : C}{\Gamma; \Xi \vdash \text{case } \text{inl}(a) \text{ of } \text{inl}(x) \rightarrow c \parallel \text{inr}(y) \rightarrow d \equiv c[a/x] : C} \oplus\text{-C}_1$
$\frac{\Gamma; \Xi \vdash a : A \quad \Gamma; \Xi \vdash b : B}{\Gamma; \Xi \vdash \langle a, b \rangle : A \& B} \&\text{-I}$	$\frac{\Gamma; \Xi \vdash \text{case } \text{inr}(b) \text{ of } \text{inl}(x) \rightarrow c \parallel \text{inr}(y) \rightarrow d : C}{\Gamma; \Xi \vdash \text{case } \text{inr}(b) \text{ of } \text{inl}(x) \rightarrow c \parallel \text{inr}(y) \rightarrow d \equiv d[b/y] : C} \oplus\text{-C}_2$
$\frac{\Gamma; \Xi \vdash t : A \& B}{\Gamma; \Xi \vdash \text{fst}(t) : A} \&\text{-E}_1$	$\frac{}{\Gamma \vdash \top \text{ linear}} \top\text{-F}$
$\frac{\Gamma; \Xi \vdash t : A \& B}{\Gamma; \Xi \vdash \text{snd}(t) : B} \&\text{-E}_2$	$\frac{\vdash \Gamma; \Xi \text{ctxt}}{\Gamma; \Xi \vdash ! : \top} \top\text{-I}$
$\frac{\Gamma; \Xi \vdash \text{fst}(\langle a, b \rangle) : A}{\Gamma; \Xi \vdash \text{fst}(\langle a, b \rangle) \equiv a : A} \&\text{-C}_1$	$\frac{}{\Gamma \vdash 0 \text{ linear}} 0\text{-F}$
$\frac{\Gamma; \Xi \vdash \text{snd}(\langle a, b \rangle) : B}{\Gamma; \Xi \vdash \text{snd}(\langle a, b \rangle) \equiv b : B} \&\text{-C}_2$	$\frac{\Gamma; \Xi \vdash t : 0}{\Gamma; \Xi, \Xi' \vdash EFQ(t) : B} 0\text{-E}$

Figure 6: Linear \multimap , $\&$, \oplus , \top and 0 type formers

4.4 Linear-Cartesian interplay

We introduce two the modal operators M and L , which transfers a linear type/term to its cartesian counterpart and vice versa. Semantically, this will establish a fiberwise monoidal adjunction between the categories of linear and cartesian types:

$$\begin{array}{ccc} & L & \\ \mathcal{L}_\Gamma & \xleftarrow{\quad} & \mathcal{T}_\Gamma \\ & \perp & \\ & M & \end{array}$$

where the exponential modality from traditional linear logic is understood as the comonad $! = LM$. The decomposition of the exponential into an adjunction goes back to at least [1], and is given an interesting new light in [11], where it is seen as a particular case of a more general procedure of encoding structure in contexts.

Below are the syntactic rules for the operators M and L .

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A_L \text{ linear}} \quad \text{L-F}$	
$\frac{\Gamma \vdash a : A}{\Gamma; \cdot \vdash a_L : A_L} \quad \text{L-I}$	$\frac{\Gamma \vdash B \text{ linear}}{\Gamma \vdash B_M \text{ type}} \quad \text{M-F}$
$\frac{(\Gamma \vdash B \text{ linear}) \quad (\vdash \Gamma; \Xi' \text{ ctxt})}{\Gamma; \Xi \vdash y : A_L \quad \Gamma, x : A; \Xi' \vdash t : B} \quad \text{L-E}$	$\frac{\Gamma; \cdot \vdash b : B}{\Gamma \vdash b_M : B_M} \quad \text{M-I}$
$\frac{\Gamma; \Xi \vdash \text{let } x \text{ be } y \text{ in } t : B}{\Gamma; \Xi, \Xi' \vdash \text{let } x \text{ be } y \text{ in } t : B} \quad \text{L-E}$	$\frac{\Gamma \vdash t : B_M}{\Gamma; \cdot \vdash \sigma(t) : B} \quad \text{M-E}$
$\frac{\Gamma; \Xi \vdash \text{let } x \text{ be } s_L \text{ in } t : B}{\Gamma; \Xi \vdash \text{let } x \text{ be } s_L \text{ in } t \equiv t[s/x] : B} \quad \text{L-C}$	$\frac{\Gamma \vdash b_M : B_M}{\Gamma; \cdot \vdash \sigma(b_M) \equiv b : B} \quad \text{M-C}_1$
$\frac{\Gamma; y : A_L, \Xi \vdash t : B \quad (\Gamma, x : A; \Xi \vdash t[x_L/y] : B) \quad \Gamma; \Xi' \vdash a : A_L}{\Gamma; \Xi, \Xi' \vdash \text{let } x \text{ be } a \text{ in } t[x_L/y] \equiv t[a/y] : B} \quad \text{L-U}$	$\frac{\Gamma \vdash \sigma(t)_M : B_M}{\Gamma \vdash \sigma(t)_M \equiv t : B_M} \quad \text{M-C}_2$

As a motivation for the semantic interpretation of L and M as adjoint functors given in 5.18, we will demonstrate that they already satisfy the relevant conditions from a syntactic point of view. That is, in the spirit of functional programming, we think of the cartesian and linear types (in a context Γ) of our theory forming the respective categories \mathcal{T} and \mathcal{L} whose objects are types and morphisms are functions and composition is given by function composition. The identity function $\lambda x.x : A \rightarrow A$, yields the identity on cartesian types, while $\lambda x.x : B \multimap B$ is the identity for linear B .

Definition 4.1. Define $\hat{\circ}_- : (B \multimap C) \multimap (A \multimap B) \multimap (A \multimap C)$ and $\circ_- : (B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C)$ by:

$$\begin{aligned} g \hat{\circ} f &= \lambda x. g(f(x)) \\ g \circ f &= \lambda x. g(f(x)) \end{aligned}$$

Theorem 4.2. M is a (syntactic) functor. In other words, for any linear types, $\Gamma \vdash A$ linear and $\Gamma \vdash B$ linear, there is a term

$$\mathbf{fmap} : (A \multimap B)_M \rightarrow A_M \rightarrow B_M$$

such that $\mathbf{fmap} (id_A)_M \equiv id_{A_M}$ and $\mathbf{fmap} ((g \hat{\circ} f)_M) \equiv \mathbf{fmap} g_M \circ \mathbf{fmap} f_M$

Proof. Define:

$$\mathbf{fmap} f := \lambda a. (\sigma(f)\sigma(a))_M$$

and see that for $f = (id_A)_M$, we have $(\sigma(id_A)\sigma(a))_M \equiv (\sigma(a))_M \equiv a : A_M$, as well as:

$$\lambda x. (\sigma(\lambda y. g(f(y)))_M(\sigma x))_M \equiv \lambda x. (\lambda y. g(f(y))(\sigma x))_M \equiv \lambda x. (g(f(\sigma(x))))_M$$

□

Similarly, for L we have:

Theorem 4.3. L is a functor. i.e. for any cartesian types, A and B , there is a term $fmap : L(A \rightarrow B) \multimap (LA \multimap LB)$ such that $fmap L(id_A) \equiv id_{LA}$.

Proof. Define: (TODO: check again, permuted let and be terms?)

$$\mathbf{fmap} f a := \text{let } \hat{f} \text{ be } f \text{ in } (\text{let } a \text{ be } y \text{ in } f(y)_L)$$

where $\hat{f} : A \multimap B$ and $y : A$. When $f = (id_A)_L$ we get:

$$\mathbf{fmap} (id_A)_L a = \text{let } a \text{ be } y \text{ in } id_A(y)_L$$

which by L-U is equal to $(id_A)_L$.

□

Furthermore, the categories of cartesian and monoidal types carry a natural monoidal structure, given by $(\times, 1)$ and (\otimes, I) , respectively, where \times is simply Σ where the second argument does not depend on the first. M and L are monoidal functors with respect to these.

As our current syntactic endeavors mainly serve as motivation for the semantic interpretation of these operators, given in 5.18, we will not prove the complete statement here. We omit the naturality conditions and only prove the following:

Theorem 4.4. *There are terms t_1, t_2, t_3, t_4 , relating the operators M and L so that the judgements:*

$$\begin{aligned} \Gamma, x : 1 \vdash t_1 : I_M \\ \Gamma, x : A_M \times B_M \vdash t_2 : (A \otimes B)_M \\ \Gamma; x : I \vdash t_3 : 1_L \\ \Gamma; x : A_L \otimes B_L \vdash t_4 : (A \times B)_L \end{aligned}$$

hold.

Proof. t_1 is given by the following expression:

$$\Gamma, x : 1 \vdash *_M : I_M$$

and t_2 can easily be derived:

$$\Gamma, x : A_M \times B_M \vdash (\sigma(pr_1(x)) \otimes \sigma(pr_2(x)))_M : (A \otimes B)_M$$

t_3 is given by:

$$\Gamma; x : I \vdash \text{let } x \text{ be } * \text{ in } \star_L : 1_L$$

t_4 is a bit more involved, as it requires two applications of L-E. For clarity, we display the complete proof tree of the term:

$$\begin{array}{c} \frac{\Gamma, x : A, y : B \vdash x : A}{\Gamma, x : A, y : B \vdash y : B} \\ \frac{\Gamma, x : A, y : B \vdash (x, y) : A \times B}{\Gamma, x : A, y : B; \cdot \vdash (x, y)_L : (A \times B)_L} \quad \frac{\Gamma, x : A; y' : B_L \vdash y' : B_L}{\Gamma, x : A; y' : B_L \vdash \text{let } y \text{ be } y' \text{ in } (x, y)_L : (A \times B)_L} \quad \frac{\Gamma, x' : A_L \vdash x' : A_L}{\Gamma; x' : A_L, y' : B_L \vdash \text{let } x \text{ be } x' \text{ in } (\text{let } y \text{ be } y' \text{ in } (x, y)_L) : (A \times B)_L} \quad \frac{\Gamma; z : A_L \otimes B_L \vdash z : A_L \otimes B_L}{\Gamma; z : A_L \otimes B_L \vdash \text{let } (x', y') \text{ be } z \text{ in } (\text{let } x \text{ be } x' \text{ in } (\text{let } y \text{ be } y' \text{ in } (x, y)_L)) : (A \times B)_L} \\ \text{L-E} \quad \frac{\Gamma, x : A; y' : B_L \vdash \text{let } y \text{ be } y' \text{ in } (x, y)_L : (A \times B)_L}{\Gamma; x' : A_L, y' : B_L \vdash \text{let } x \text{ be } x' \text{ in } (\text{let } y \text{ be } y' \text{ in } (x, y)_L) : (A \times B)_L} \\ \otimes\text{-E} \quad \frac{\Gamma; x' : A_L, y' : B_L \vdash \text{let } x \text{ be } x' \text{ in } (\text{let } y \text{ be } y' \text{ in } (x, y)_L) : (A \times B)_L}{\Gamma; z : A_L \otimes B_L \vdash \text{let } (x', y') \text{ be } z \text{ in } (\text{let } x \text{ be } x' \text{ in } (\text{let } y \text{ be } y' \text{ in } (x, y)_L)) : (A \times B)_L} \end{array}$$

□

Finally, we show that M and L can be thought of as adjoint functors, in any context. In other words, we will show that there exists a “natural transformation” $\epsilon : LM \rightarrow 1$ satisfying the following universal property: For any $f : L(A) \rightarrow B$, there is a unique morphism $g : A \rightarrow B_M$, such that $\epsilon_B \circ L(g) = f$. Translated into the syntax of our type theory, the statement becomes the following:

Theorem 4.5 ($L \dashv M$). *There is a term $\Gamma; \beta_1 : B_{LM} \vdash \epsilon_B : B$ with the following property:*

For any term: $\Gamma; y : A_L \vdash f : B$, there is a unique term $\Gamma, x : A \vdash g : B_M$ such that $\Gamma; y : A_L \vdash \epsilon_B[\text{let } x \text{ be } y \text{ in } g_L / \beta_1] \equiv f : B$.

Proof. The counit $\epsilon : LM \rightarrow 1_{\mathcal{C}_\Gamma}$ is at any component B given by:

$$\begin{aligned} \Gamma, \beta_1 : B_{LM} \vdash \beta_1 : B_{LM} \\ \Gamma, \beta_2 : B_M; \cdot \vdash \sigma(\beta_2) : B \\ \Gamma; \beta_1 : B_{LM} \vdash \text{let } \beta_2 \text{ be } \beta_1 \text{ in } \sigma(\beta_2) : B \end{aligned}$$

where the last line is given by applying L-E to the first and third line.

For any $\Gamma; x : A_L \vdash f : B$, we get the corresponding g through:

$$\begin{aligned} \Gamma, x : A \vdash x : A \\ \Gamma, x : A; \cdot \vdash x_L : A \\ \Gamma, x : A; \cdot \vdash f[x_L / y] : B \\ \Gamma, x : A \vdash f[x_L / y]_M : B_M \end{aligned}$$

making Lg the term:

$$\Gamma; y : A_L \vdash \text{let } x \text{ be } y \text{ in } f[x_L / y]_{LM} : B_{LM}$$

our composite $\epsilon_B \circ Lg$ is given by substituting the above for β_1 in the term corresponding to ϵ_B , yielding:

$$\Gamma; y : A_L \vdash \text{let } \beta_2 \text{ be } (\text{let } x \text{ be } y \text{ in } f[x_L / y]_{LM}) \text{ in } \sigma(\beta_2) : B$$

Finally, if we substitute x_L for y in the above, we can rewrite the expression using L-C to:

$$\begin{aligned} \Gamma, x : A; \cdot \vdash \text{let } \beta_2 \text{ be } f_{LM} \text{ in } \sigma(\beta_2) : B &\equiv \\ \Gamma, x : A; \cdot \vdash \sigma(\beta_2)[f_M / \beta_2] &\equiv f : B \end{aligned}$$

so by L-U, we can transform this equality to the desired

$$\Gamma; y : A_L \vdash \text{let } \beta_2 \text{ be } (\text{let } x \text{ be } y \text{ in } f[x_L/y]_{LM}) \text{ in } \sigma(\beta_2) \equiv f : B$$

It remains to show that for any other term $\Gamma; x : A \vdash h : B_M$ such that $\epsilon_B \circ Lh = f$, we have $g = h$. Syntactically, $\epsilon_B \circ Lh = f$ corresponds to the judgement:

$$\Gamma; y : A_L \vdash \text{let } \beta_2 \text{ be } (\text{let } x \text{ be } y \text{ in } h_L) \text{ in } \sigma(\beta_2) \equiv f : B$$

If we weaken the cartesian context by $x : A$, we can substitute x_L for y and get:

$$\Gamma, x : A; \cdot \vdash \text{let } \beta_2 \text{ be } (\text{let } x \text{ be } x_L \text{ in } h_L) \text{ in } \sigma(\beta_2) \equiv \sigma(h) \equiv f[x_L/y] : B$$

finally, we apply M and get:

$$\Gamma, x : A \vdash \sigma(h)_M \equiv h \equiv f[x_L/y]_M : B$$

□

Since we are inclined to think of M as right adjoint, we should expect it to preserve limits. This is true, at least for product types:

Theorem 4.6. *For all linear types A and B in Γ , there is an isomorphism:*

$$\Gamma \vdash A_M \times B_M \cong (A \& B)_M$$

Proof. Construct a function $f : A_M \times B_M \rightarrow (A \& B)_M$ by:

$$f \equiv \lambda x. \left(\sigma(\text{pr}_1(x)), \sigma(\text{pr}_2(x)) \right)_M$$

and $g : (A \& B)_M \rightarrow A_M \times B_M$:

$$g \equiv \lambda y. \left((\text{fst}(\sigma(y)))_M, (\text{snd}(\sigma(y)))_M \right)$$

□

which are easily seen to be mutually inverse.

It is hard to reason about equality of linear terms since the ordinary Id -type is a dependent type, and in the theory we do not allow for type dependency on linear terms. In the presence of the M operator, however, we can form the type $a_M = b_M$ for two terms $\Gamma; \cdot \vdash a, b : A$, which can serve as a surrogate for linear equality. If we add extensionality to our theory so that propositional equality implies judgemental equality, then $a_M =_{A_M} b_M$ implies $\Gamma; \cdot \vdash a \equiv b : A$.

4.4.1 Linear dependent types

Since we allow linear types to depend on terms of cartesian types, we can form new versions of the Π - and Σ -types. We will denote these linear variants of Π - and Σ -types by \sqcap and \sqsubset , respectively.

$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ linear}}{\Gamma \vdash \sqcap_{x:A} B \text{ linear}} \quad \sqcap\text{-F}$		$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ linear}}{\Gamma \vdash \sqsubset_{x:A} B \text{ linear}} \quad \sqsubset\text{-F}$	
$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma, x : A; \Xi \vdash b : B}{\Gamma; \Xi \vdash \lambda x. b : \sqcap_{x:A} B} \quad \sqcap\text{-I}$		$\frac{\Gamma \vdash s : A \quad \Gamma; \Xi \vdash b : B[s/x]}{\Gamma; \Xi \vdash (s, b) : \sqsubset_{x:A} B} \quad \sqsubset\text{-I}$	
$\frac{\Gamma; \Xi \vdash t : \sqcap_{x:A} B \quad \Gamma \vdash a : A}{\Gamma; \Xi \vdash t(a) : B[a/x]} \quad \sqcap\text{-E}$		$\frac{\vdash \Gamma; \Xi' \text{ ctxt} \quad \Gamma \vdash C \text{ linear} \quad \Gamma; \Xi \vdash t : \sqsubset_{x:A} B \quad \Gamma, x : A; \Xi', y : B \vdash c : C}{\Gamma; \Xi, \Xi' \vdash \text{let } x, y \text{ be } t \text{ in } c : C} \quad \sqsubset\text{-E}$	
$\frac{\Gamma; \Xi \vdash \lambda x. b(a) : \sqcap_{x:A} B}{\Gamma; \Xi \vdash \lambda x. b(a) \equiv b[a/x] : B[a/x]} \quad \sqcap\text{-C}$		$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (s, t) \text{ in } c : C}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (s, t) \text{ in } c \equiv c[s/x][t/y] : C} \quad \sqsubset\text{-C}$	

An example of a \sqcap type, consider for any linear type A , the n -fold tensor product $A^n \equiv A \otimes A \dots A$, which we can define by induction over the natural numbers (using universes) via:

$$\begin{aligned} A^0 &\equiv I \\ A^{\text{suc}(n)} &\equiv A \otimes A^n \end{aligned}$$

with this construct, recall the example of burning hydrogen given in 3.1, now expressed as a linear function $\text{burn} : O_2 \otimes H_2 \otimes H_2 \multimap H_2O \otimes H_2O$. With a dependent, linear function type, we can generalize this process to the function:

$$\text{burn} : \sqcap_{n:\mathbb{N}} O_2^n \otimes H_2^{2n} \multimap H_2O^{2n}$$

An example of the \sqsubset type comes from Krishnaswami's treatment of linear dependent logic as a way to model imperative programs [10]. Here, a primitive type of memory locations, $\Gamma \vdash \text{Loc}$ type, is introduced, the terms x of which can reference a term of any cartesian type A by means of a term of the linear pointer type $[x \mapsto A]$:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : \text{Loc} \vdash [x \mapsto A] \text{ linear}}$$

Given a location $a : \text{Loc}$ and a term $t : A$ in a context Γ , one may allocate some memory at a , and create a pointer $a \mapsto t$ at a certain fixed cost, Ξ :

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash a : \text{Loc}}{\Gamma; \Xi \vdash a \mapsto t : [a \mapsto A]}$$

But if we want to reason more generally about our pointers, we might be interested in a more general pointer type, where the location variable is bound in a \sqsubset -type. This type is formed in any context Γ , for any type A :

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : \text{Loc} \vdash [x \mapsto A] \text{ linear}}{\Gamma \vdash \sqsubset_{x:A} [x \mapsto A] \text{ linear}}$$

with terms introduced by:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash a : \text{Loc} \quad \Gamma; \Xi \vdash t : [a \mapsto A]}{\Gamma; \Xi \vdash (a, t) : \sqsubset_{x:\text{Loc}} [x \mapsto A]}$$

Another example takes inspiration from the realm of blockchain technology, where one deals with resources like bitcoin or ether balances. Say we model ownership of ether by a public address (generated from a cryptographic public key) having the linear type $\text{addr}_1 : \text{ETH}$. Then we can model a decentralized mathematical prize committee, which give out a reward of 1 ether to whoever proves the Goldbach conjecture.

Since the reward is constant, this is not truly a dependent linear function and we abusively notate it by \multimap :

$$; x : \text{ETH} \vdash \text{GoldbachReward} : \Pi_{x:\mathbb{N}} \Sigma_{y,z:\mathbb{N}} \text{Prime}(y) \times \text{Prime}(z) \times y + z =_{\mathbb{B}} 2(\text{suc}(\text{suc}(x))) \multimap \text{ETH}$$

To execute such a function and receive the reward, one would need to exhibit that there is an address x with one ether to begin with and provide a proof of the Goldbach conjecture.

The sense in which \sqsubset and \sqsubset are "linear analogues" of Π and Σ can be formalized in the following way:

Theorem 4.7. *For all $\Gamma \vdash A$ type and $\Gamma, x : A \vdash B$ linear, there is an isomorphism:*

$$\Pi_{x:A} B_M \cong (\sqsubset_{x:A} B)_M$$

Proof. Construct the function

$$h := \lambda f. (\lambda x. \sigma(f(x)))_M : \Pi_{x:A} B_M \rightarrow (\sqsubset_{x:A} B)_M$$

and in the other direction:

$$h^{-1} := \lambda g. \lambda y. (\sigma(g)(y))_M : (\sqsubset_{x:A} B)_M \rightarrow \Pi_{x:A} B_M$$

which are mutually inverse:

$$\begin{aligned} h \circ h^{-1} &:= \lambda \alpha. \lambda f. (\lambda x. \sigma(f(x)))_M \left(\lambda g. \lambda y. (\sigma(g)(y))_M \right) (\alpha) \equiv \\ &\lambda \alpha. \lambda f. (\lambda x. \sigma(f(x)))_M (\lambda y. (\sigma(\alpha)(y))_M) \equiv \\ &\lambda \alpha. \left(\lambda x. \sigma \left((\lambda y. (\sigma(\alpha)(y))_M)(x) \right) \right)_M \equiv \\ &\lambda \alpha. \left(\lambda x. \sigma \left((\sigma(\alpha)(x))_M \right) \right)_M \equiv \\ &\lambda \alpha. (\lambda x. \sigma(\alpha)(x))_M \equiv \lambda \alpha. \alpha \\ h^{-1} \circ h &:= \lambda \alpha. (\lambda g. \lambda y. (\sigma(g)(y))_M) \left(\lambda f. (\lambda x. \sigma(f(x)))_M \right) (\alpha) \equiv \\ &\lambda \alpha. (\lambda g. \lambda y. (\sigma(g)(y))_M) (\lambda x. \sigma(\alpha)(x))_M \equiv \\ &\lambda \alpha. \lambda y. (\sigma \left((\lambda x. \sigma(\alpha)(x))_M \right) (y))_M \equiv \\ &\lambda \alpha. \lambda y. (\lambda x. \sigma(\alpha)(x))_M (y) \equiv \\ &\lambda \alpha. \lambda y. (\sigma(\alpha)(y))_M \equiv \\ &\lambda \alpha. \lambda y. \alpha(y) \equiv \lambda \alpha. \alpha \end{aligned}$$

□

Similarly, assuming some extra *commuting conversions* and uniqueness rules, we can display a linear isomorphism:

$$(\Sigma_{x:A} B)_L \cong \sqsubset_{x:A} B_L$$

By a linear isomorphism, we mean a pair of linear functions $j : (\Sigma_{x:A} B)_L \multimap \sqsubset_{x:A} B_L$ and $j^{-1} : \sqsubset_{x:A} B_L \multimap (\Sigma_{x:A} B)_L$ such that $j \circ j^{-1} = \text{id}$ and $j^{-1} \circ j = \text{id}$.

A common practice in linear type theory is to assume a collection of rules which allows permuting the eliminator terms of various type formers in a controlled manner. For example, following ??, we define *environments* recursively in the following way:

Definition 4.8. An **environment** $E[-]$, is an object defined recursively as follows:

$$\begin{aligned} E[-] := & \text{ let } * \text{ be } E[-] \text{ in } t \mid \text{ let } * \text{ be } t \text{ in } E[-] \mid t \otimes E[-] \mid E[-] \otimes t \mid \\ & \text{ let } x, y \text{ be } E[-] \text{ in } t \mid \text{ let } x, y \text{ be } t \text{ in } E[-] \mid \lambda x. E[-] \mid E[-]t \mid (E[-], t) \mid \\ & (t, E[-]) \mid tE[-] \mid (E[-])_L \mid \text{ let } x \text{ be } E[-] \text{ in } t \mid \text{ let } x \text{ be } t \text{ in } E[-] \end{aligned}$$

We can then consider the addition of the following commuting conversions:

- For environments $E[-]$:

$$\frac{\Gamma; \Xi \vdash \text{let } * \text{ be } t \text{ in } E[u] : A}{\Gamma; \Xi \vdash \text{let } * \text{ be } t \text{ in } E[u] \equiv E[\text{let } * \text{ be } t \text{ in } u]} \text{ CC-I}$$

- For environments $E[-]$ that do not bind x or y :

$$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } t \text{ in } E[u] : A}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } t \text{ in } E[u] \equiv E[\text{let } x, y \text{ be } t \text{ in } u] : A} \text{ CC-}\otimes/\sqsubset$$

- For environments $E[-]$ that do not bind x :

$$\frac{\Gamma; \Xi \vdash \text{let } x \text{ be } t \text{ in } E[u] : A}{\Gamma; \Xi \vdash \text{let } x \text{ be } t \text{ in } E[u] \equiv E[\text{let } x \text{ be } t \text{ in } u] : A} \text{ CC-L}$$

Unfortunately, since we cannot reasonably decide on a direction for these judgemental equalities, the addition of commuting conversions like these makes us unable to construct a term rewriting procedure for our type theory. Therefore, we will not assume them in general, but only for results like the following:

Theorem 4.9. Assuming the commuting conversions above, and the following uniqueness rule for \sqsubset :

$$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } t \text{ in } (x, y) : A}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } t \text{ in } (x, y) \equiv t : A} \sqsubset\text{-U}$$

there is an isomorphism

$$(\Sigma_{x:A} B)_L \cong \sqsubset_{x:A} B_L$$

Proof. We first construct the linear function $j : (\Sigma_{x:A} B)_L \multimap \sqsubset_{x:A} B_L$ by:

$$j := \lambda f. \text{let } p \text{ be } f \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) : (\Sigma_{x:A} B)_L \multimap \sqsubset_{x:A} B_L$$

and its inverse, j^{-1} , by:

$$\frac{\Gamma; q : \sqsubset_{x:A} B_L \vdash q : \sqsubset_{x:A} B_L \quad \frac{\Gamma, x : A; y : B_L \vdash y : B_L \quad \frac{\Gamma, x : A, z : B \vdash x : A \quad \Gamma, x : A, z : B \vdash z : B}{\Gamma, x : A, z : B \vdash (x, z) : \Sigma_{x:A} B}}{\Gamma, x : A, z : B; \cdot \vdash (x, z)_L : (\Sigma_{x:A} B)_L}}{\Gamma; q : \sqsubset_{x:A} B_L \vdash \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) : (\Sigma_{x:A} B)_L} \quad \frac{\Gamma; q : \sqsubset_{x:A} B_L \vdash \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) : (\Sigma_{x:A} B)_L}{\Gamma; \cdot \vdash \lambda q. \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) : \sqsubset_{x:A} B_L \multimap (\Sigma_{x:A} B)_L}$$

and see that:

$$\begin{aligned} j \circ j^{-1} & \equiv \lambda \alpha. \lambda f. \text{let } p \text{ be } f \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) (\lambda q. \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)) \alpha \equiv \\ & \lambda \alpha. \lambda f. \text{let } p \text{ be } f \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) (\text{let } x, y \text{ be } \alpha \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)) \equiv \\ & \lambda \alpha. \text{let } p \text{ be } (\text{let } x, y \text{ be } \alpha \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)) \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) \equiv (\text{CC-L}) \\ & \lambda \alpha. \text{let } x, y \text{ be } \alpha \text{ in } \left(\text{let } p \text{ be } (\text{let } z \text{ be } y \text{ in } (x, z)_L) \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) \right) \equiv (\text{CC-L}) \\ & \lambda \alpha. \text{let } x, y \text{ be } \alpha \text{ in } \left(\text{let } p \text{ be } (\text{let } z \text{ be } y \text{ in } (x, z))_L \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) \right) \equiv (\text{CC-L}) \\ & \lambda \alpha. \text{let } x, y \text{ be } \alpha \text{ in } \left(\text{let } p \text{ be } (x, \text{let } z \text{ be } y \text{ in } z)_L \text{ in } \left(\text{pr}_1(p), (\text{pr}_2(p))_L \right) \right) \equiv (\text{L-C} + \Sigma\text{-C}) \\ & \lambda \alpha. \text{let } x, y \text{ be } \alpha \text{ in } \left(x, (\text{let } z \text{ be } y \text{ in } z)_L \right) \equiv (\text{CC-L}) \\ & \lambda \alpha. \text{let } x, y \text{ be } \alpha \text{ in } \left(x, (\text{let } z \text{ be } y \text{ in } z_L) \right) \equiv (\text{L-U} + \sqsubset\text{-U}) \\ & \lambda \alpha. \alpha \end{aligned}$$

and in the other direction:

$$\begin{aligned}
j^{-1} \circ j &\equiv \lambda \alpha. \lambda q. \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) (\lambda f. \text{let } p \text{ be } f \text{ in } (\text{pr}_1(p), (\text{pr}_2(p))_L)) \alpha \equiv \\
&\lambda \alpha. \lambda q. \text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) (\text{let } p \text{ be } \alpha \text{ in } (\text{pr}_1(p), (\text{pr}_2(p))_L)) \equiv \\
&\lambda \alpha. \text{let } x, y \text{ be } (\text{let } p \text{ be } \alpha \text{ in } (\text{pr}_1(p), (\text{pr}_2(p))_L)) \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L) \equiv (\text{CC-L}) \\
&\lambda \alpha. \text{let } p \text{ be } \alpha \text{ in } (\text{let } x, y \text{ be } (\text{pr}_1(p), \text{pr}_2(p)_L) \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)) \equiv (\square\text{-C}) \\
&\lambda \alpha. \text{let } p \text{ be } \alpha \text{ in } (\text{let } z \text{ be } \text{pr}_2(p)_L \text{ in } (\text{pr}_1(p), z)_L) \equiv (\text{L-C}) \\
&\lambda \alpha. \text{let } p \text{ be } \alpha \text{ in } ((\text{pr}_1(p), \text{pr}_2(p))_L) \equiv (\Sigma\text{-U}) + (\text{L-U}) \\
&\lambda \alpha. \alpha
\end{aligned}$$

□

As outlined in section 5.2, the semantics of the type formers Π , \sqcap and Σ , \sqsubset are as left and right adjoints to reindexing functors respectively, which can by lemma 6.2 be constructed via limits and colimits, these results reflect the fact that the left adjoint L preserves colimits and the right adjoints M preserves limits.

4.5 Universes

Our cartesian and linear types live in separate universes, U and L . Since linear types are type checked in a purely cartesian context, the linear universe is a cartesian type. Our universes are given á la Tarski and are closed under all previously introduced type formers. Below, we only outline the closure over the type formers Π , \otimes and \sqcap , but this should be sufficient to give the idea. For more detail we refer to Hofmann's *Syntax and Semantics of Dependent types*, [6], or Krishnaswami's *Integrating Linear and Dependent types*, [10].

	$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash L \text{ type}} \quad \text{L-F}$
$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash U \text{ type}} \quad \mathcal{U}\text{-F}$	$\frac{\Gamma \vdash t : L}{\Gamma \vdash \text{El}(t) \text{ linear}} \quad \text{L-El-F}$
$\frac{\Gamma \vdash t : U}{\Gamma \vdash \text{El}(t) \text{ type}} \quad \text{U-El-F}$	$\frac{\Gamma \vdash t : L \quad \Gamma \vdash s : L}{\Gamma \vdash t \hat{\otimes} s : L} \quad \text{L-}\otimes$
$\frac{\Gamma \vdash t : U \quad \Gamma, x : \text{El}(t) \vdash v : U}{\Gamma \vdash \hat{\Pi}_{x:\text{El}(t)} v : U} \quad \text{U-}\Pi$	$\frac{\Gamma \vdash t : L \quad \Gamma \vdash s : L}{\text{El}(t \hat{\otimes} s) \equiv \text{El}(s) \otimes \text{El}(t)} \quad \text{L-}\otimes\text{-Ty}$
$\frac{\Gamma \vdash t : U \quad \Gamma, x : \text{El}(t) \vdash v : U}{\Gamma \vdash \text{El}(\hat{\Pi}_{x:\text{El}(t)} v) \equiv \Pi_{x:\text{El}(t)} \text{El}(v)} \quad \text{U-}\Pi\text{-Ty}$	$\frac{\Gamma \vdash t : U \quad \Gamma, x : \text{El}(t) \vdash s : L}{\Gamma \vdash \hat{\sqcap}_{x:\text{El}(t)} s : L} \quad \text{L-}\sqcap$
	$\frac{\Gamma \vdash t : U \quad \Gamma, x : \text{El}(t) \vdash s : L}{\Gamma \vdash \text{El}(\hat{\sqcap}_{x:\text{El}(t)} s) \equiv \sqcap_{x:\text{El}(t)} \text{El}(s)} \quad \text{L-}\sqcap\text{-Ty}$

As an example using universes, we can properly formulate the identity function of cartesian or linear types by:

$$\begin{aligned}
&\text{id} : \Pi_{A:\mathcal{U}} A \rightarrow A \\
&\text{id } A \ a \equiv a
\end{aligned}$$

or

$$\begin{aligned}
&\text{id} : \sqcap_{A:\mathcal{L}} A \multimap A \\
&\text{id } A \ a \equiv a
\end{aligned}$$

respectively.

5 Semantics

5.1 Structural semantic core

To explore the models of linear dependent type theory we begin by constructing a categorical structure which abstracts the key features of the theory. We will utilize the notion of a *comprehension category*, which provides the most general structure in which we can deal with the structural rules like context extensions and substitutions. Once this has been taken care of we may consider what extra conditions have to be imposed in order for the model to support various type constructors, and then provide concrete models that satisfy these conditions.

There are only two type constructors that will be assumed in the general semantic structure: the linear tensor product and unit. This simplifies the core semantics by allowing us to use symmetric monoidal categories instead of multicategories when interpreting linear types.

The idea behind the core of the semantics is to construct a comprehension category [8], consisting of a base category of contexts, a 'cartesian' fibration consisting of cartesian types equipped with context extensions, and a lax symmetric monoidal fibration consisting of linear types.

Definition 5.1. A **comprehension category** consists of a commutative diagram of functors

$$\begin{array}{ccc} \mathcal{T} & \xrightarrow{\pi} & \mathcal{C}^{\rightarrow} \\ & \searrow p & \downarrow \text{cod} \\ & & \mathcal{C} \end{array}$$

where B^{\rightarrow} is the arrow category of B and $\text{cod} : B^{\rightarrow} \rightarrow B$ denotes the codomain functor such that:

1. \mathcal{C} has a terminal object
2. $p : \mathcal{T} \rightarrow \mathcal{C}$ is a Grothendieck fibration,
3. $\pi : \mathcal{T} \rightarrow \mathcal{C}^{\rightarrow}$ takes cartesian morphisms in \mathcal{T} to cartesian morphisms in $\mathcal{C}^{\rightarrow}$

Notice that by the second condition, cartesian morphisms in \mathcal{T} are mapped to pullback squares in \mathcal{C} :

A cartesian morphism $(p, q) : f \rightarrow g$ in $\mathcal{C}^{\rightarrow}$ is a commutative square in \mathcal{C}

$$\begin{array}{ccc} B & \xrightarrow{g} & B' \\ \downarrow p & & \downarrow q \\ A & \xrightarrow{f} & A' \end{array}$$

such that for any $E', E, q' : E' \rightarrow B', e : E \rightarrow E'$ and $p' : E \rightarrow A$ as in the following diagram:

$$\begin{array}{ccc} E & \xrightarrow{e} & E' \\ \downarrow \text{id}_E & & \downarrow q' \\ E & \xrightarrow{q'e} & B' \\ \downarrow p' & & \downarrow q \\ A & \xrightarrow{f} & A' \end{array}$$

there is a unique arrow $u : E \rightarrow B$ such that $p' = p \circ u$ and $q'e = g \circ u$. This is precisely the universal property of the pullback. If \mathcal{C} has all pullbacks, then $\text{cod} : \mathcal{C}^{\rightarrow} \rightarrow \mathcal{C}$ is a fibration and we immediately have a comprehension category. Maps of \mathcal{C} that arise via $\pi(A)$ for some $A \in \mathcal{T}$ will be called *projections*, and by the remark above, pullbacks of projections always exist. Terms of a type will be interpreted as sections of the corresponding projections and pullbacks of \mathcal{C} will play the role of substitutions. Here we need to be careful, however:

Remark 5.2. On Coherence. A subtlety often overlooked in the construction of models of type theory are the various issues arising from the fact that from a categorical point of view, it is natural to consider objects up to isomorphism, whereas in type theory we speak of equality “on the nose” [5]. In particular, when interpreting substitution as a pullback, we need to take into account that, in the syntax, substitution is strictly functorial, while pullbacks, in general, are not. For example, the pullback of f along $g \circ h$ is *isomorphic to* the pullback of π_1 along h , below, but not necessarily equal to

it.

$$\begin{array}{ccc}
A \times_C B' \cong (A \times_C B) \times_C B' & \longrightarrow & B' \\
\downarrow & & \downarrow h \\
A \times_C B & \xrightarrow{\pi_1} & B \\
\downarrow \pi_2 & & \downarrow g \\
A & \xrightarrow{f} & C
\end{array}$$

This problem can generally be overcome by showing that there exists a suitable choice of pullbacks such that this isomorphism is an equality, and this is essentially why we are forcing our comprehension category to come equipped with liftings of every map in the base category, so that our fibration is *split*. In fact, every comprehension category is equivalent to a strict one, and if a full comprehension category supports various type formers, then these will also be supported in the corresponding split version. [12]

In dealing with linear dependent type theory, we run into a similar problem. Equipped with a (lax) monoidal fibration, $q : \mathcal{L} \rightarrow \mathcal{C}$, we will interpret linear contexts $\Xi = a_1 : A_1, a_2 : A_2 \dots a_n : A_n$ over a cartesian context Γ as the tensor product $[[\Xi]] = [[A_1]] \otimes [[A_2]] \dots \otimes [[A_n]]$ in the monoidal category \mathcal{L}_Γ . The weakening of a linear context Ξ , as given by the rule:

$$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A; \Xi \text{ ctxt}} \text{ C-Weak}$$

will be interpreted as the image of the object $[[\Xi]] \in \mathcal{L}_\Gamma$ under the functor $\pi_A^* : \mathcal{L}_\Gamma \rightarrow \mathcal{L}_{\Gamma.A}$ induced by the projection morphism $\pi_A : \Gamma.A \rightarrow \Gamma$ in the base. But in general, we do not have an equality of $\pi_A^*(A) \otimes \pi_A^*(B)$ and $\pi_A^*(A \otimes B)$. To accommodate this discrepancy, we introduce the notion of a split monoidal fibration:

Definition 5.3. A **split monoidal fibration** is a lax monoidal fibration $q : \mathcal{L} \rightarrow \mathcal{C}$, equipped with a cleavage $\{u^*, \bar{u}\}_{u \in \mathcal{C}}$, so that every lift u^* is a strict monoidal functor, $u^*v^* = (vu)^*$ and $1_\Gamma^* = 1_{\mathcal{L}_\Gamma}$ for for all compatible liftings u^* and v^* and objects $\Gamma \in \mathcal{C}$.

Notice that this condition is quite strong, as the induced functors of a lax monoidal fibration are generally only lax monoidal. This disqualifies the graph fibration ?? from constituting a model of our type theory. However, every lax monoidal fibration where the induced functors are strong will be equivalent to a split monoidal fibration. Equipped with these notions, we define the general structure of the structural core of our type theory:

A model for linear dependent type theory consists of a split comprehension category $p : \mathcal{T} \rightarrow \mathcal{C}$ and a split monoidal fibration $q : \mathcal{L} \rightarrow \mathcal{C}$, as illustrated in the following picture:

$$\begin{array}{ccccc}
& & \mathcal{T} & \xrightarrow{\pi} & \mathcal{C}^{\rightarrow} \\
& \searrow q & \downarrow p & \swarrow \text{cod} & \\
\mathcal{L} & & \mathcal{C} & &
\end{array}$$

Before showing how this works in detail, the following lemma will come in handy:

Lemma 5.4. The pullback of an arrow, g , has a section if g does. In other words, given a pullback of g :

$$\begin{array}{ccc}
A \times_C B & \xrightarrow{k} & B \\
\downarrow g^* & & \downarrow g \\
A & \xrightarrow{f} & C
\end{array}
\quad \left(\begin{array}{c} \curvearrowright \\ g \quad h \\ \curvearrowright \end{array} \right)$$

such that $gh = 1_C$, then h^* exists and is a section of g^* .

Proof. Since A with projections 1_A and hf forms a cone to the cospan, there exists a unique map $u : A \rightarrow A \times_C B$ such that $hfu = k$ and $g^*u = 1_A$, making u a section of g^* . \square

We prove soundness by defining an interpretation function by induction on the derivation rules of Figure 2.

Theorem 5.5 (Soundness). *A split comprehension category $p : \mathcal{T} \rightarrow \mathcal{C}$ together with a split monoidal fibration $q : \mathcal{L} \rightarrow \mathcal{C}$ is a model for the linear dependent type theory consisting of the structural rules presented in Figure 2.*

Proof. We construct an interpretation function $[[-]]$, which sends:

- Cartesian contexts Γ to objects of \mathcal{C} , considered up to definitional equality and renaming of bound variables.
- Linear contexts Ξ in Γ to objects of $\mathcal{L}_{[[\Gamma]]}$.
- Cartesian types A in Γ to objects of $\mathcal{T}_{[[\Gamma]]}$.
- Linear types B in Γ to objects of $\mathcal{L}_{[[\Gamma]]}$.

- Cartesian terms $M : A$ in Γ to sections of $\pi([A]) : [[\Gamma, A]] \rightarrow [[\Gamma]]$.
- Linear terms $b : B$ in $\Gamma; \Xi$ to morphisms $[[b]] : [[\Xi]] \rightarrow [[B]]$.

Proceeding by induction on the derivation rules, we will often abuse notation slightly and denote semantic objects the same as their syntactic counterparts.

- Case of CI-Base: $[[\cdot]]$ is the terminal object $\mathbf{1}$ of \mathcal{C} .
- Case of CM-Base: $[[\cdot; \cdot]]$ is the unit of \mathcal{L}_1 .
- Case of C-int-ext: By the induction hypothesis, we are given A in \mathcal{T}_Γ and need to display an object $\Gamma.A$ in \mathcal{C} . This object is the domain of the morphism that A is mapped to via π :

$$\Gamma.A \xrightarrow{\pi_A} \Gamma$$

- Case of C-lin-ext: Given objects $[[\Xi]]$ and $[[A]]$ in $\mathcal{L}_{[[\Gamma]]}$, the extended context $\Gamma; \Xi, x : A$ is interpreted as the object $[[\Xi]] \otimes [[A]] \in \mathcal{L}_{[[\Gamma]]}$.
- Case of C-weak-1: Given $A, \Delta \in \mathcal{T}_\Gamma$, we send Δ through the functor $\pi_A^* : \mathcal{T}_\Gamma \rightarrow \mathcal{T}_{\Gamma.A}$ induced by lifting the morphism $\pi_A : \Gamma.A \rightarrow \Gamma$. The resulting object $\pi_A^*(\Delta)$ will be sent to the context $p(\pi_A^*(\Delta)) = \Gamma.A.\Delta\{\pi_A\}$ via p , which is the interpretation of the conclusion of the rule C-weak-1.
- Case of C-weak-2: Notice that since π is a cartesian functor, the context above fits into the following pullback:

$$\begin{array}{ccc} \Gamma.A.\Delta\{\pi_A\} & \xrightarrow{q} & \Gamma.\Delta \\ \downarrow \pi_{\Delta\{\pi_A\}} & & \downarrow \pi_\Delta \\ \Gamma.A & \xrightarrow{\pi_A} & \Gamma \end{array}$$

so by lifting q to the monoidal functor $q^* : \mathcal{L}_{\Gamma.\Delta} \rightarrow \mathcal{L}_{\Gamma.A.\pi^*(\Delta)}$. The context we are looking for is the image of Ξ under this functor.

- Case of C-lin-ext: For objects Ξ and A in \mathcal{L}_Γ , we let their tensor product $\Xi \otimes A$ denote the extended context $\Gamma; \Xi, x : A$.
- Case of Lin-exch. Since our lax monoidal fibration is symmetric, we have $\Xi \otimes A \otimes B \otimes \Xi' \cong \Xi \otimes B \otimes A \otimes \Xi'$ in \mathcal{L}_Γ . Applying exchange to a judgement corresponds to composing with this isomorphism.
- Case of Weak-L. For any $A \in \mathcal{T}_\Gamma$ and morphism $t : \Xi \rightarrow A$ in $\mathcal{L}_{\Gamma, \Gamma'}$, we can transfer t along the functor $q_{A, \Gamma'}^* : \mathcal{L}_{\Gamma, \Gamma'} \rightarrow \mathcal{L}_{\Gamma.A.\Gamma'\{\pi_A\}}$ induced by the map $q_{A, \Gamma'}$ arising from the following pullback diagram:

$$\begin{array}{ccc} \Gamma.A.\Gamma'\{\pi_A\} & \xrightarrow{\quad} & \Gamma.\Gamma' \\ \downarrow q_{A, \Gamma'} & & \downarrow \pi'_{\Gamma} \\ \Gamma.A & \xrightarrow{\pi_A} & \Gamma \end{array}$$

to yield a morphism $q_{A, \Gamma'}^*(t) : q_{A, \Gamma'}^*(\Xi) \rightarrow q_{A, \Gamma'}^*(A')$.

- Case of Weak-I. The morphism $q_{A, \Gamma'}$ above also induces a functor $q_{A, \Gamma'}^* : \mathcal{T}_{\Gamma, \Gamma'} \rightarrow \mathcal{T}_{\Gamma.A.\Gamma'\{\pi_A\}}$.
- For Int-subst-1, the judgement \mathcal{J} can take three forms:

$$\mathcal{J} = B[M/x] \text{ type}$$

$$\mathcal{J} = b[M/x] : B[M/x]$$

$$\mathcal{J} = B[M/x] \text{ linear}$$

The corresponding categorical structure work by way of relating substitution in the theory to pullbacks in \mathcal{C} . We begin with showing that there is a type $B[M/x]$ in the context $\Gamma.\Gamma'[M/x]$. Suppose we are given the following objects of \mathcal{T} :

$$A \in \mathcal{T}_\Gamma$$

$$\Gamma' \in \mathcal{T}_{\Gamma.A}$$

$$B \in \mathcal{T}_{\Gamma.A.\Gamma'}$$

and a section:

$$M : \Gamma \rightarrow \Gamma.A$$

of the projection π_A . We first lift M to a cartesian arrow $M^* : \Gamma.\Gamma'\{M\} \rightarrow \Gamma'$, which is sent to the following pullback square by π :

$$\begin{array}{ccc} \Gamma.\Gamma'\{M\} & \xrightarrow{q_{(M,\Gamma')}} & \Gamma.A.\Gamma' \\ \downarrow \pi_{\Gamma'\{M^*\}} & & \downarrow \pi_{\Gamma'} \\ \Gamma & \xrightarrow{M} & \Gamma.A \end{array}$$

where $q_{(M,\Gamma')}$ is some map whose existence is given by the structure of the comprehension. We then lift $q_{(M,\Gamma')}$ to obtain a cartesian arrow $q_{(M,\Gamma')}^* : \Gamma.\Gamma'\{M^*\}.B\{q_{(M,\Gamma')}^*\} \rightarrow \Gamma.A.\Gamma'.B$ which fits in the following pullback diagram:

$$\begin{array}{ccc} \Gamma.\Gamma'\{M^*\}.B\{q_{(M,\Gamma')}^*\} & \xrightarrow{q_{(q_{(M,\Gamma')},B)}} & \Gamma.A.\Gamma'.B \\ \downarrow \pi_{B\{q_{(M,\Gamma')}^*\}} & & \downarrow \pi_B \\ \Gamma.\Gamma'\{M^*\} & \xrightarrow{q_{(M,\Gamma')}} & \Gamma.A.\Gamma' \end{array}$$

The element $\Gamma.\Gamma'\{M^*\}.B\{q_{(M,\Gamma')}^*\}$ of $\mathcal{T}_{\Gamma.\Gamma'\{M^*\}}$ along with its associated projection will be our interpretation of $\Gamma.\Gamma'[M/x] \vdash B[M/x]$.

Now suppose there is a section $b : \Gamma.A.\Gamma' \rightarrow \Gamma.A.\Gamma'.B$ of the projection π_B . To display an element of $B[M/x]$ is to give a section of $\pi_{B\{q_{(M,\Gamma')}^*\}}$. By lemma 5.4, we get such a section by pulling back b along $q_{(q_{(M,\Gamma')},B)}$.

Finally, if B is an object of $\mathcal{L}_{\Gamma.A.\Gamma'}$, then the image of B under the functor $q_{(M,\Gamma')}^* : \mathcal{L}_{\Gamma.A.\Gamma'} \rightarrow \mathcal{L}_{\Gamma.\Gamma'\{M^*\}}$ will be our interpretation of $B[M/x]$ as a linear type in the context $\Gamma, \Gamma[M/x]$.

- Case of Int-subst-2. The interpretation of Int-subst-2 is the image of t under $q_{(M,\Gamma')}^*$.
- Case of Lin-subst. Given morphisms $t : \Xi \otimes A \rightarrow B$ and $M : \Xi' \rightarrow A$ we get a morphism $t \circ (id_\Xi \otimes M) : \Xi \otimes \Xi' \rightarrow B$. Precomposing with the isomorphism $\otimes \Xi, \Xi' \cong \Xi \otimes \Xi'$ yields the desired morphism $\otimes \Xi, \Xi' \rightarrow B$.
- Case of Int-var. For any $A \in \mathcal{T}_\Gamma$, the pullback:

$$\begin{array}{ccc} \Gamma.A.A\{\pi_A\} & \xrightarrow{q} & \Gamma.A \\ \downarrow \pi_{A\{\pi_A\}} & & \downarrow \pi_A \\ \Gamma.A & \xrightarrow{\pi_A} & \Gamma \end{array}$$

is in fact (isomorphic to) the product $\Gamma.A \times \Gamma.A$, and so the diagonal map $\Delta : \Gamma.A \rightarrow \Gamma.A \times \Gamma.A$ forms a section to both projections. This map is thus the interpretation of the judgement $\Gamma, x : A \vdash x : A$.

- Case of Lin-var. The morphism corresponding to the linear variable term $\Gamma; x : A \vdash x : A$ is given by the identity morphism on A in \mathcal{L}_Γ .

□

5.2 Semantic type formers

In the following, we assume that the comprehension category comprising the core of our syntax is full, i.e. that the functor $\pi : \mathcal{T} \rightarrow \mathcal{C}^\rightarrow$ is full and faithful. This allows us to think of homsets $\mathcal{T}_\Gamma(A, B)$ as living over the base category projections $\mathcal{C}_\Gamma(\pi_A : \Gamma.A \rightarrow \Gamma, \pi_B : \Gamma.B \rightarrow \Gamma)$.

Definition 5.6. A **model of LDTT** is a full, split comprehension category $\pi : \mathcal{T} \rightarrow \mathcal{C}^\rightarrow$ and a split monoidal fibration $q : \mathcal{L} \rightarrow \mathcal{C}$.

It will sometimes be useful to also assume that the comprehension category has a *unit*, as this will give rise to an isomorphism $\mathcal{C}_\Gamma(\pi(A), \pi(B)) \cong \mathcal{T}_{\Gamma.A}(1, \pi_A^*(B))$ [8].

Definition 5.7. A fibration $p : \mathcal{T} \rightarrow \mathcal{C}$ **admits a terminal object** if there is a functor $\mathbf{1} : \mathcal{C} \rightarrow \mathcal{T}$, such that for every $\Gamma \in \mathcal{C}$, $\mathbf{1}(\Gamma)$ is a terminal object of the fiber \mathcal{T}_Γ , and for every $f : \Delta \rightarrow \Gamma$, the canonical map $\alpha : f^*(\mathbf{1}(\Gamma)) \rightarrow \mathbf{1}(\Delta)$ is an isomorphism.

As a result of this definition, the functor $\mathbf{1}$ is full and faithful.

Definition 5.8. A comprehension category $\pi : \mathcal{T} \rightarrow \mathcal{C}^\rightarrow$ **has a unit** if the fibration $p : \mathcal{T} \rightarrow \mathcal{C}$ admits a terminal object functor $\mathbf{1}$ which is left adjoint to the functor $s(\pi) : \mathcal{T} \rightarrow \mathcal{C}$, selecting the source of the morphism assigned by π .

Most of our models will naturally be equipped with such a structure, and this yields the nice property that $\mathcal{C}_{/\Gamma}(\Gamma.A, \Gamma.B) \cong \mathcal{T}_{\Gamma.A}(1, \pi_A^*(B))$, allowing us to shift in perspective to think of terms depending on Γ and A as either living in the slice category $\mathcal{C}_{/\Gamma}$ or in the fiber $\mathcal{T}_{\Gamma.A}$. Importantly, since $\mathbf{1}$ is full and faithful, the unit of the adjunction is a natural isomorphism, so for every $A \in \mathcal{T}_\Gamma$, we have $\eta_A : \Gamma.A \cong \Gamma.A.1$.

Definition 5.9. A model of LD TT **supports** *I*-types if, for every Γ , there exists an object $[[I]] \in \mathcal{L}_\Gamma$, equipped with a morphism $[[*]] : I \rightarrow [[I]]$, such that for every pair of maps $t : \Xi' \rightarrow [[I]]$ and $c : \Xi \rightarrow C$, there exists a map $[[\text{let } t \text{ be } * \text{ in } c]] : \otimes \Xi, \Xi' \rightarrow C$ such that $[[\text{let } * \text{ be } * \text{ in } c]] = c$.

This is trivially satisfied by every model by setting $[[I]] = I$, $[[*]] = 1_I$, and $[[\text{let } t \text{ be } * \text{ in } c]]$ to the composite $\rho(c \otimes [[t]])\alpha : \otimes \Xi, \Xi' \cong \Xi \otimes \Xi' \rightarrow C \otimes I \cong C$, where ρ is the right unitor for the monoidal structure of \mathcal{L}_Γ , and α is the isomorphism $\otimes \Xi, \Xi' \cong \Xi \otimes \Xi$. As for the identity $\rho(c \otimes 1_I)\alpha = c$, notice that $\alpha : \otimes \Xi \cong \Xi \otimes [\cdot] = \Xi \otimes I$ is given by ρ^{-1} , so we get $\rho(c \otimes 1_I) = c\rho$, from the naturality of ρ .

Definition 5.10. A model of LD TT **supports** \otimes -types, if, for every $A, B \in \mathcal{L}_\Gamma$, there exists an object $[[A \otimes B]] \in \mathcal{L}_\Gamma$, such that for any arrows $a : \Xi \rightarrow A$, $b : \Xi' \rightarrow B$, there exists an arrow $[[a \otimes b]] : \otimes \Xi, \Xi' \rightarrow [[A \otimes B]]$ and for arrows $t : \Xi' \rightarrow [[A \otimes B]]$ and $c : \otimes \Xi \otimes A \otimes B \rightarrow C$, there exists an arrow $[[\text{let } x \otimes y \text{ be } t \text{ in } c : C]]$ such that $[[\text{let } x \otimes y \text{ be } a \otimes b \text{ in } c]] = c$.

Again, it is not hard to see that there is a canonical interpretation of this in any model of LD TT sending $[[A \otimes B]]$ to $[[A]] \otimes [[B]]$. Some care is needed in order to always make sure contexts are left associated.

Similarly, one can define what it means for a model of LD TT to support $\multimap, \oplus, \&, \top$ and 0 . We may think of these conditions as corresponding to weak versions of internal homs, binary products and coproducts, and terminal and initial object of the fibers, that are stable under reindexing functors. Therefore, whenever the fibers of our model are monoidal closed, complete or cocomplete, we know that it supports the corresponding type formers. This is well established and not the focus of the interplay between linear and dependent types we explore here. For a more detailed treatment we refer the reader to [13].

What it means for a model of linear dependent type theory to *support* Π -types is directly inherited from the standard, non-linear case.

Definition 5.11. A model of LD TT **supports** Π -types if, for all $A \in \mathcal{T}_\Gamma$, the induced functor $\pi_A^* : \mathcal{T}_\Gamma \rightarrow \mathcal{T}_{\Gamma.A}$ has a right adjoint $\Pi_A : \mathcal{T}_{\Gamma.A} \rightarrow \mathcal{T}_\Gamma$ satisfying the following Beck-Chevalley condition: For all pullbacks in \mathcal{C} :

$$\begin{array}{ccc} \Gamma.E & \xrightarrow{q_{E,E'}} & \Delta.E' \\ \downarrow \pi_E & & \downarrow \pi_{E'} \\ \Gamma & \xrightarrow{f} & \Delta \end{array} \quad (1)$$

inducing the following functors between fibers:

$$\begin{aligned} q_{E,E'}^* &: \mathcal{T}_{\Delta.E'} \rightarrow \mathcal{T}_{\Gamma.E} \\ f^* &: \mathcal{T}_\Delta \rightarrow \mathcal{T}_\Gamma \\ \Pi_E &: \mathcal{T}_{\Gamma.E} \rightarrow \mathcal{T}_\Gamma \\ \Pi_{E'} &: \mathcal{T}_{\Delta.E'} \rightarrow \mathcal{T}_\Delta \end{aligned}$$

The canonical natural transformation $f^*\Pi_{E'} \rightarrow \Pi_E q_{E,E'}^*$ induced by the adjunction is a natural isomorphism.

The Beck-Chevalley condition is in effect saying that substitution commutes with the Π -type. For example, if $E' = E[\bar{t}/\bar{y}]$, with the map f^* representing the substitution $[\bar{t}/\bar{y}]$, then the Beck-Chevalley condition reads that for all $B \in \mathcal{T}_\Gamma$, we have $(\Pi_{x:E} B)[\bar{t}/\bar{y}] \cong \Pi_{x:E[\bar{t}/\bar{y}]} B[\bar{t}/\bar{y}]$.

As the rules Σ contains one more eliminator than usual ($\Sigma\text{-E}_2$ in Figure 3), one might wonder whether this requires additional conditions for the semantic type formers to ensure that these are well behaved with respect to the linear fibers. We will shortly see that this is not the case.

Definition 5.12. A model of LD TT **supports** Σ -types if it satisfies the following:

1. For all $A \in \mathcal{T}_\Gamma$, the induced functor $\pi_A^* : \mathcal{T}_\Gamma \rightarrow \mathcal{T}_{\Gamma.A}$ has a left adjoint, Σ_A ,
2. such that for all pullbacks (as in 1), these satisfy the Beck-Chevalley condition, i.e. the natural transformation: $\Sigma_E q^* \rightarrow f^* \Sigma_{E'}$ is a natural isomorphism, and
3. the induced map $\text{pair}_{A,B} : \Gamma.A.B \rightarrow \Gamma.\Sigma_A B$ is an isomorphism

The map $\text{pair}_{A,B} : \Gamma.A.B \rightarrow \Gamma.\Sigma_A B$ such that $\pi_{\Sigma_A B} \text{pair}_{A,B} = \pi_A \pi_B$, arises as the image of the unit $\eta : B \rightarrow \pi_A^*(\Sigma_A B)$ in \mathcal{T}_A under the comprehension functor $\pi : \mathcal{T} \rightarrow \mathcal{C}^\rightarrow$. The inverse of the pairing map will be denoted $(pr_1, pr_2) : \Gamma.\Sigma_A B \rightarrow \Gamma.A.B$.

This structure is sufficient to support new elimination rule ($\Sigma\text{-E}_2$):

Theorem 5.13. *If a model of LD TT supports Σ -types, then for every object $C \in \mathcal{L}_{\Gamma.\Sigma_A B}$ and morphism $c : \Xi \rightarrow C[\text{pair}_{A,B}]$ in $\mathcal{L}_{\Gamma.A.B}$ and section $s : \Gamma \rightarrow \Gamma.\Sigma_A B$, there exists a morphism $\hat{c}_s : \Xi\{(pr_1, pr_2)\} \rightarrow C\{s\}$ such that given sections $a : \Gamma \rightarrow \Gamma.A$ and $b : \Gamma.A \rightarrow \Gamma.A.B$, then $\hat{c}_{(a,b)} = c\{ba\} : \Xi\{ba\} \rightarrow C\{ba\}$.*

Proof. The situation is illustrated in the following diagram:

$$\begin{array}{ccc}
 & & (pr_1, pr_2) \\
 & \swarrow & \searrow \\
 \Gamma.A.B & \xrightarrow{pair_{A,B}} & \Gamma.\Sigma_A B \\
 \downarrow \pi_{A,B} & & \swarrow \pi_{\Sigma_A B} \\
 \Gamma & &
 \end{array}$$

Let $\hat{c}_s = c\{((pr_1, pr_2)s)\}$. First, this morphism has the correct target since we have

$$C\{((pr_1, pr_2)s)\}\{pair_{A,B}\} = C\{(pair_{A,B}(pr_1, pr_2)s)\} = C\{s\}$$

, relying on the fact our lax monoidal fibration is split. Secondly, we need to show that given sections $a : \Gamma \rightarrow \Gamma.A$ and $b : \Gamma.A \rightarrow \Gamma.A.B$, we have $c\{((pr_1, pr_2)(a, b))\} = c\{a\}\{b\}$:

$$\{(pr_1, pr_2)(a, b)\} = \{(pr_1, pr_2)pair_{A,B}ba\} = \{ba\} = \{a\}\{b\}$$

□

We may also speak of a model **supporting simple products**, which simply correspond to products in the fibers of \mathcal{T} which are stable under reindexing functors. These are of course implied if the model supports Σ -types. When it comes to Id-types, the situation is not as fortunate. If one wants to keep the theory intensional, we need to add condition (3) to make sure that the semantic identity types satisfy the added elimination and computation rule, $=-E_2$ and $=-C_2$ in Figure 4.

Definition 5.14 (Id-types). A model of LDTT **supports Id-types** if, for all $A \in \mathcal{T}_\Gamma$, there exists an object $Id_A \in \mathcal{T}_{\Gamma.A.A\{\pi_A\}}$ and a morphism $r_A : \Gamma.A \rightarrow \Gamma.A.A\{\pi_A\}.Id_A$ such that:

1. The following diagram commutes:

$$\begin{array}{ccc}
 \Gamma.A & & \\
 \downarrow r_A & \searrow v_A & \\
 \Gamma.A.A\{\pi_A\}.Id_A & \xrightarrow{\pi_{Id_A}} & \Gamma.A.A\{\pi_A\}
 \end{array}$$

2. For any commutative diagram:

$$\begin{array}{ccc}
 \Gamma.A & \longrightarrow & \Delta.C \\
 \downarrow r_A & & \downarrow \pi_C \\
 \Gamma.A.A\{\pi_A\}.Id_A & \longrightarrow & \Delta
 \end{array}$$

there exists a lift $J : \Gamma.A.A\{\pi_A\}.Id_A \rightarrow \Delta.C$ making the two triangles commute.

3. For any pair of objects, $C, \Xi \in \mathcal{L}_{\Gamma.A.A+.Id}$, sections $M, N : \Gamma \rightarrow \Gamma.A$, $P : \Gamma \rightarrow \Gamma.Id_A\{N^+\}\{M\}$, and morphism $c : \Xi\{r_A\} \rightarrow C\{r_A\}$, there exists a morphism $\hat{c}_{[M,N,P]} : \Xi\{P\}\{N^+\}\{M\} \rightarrow C\{P\}\{N^+\}\{M\}$ such that $\hat{c}_{[M,M,refl]} = c\{M\}$.

A simpler, but stronger condition, implying condition (3) in the intensional setting is that for every $A \in \mathcal{T}_\Gamma$, there is an adjunction $\pi_{A,Id}^* \dashv r^*$ between the fibers of \mathcal{L} . This yields for every $c : \Xi\{r\} \rightarrow C\{r\}$, a morphism $\hat{c} : \pi_{A,Id}^*(\Xi\{r\}) \rightarrow C\{r\}$, which, given sections $M, N : \Gamma \rightarrow \Gamma.A$ and $P : \Gamma.A.A^+ \rightarrow \Gamma.A.A^+.Id$, we can send to $\hat{c}\{M\}\{N\}\{P\} : \Xi\{M\}\{N\}\{P\} \rightarrow C\{M\}\{N\}\{P\}$.

If one were to work within an extensional type theory, where the rule:

$$\frac{\Gamma \vdash p : a =_A b}{\Gamma \vdash a \equiv b : A}$$

is added, then the third condition is not needed:

Theorem 5.15. A model of LDTT with extensional equality supports Id-types if, for every $A \in \mathcal{T}_\Gamma$, there are objects $Id_A \in \mathcal{T}_{\Gamma.A}$ satisfying conditions 1 and 2 above.

Proof. In extensional type theory, every identity term $\Gamma, x, y : A \vdash P : x =_A y$ can be shown to be equal to reflexivity by forming the identity type:

$$\Gamma, x, y : A, p : x =_A y \vdash p =_{x=A x} refl(x) \text{ type}$$

this is well typed, since by $\Gamma, x, y : A, p : x =_A y \vdash p : x =_A y$, we conclude the definitional equality $x \equiv y$, so the types $x =_A y$ and $x =_A x$ are definitionally equal. We can then inhabit the type $p =_{x=A x} refl(x)$ using the elimination principle from the equality $\Gamma, x : A, refl(x) : x =_A x \vdash refl(refl(x)) : refl(x) =_{x=A x} refl(x)$. Since we have inhabited the propositional equality, the equality holds judgementally by extensionality.

Since we identify judgementally equal terms in our semantics, this implies that any section of $\pi_{A,Id} : \Gamma.A.A^+.Id \rightarrow \Gamma.A$ must be equal to $r_A : \Gamma.A \rightarrow \Gamma.A.A^+.Id$. Condition 3 then reads rather trivially that for any morphism $c : \Xi \rightarrow C\{r_A\}$ and section $M : \Gamma \rightarrow \Gamma.A$, there should exist a morphism $\hat{c}_M : \Xi\{M\} \rightarrow C\{M\}\{r\}$ which we get by $\hat{c}_M = c\{M\}$. □

The semantic type formers for the linear dependent \sqcap and \sqcup is akin to that of Π and Σ . They are given by adjoints to the functors between fibers of \mathcal{L} induced by the projection maps in \mathcal{C} .

Definition 5.16. A model of LDTT **supports \sqcap -types** if, for all $A \in \mathcal{T}_\Gamma$, the induced monoidal functor $\pi_A^* : \mathcal{L}_\Gamma \rightarrow \mathcal{L}_{\Gamma.A}$ has a monoidal right adjoint, \sqcap_A satisfying the following Beck-Chevalley condition:
For all pullback squares in \mathcal{C} of the following form:

$$\begin{array}{ccc} \Gamma.E & \xrightarrow{q_{E,E'}} & \Delta.E' \\ \downarrow \pi_E & & \downarrow \pi_{E'} \\ \Gamma & \xrightarrow{f} & \Delta \end{array}$$

, the canonical natural transformation $f^* \sqcap_{E'} \rightarrow \sqcap_E q_{E,E'}^*$ is a natural isomorphism.

Definition 5.17. It **supports \sqcup -types** if, for all $A \in \mathcal{T}_\Gamma$, the functor every π_A^* has a monoidal left adjoint, satisfying the following:

1. (Beck-Chevalley): For all pullbacks squares as above, the natural transformation $\sqcup_E q^* \rightarrow f^* \sqcup_{E'}$ is a natural isomorphism.
2. (Frobenius reciprocity): For all objects $\Xi \in \mathcal{L}_\Gamma$ and $B \in \mathcal{L}_{\Gamma.A}$, the canonical morphism $\sqcup_A (\Xi \{ \pi_A \} \otimes B) \rightarrow \Xi \otimes \sqcup_A B$ is an isomorphism.

To illustrate why these conditions form the appropriate semantic counterpart to the derivation rules, we will outline the case for \sqcup in detail.

For every object $B \in \mathcal{L}_{\Gamma.A}$, we get a type $\sqcup_A B \in \mathcal{L}_\Gamma$, such that, given any section $s : \Gamma \rightarrow \Gamma.A$, and map $b : \Xi \rightarrow B\{s\}$ in \mathcal{L}_Γ , we get a map:

$$s^* \eta_B \circ b : \Xi \rightarrow B\{s\} \rightarrow s^* \pi_A^* (\sqcup_A B) = \sqcup_A B$$

where $\eta_B : B \rightarrow \pi^* (\sqcup_A B)$ is the unit of the adjunction. This map is the interpretation of the term (s, b) given by the introduction rule \sqcup -I.

To interpret the elimination rule, we are given maps $t : \Xi \rightarrow \sqcup_A B$ in \mathcal{L}_Γ and $c : \Xi' \{ \pi_A \} \otimes B \rightarrow C \{ \pi_A \}$ in $\mathcal{L}_{\Gamma.A}$. By the adjunction, there exists a map $\hat{c} : \sqcup_A (\Xi' \{ \pi_A \} \otimes B) \rightarrow C$ such that $\eta_{\Xi' \{ \pi_A \} \otimes B} \circ \pi_A^* \hat{c} = c$. By the Frobenius reciprocity condition, we have an isomorphism: $\beta_{\Xi', B} : \Xi' \otimes \sqcup_A B \rightarrow \sqcup_A (\Xi' \{ \pi_A \} \otimes B)$, and we get a map:

$$\hat{c} \beta : \Xi' \otimes \sqcup_A B \cong \sqcup_A (\Xi' \{ \pi_A \} \otimes B) \rightarrow C$$

which we can precompose with $1_{\Xi'} \otimes t$ to get a map from $\Xi' \otimes \Xi$ to C . We can then precompose with an appropriate isomorphism $\alpha' : \otimes (\Xi', \Xi) \rightarrow \Xi' \otimes \Xi$ to get a map from $\otimes \Xi', \Xi$ to C , which will be the interpretation of the elimination term from the rule \sqcup -E.

The computation rule boils down to showing that the following triangle commutes:

$$\begin{array}{ccc} \{s\}(\Xi' \{ \pi_A \} \otimes B) & & \\ \downarrow s^* \eta_{(\Xi' \{ \pi_A \} \otimes B)} & \searrow s^* c & \\ s^* \pi_A^* \sqcup_A (\Xi' \{ \pi_A \} \otimes B) = \sqcup_A (\Xi' \{ \pi_A \} \otimes B) & \xrightarrow{\hat{c}} & C \end{array}$$

which follows from applying the functor s^* to the equality $\eta_{\Xi' \{ \pi_A \} \otimes B} \circ \pi_A^* \hat{c} = c$.

Definition 5.18. A model of LDTT with unit and simple products **supports the operators M and L** if there exists functors $M : \mathcal{L} \leftrightarrow \mathcal{T} : L$ which are cartesian with respect to the fibrations $p : \mathcal{T} \rightarrow \mathcal{C}$ and $q : \mathcal{L} \rightarrow \mathcal{C}$, such that $L \dashv M$ is a fibred adjunction and there is an isomorphism of hom-sets:

$$\mathcal{L}_{\Gamma.A}(\pi_A^*(\Xi'), \pi_A^*(B)) \cong \mathcal{L}_\Gamma(LA \otimes \Xi', B)$$

Recall that a fibred adjunction implies that there are natural isomorphisms making the following diagram commute:

$$\begin{array}{ccc} \mathcal{L}_\Gamma & \begin{array}{c} \xleftarrow{L_\Gamma} \\ \xrightarrow{M_\Gamma} \end{array} & \mathcal{T}_\Gamma \\ \downarrow \pi_A^{*(\mathcal{L})} & & \downarrow \pi_A^{*(\mathcal{T})} \\ \mathcal{L}_{\Gamma.A} & \begin{array}{c} \xleftarrow{L_{\Gamma.A}} \\ \xrightarrow{M_{\Gamma.A}} \end{array} & \mathcal{T}_{\Gamma.A} \end{array}$$

which from a syntactic perspective ensures that M and L commute with substitution.

The final condition of the definition is what yields the elimination and computation rules L-U, and while it might appear somewhat unnatural semantically, it does turn out to hold automatically in a large class of models, due to the following result:

Theorem 5.19. *In a model of LDTT that supports \multimap type formers, then any fibred adjunction $L \dashv M$ where $L(1) \cong I$ satisfies $\mathcal{L}_{\Gamma.A}(\pi_A^*(\Xi'), \pi^*(B)) \cong \mathcal{L}_{\Gamma}(LA \otimes \Xi', B)$.*

Proof. A model supporting internal homs must have reindexing functions which preserve these. That is, we have an isomorphism $\pi_A^*[\Xi, B] \cong [\pi_A^*\Xi, \pi_A^*B]$. We get a chain of isomorphisms:

$$\begin{aligned} \mathcal{L}_{\Gamma}(LA \otimes \Xi, B) &\cong \\ \mathcal{L}_{\Gamma}(LA, [\Xi, B]) &\cong \\ \mathcal{T}_{\Gamma}(A, M_{\Gamma}[\Xi, B]) &\cong \\ \mathcal{T}_{\Gamma.A}(1, \pi_A^*(M_{\Gamma}[\Xi, B])) &\cong \\ \mathcal{T}_{\Gamma.A}(1, M_{\Gamma.A}\pi_A^*[\Xi, B]) &\cong \\ \mathcal{L}_{\Gamma.A}(L_{\Gamma.A}(1), \pi_A^*[\Xi, B]) &\cong \\ \mathcal{L}_{\Gamma.A}(I, \pi_A^*[\Xi, B]) &\cong \\ \mathcal{L}_{\Gamma.A}(I, [\pi_A^*\Xi, \pi_A^*B]) &\cong \\ \mathcal{L}_{\Gamma.A}(\pi_A^*\Xi, \pi_A^*B). \end{aligned}$$

□

I wonder if there is a more natural condition which does not assume \multimap -types that implies $\mathcal{L}_{\Gamma.A}(\pi_A^*(\Xi'), \pi^*(B)) \cong \mathcal{L}_{\Gamma}(LA \otimes \Xi', B)$, but I cannot find it right now.

There are various degrees to which a model can support universes. The baseline condition is an object U in every fiber such there is an object in the same fiber for every section of the corresponding projection. In order to even state what it means for a universe to be closed under various types, we need to assume that our model supports such types.

Definition 5.20. A model of LDTT **supports cartesian universes**, if for every $\Gamma \in \mathcal{C}$, there exists an object $U \in \mathcal{T}_{\Gamma}$, such that, for every section $t : \Gamma \rightarrow \Gamma.U$ to the corresponding projection, there exists an object $El(t) \in \mathcal{T}_{\Gamma}$. For a model that supports the type former in question, the universe is closed under:

- Π -types, (or Σ) if, for sections $t : \Gamma \rightarrow \Gamma.U$ and $v : \Gamma.El(t) \rightarrow \Gamma.El(t).U$, there is a section $\hat{\Pi}_{El(t)}v : \Gamma \rightarrow \Gamma.U$ such that $El(\hat{\Pi}_{El(t)}) = \Pi_{El(t)}El(v)$ (or Σ) i.e. if the object corresponding to $\hat{\Pi}_{El(t)}v$ is equal the image of $El(v)$ under the functor corresponding to the semantic type former.
- Id -types, if, for sections $A : \Gamma \rightarrow \Gamma.U$, $a, b : \Gamma \rightarrow \Gamma.El(A)$, there is a section $\hat{Id}(a, b) : \Gamma \rightarrow \Gamma.U$, such that $El(\hat{Id}(a, b)) = Id_A$.

Linear universes are defined in the same fashion, keeping in mind that the linear universe is a cartesian type.

Definition 5.21. A model of LDTT **supports linear universes**, if, for every $\Gamma \in \mathcal{C}$, there is an object $L \in \mathcal{T}_{\Gamma}$, such that for every section $t : \Gamma \rightarrow \Gamma.L$ to the corresponding projection, there exists an object $El(t)$ in \mathcal{L}_{Γ} . For a model that supports the type former in question, the linear universe is closed under this type if:

- I , (or \top , 0): There is a section $\hat{I} : \Gamma \rightarrow \Gamma.L$, such that $El(\hat{I}) = I$ in \mathcal{L}_{Γ} .
- \otimes (or \multimap , $\&$): For every pair of sections $a, b : \Gamma \rightarrow \Gamma.L$, there is a section $a \hat{\otimes} b$ (or $\hat{\multimap}$, $\hat{\&}$) such that $El(a \hat{\otimes} b) = El(a) \otimes El(b)$

Definition 5.22. Given a model supporting both cartesian and linear universes, these are closed under:

- M , if, for every section $x : \Gamma \rightarrow \Gamma.L$, there exists an element $\hat{M}(x) : \Gamma \rightarrow \Gamma.U$ such that $El(\hat{M}(x)) = M(El(x))$.
- L , if, for every section $x : \Gamma \rightarrow \Gamma.U$, there exists an element $\hat{L}(x) : \Gamma \rightarrow \Gamma.L$ such that $El(\hat{L}(x)) = L(El(x))$.
- \sqcap (or \sqcup), if, for sections $t : \Gamma \rightarrow \Gamma.U$ and $s : \Gamma.El(t) \rightarrow \Gamma.El(t).L$, there exists a section $\hat{\sqcap}_{El(t)}s : \Gamma \rightarrow \Gamma.L$ (or $\hat{\sqcup}$), such that $El(\hat{\sqcap}_{El(t)}s) = \sqcap_{El(t)}El(s)$.

6 Models

6.1 Set indexed families

Our first model will be based on the standard set-theoretic interpretation of dependent type theory [6]. The extension of this model to the linear realm is fairly straightforward, and provides a good springboard for examples to come. (It is also sufficient for showing that linear dependent type theory is a proper generalisation of both dependent type theory and linear type theory, as outlined in [17].)

Our linear and cartesian fibrations will both be constructed as fibrations of set indexed families:

Definition 6.1 ($Fam(\mathcal{C})$). For an arbitrary category \mathcal{C} , let $Fam(\mathcal{C})$ denote the category whose objects consists of pairs (S, f) where S is a set and f is a function $f : S \rightarrow Ob(\mathcal{C})$. Morphisms of $Fam(\mathcal{C})$ are pairs $(u, \alpha) : (S, f) \rightarrow (S', g)$ where $u : S \rightarrow S'$ and $\alpha : S \rightarrow Mor(\mathcal{C})$ such that $\alpha(s) : f(s) \rightarrow g(u(s))$ for all $s \in S$. Composition is given, for two compatible morphisms (u, α) and (v, β) , by:

$$(v, \beta) \circ (u, \alpha) = (v \circ u, g(u(s)) \circ f(s))$$

for all $s \in S$.

By projecting a family to its indexing set, we get a fibration $p : Fam(\mathcal{C}) \rightarrow \mathbf{Set}$, and if \mathcal{C} has a terminal object, \top such that the hom-sets $\mathcal{C}(\top, A)$ are small for all $A \in \mathcal{C}$, we can form a comprehension category with unit by defining $\pi : Fam(\mathcal{C}) \rightarrow \mathbf{Set}^{\rightarrow}$ as follows.

On objects, let

$$\pi(S, f) = fst : \{(s, t) \mid s \in S, t : \top \rightarrow f(s)\} \rightarrow S$$

For a morphism $(u, \alpha) : (S, f) \rightarrow (S', g)$, let

$$q_{(u, \alpha)} : \{(s, t) \mid s \in S, t : \top \rightarrow f(s)\} \rightarrow \{(s', t') \mid s' \in S', t' : \top \rightarrow g(s')\}$$

be defined by $q_{(u, \alpha)}(s, t) = (u(s), \alpha(s) \circ t)$. The functor π sends morphisms (u, α) to squares:

$$\begin{array}{ccc} \{(s, t) \mid s \in S, t : \top \rightarrow f(s)\} & \xrightarrow{q_{(u, \alpha)}} & \{(s', t') \mid s' \in S', t' : \top \rightarrow g(s')\} \\ \downarrow fst & & \downarrow fst \\ S & \xrightarrow{u} & S' \end{array}$$

in \mathbf{Set} . This comprehension category will be full if the global sections functor $\mathcal{C}(\top, -) \rightarrow \mathbf{Set}$ is full and faithful.

For a morphism $u : S \rightarrow p(S', f) = S'$ the canonical choice of a lift $u^* = (u, i) : (S, fu) \rightarrow (S', f)$ where $i : S \rightarrow Mor(\mathcal{C})$ is defined by $s \mapsto 1_{fu(s)}$ makes this a split fibration.

The cartesian part of our semantic structure will simply be the fibration $p : Fam(\mathbf{Set}) \rightarrow \mathbf{Set}$ with $\pi : Fam(\mathbf{Set}) \rightarrow \mathbf{Set}^{\rightarrow}$ as outlined above. Then for any symmetric monoidal category \mathcal{V} we form a lax monoidal fibration by letting the unit object be given at any fiber by mapping a set S to the family constant at the unit of \mathcal{V} , and define the tensor product as:

$$(S, g) \otimes (S, f) = (S, f \otimes g) = (S, \lambda s. f(s) \otimes g(s))$$

For $u : S \rightarrow p(S', f) = S'$ in the base, we have:

$$u^*((S', \alpha) \otimes (S', \beta)) = (S, \lambda s. \alpha u(s) \otimes \beta(u(s))) = u^*(S', \alpha) \otimes u^*(S', \beta)$$

Since the tensor product is defined pointwise, it is preserved by reindexing functors, so this is in fact a split monoidal fibration. Putting this together, this forms our first concrete model of the type theory:

$$\begin{array}{ccccc} Fam(\mathcal{V}) & & Fam(\mathbf{Set}) & \xrightarrow{\pi} & \mathbf{Set}^{\rightarrow} \\ & \searrow q & \downarrow p & \swarrow cod & \\ & & \mathbf{Set} & & \end{array}$$

This model trivially supports \otimes and I -types (all models do). For any fibration $p : Fam(\mathcal{C}) \rightarrow \mathbf{Sets}$, its fiberwise limits and colimits are constructed pointwise, so they exist if and only if they do in \mathcal{C} . Since reindexing functors is simply precomposition, they preserve these (co)limits. We therefore get that the families model supports the type formers $\oplus, 0, \&$ and \top if \mathcal{V} has binary coproducts, initial object, binary products and terminal object respectively. Similarly, it supports \multimap if \mathcal{V} is monoidal closed, by $[(S, f), (S, g)] = (S, \lambda s. [f(s), g(s)])$. This is a right adjoint to \otimes , since at every component, the set of maps $f(s) \otimes g(s) \rightarrow h(s)$ is isomorphic to $f(s) \rightarrow [g(s), h(s)]$.

For the dependent types, we utilize the following result:

Theorem 6.2. For a fibration $p : Fam(\mathcal{C}) \rightarrow \mathbf{Sets}$ and a function $u : S \rightarrow S'$ in the base, $u^* : Fam(\mathcal{C})_{S'} \rightarrow Fam(\mathcal{C})_S$ has a left (right) adjoints if and only if \mathcal{C} has small coproducts (products).

Proof. For \mathcal{C} with small coproducts, define the left adjoint to u^* by $(S, f) \mapsto (S', \lambda s'. \prod_{s \in u^{-1}(s')} f(s))$. Similarly for \mathcal{C} with small products, define a right adjoint to u^* by $(S, f) \mapsto (S', \lambda s'. \prod_{s \in u^{-1}(s')} f(s))$.

In the other direction, the set of objects we want to take the (co)product of forms a family $A = \{A_i\}_i$ in $Fam(\mathcal{C})$. The lift of the unique map $! : p(A) \rightarrow \mathbf{1}$ in the base has a right (left) adjoint, making $\Pi_!(A)$ an object of $Fam(\mathcal{C})_1 \cong \mathcal{C}$, so that for every object $B \in \mathcal{C}$, we have:

$$\mathcal{C}(B, \Pi_!(A)) \cong Fam(\mathcal{C})_S(!^*B, A)$$

where $!^*B = (S, \lambda s.B)$. This means that every map on the right hand side, which consists of a family of morphisms $f_i : B \rightarrow A_i$ gives rise to a unique map on the left hand side, which is precisely the universal property of the product. \square

Since reindexing functors of $Fam(\mathcal{C})$ are simply given by precomposition, adjoints to reindexing functors always satisfy Beck-Chevalley condition:

Theorem 6.3. *If every lift $f^* : Fam(\mathcal{C})_S \rightarrow Fam(\mathcal{C})_{S'}$ has a right adjoint G_{f^*} , defined as in Theorem 6.2, then, for all objects $A = (\Gamma, a)$ and pullbacks of the following form:*

$$\begin{array}{ccc} \Gamma'.f^*\{A\} & \xrightarrow{q_{A,f}} & \Gamma.A \\ \downarrow \pi_{f^*\{A\}} & & \downarrow \pi_A \\ \Gamma' & \xrightarrow{f} & \Gamma \end{array}$$

there is a natural isomorphism $\eta : G_{\pi_A^*} f^* \cong q_{A,f}^* G_{\pi_{f^*\{A\}}} : Fam(\mathcal{C})_{\Gamma'} \rightarrow Fam(\mathcal{C})_{\Gamma'.f^*\{A\}}$.

Proof. Let $u : (\Gamma, \alpha) \rightarrow (\Gamma, \beta)$ be a morphism between two objects in $Fam(\mathcal{C})_{\Gamma}$. Then $q_{A,f}^* G_{\pi_{f^*\{A\}}}(u)$ is a morphism between:

$$\begin{aligned} & \left(\{(\gamma', t) \mid \gamma' \in \Gamma', t : 1 \rightarrow A_{f(\gamma')}\}, \lambda(\gamma', t). \prod_{\gamma \in f(\text{fst}(\gamma', t))} \alpha(\gamma) \right) \rightarrow \\ & \left(\{(\gamma', t) \mid \gamma' \in \Gamma, t : 1 \rightarrow A_{f(\gamma')}\}, \lambda(\gamma', t). \prod_{\gamma \in f(\text{fst}(\gamma', t))} \beta(\gamma) \right) \end{aligned}$$

induced by the UMP of the product from the family of morphisms: $u(\gamma) : \alpha(\gamma) \rightarrow \beta(\gamma)$, while $G_{\pi_A^*} f^*(u)$ is the morphism:

$$\begin{aligned} & \left(\{(\gamma', t) \mid \gamma' \in \Gamma', t : 1 \rightarrow A_{f(\gamma')}\}, \lambda(\gamma', t). \prod_{\gamma \in \text{fst}(f(\gamma'), t)} \alpha(\gamma) \right) \rightarrow \\ & \left(\{(\gamma', t) \mid \gamma' \in \Gamma, t : 1 \rightarrow A_{f(\gamma')}\}, \lambda(\gamma', t). \prod_{\gamma \in \text{fst}(f(\gamma'), t)} \beta(\gamma) \right) \end{aligned}$$

arising in the same way. But these are clearly isomorphic in a canonical way. \square

To show that the families model supports Π and Σ type formers, then, it remains to check that there exists an isomorphism $\Gamma.\Sigma_A B \cong \Gamma.A.B$.

Expanding the left hand side, we get:

$$\{(\gamma, t) \mid \gamma \in \Gamma, t \in \prod_{((\gamma, a), b) \in \text{fst}^{-1}(\gamma, a)} ((\gamma, a), b)\}$$

while the right hand side is the set:

$$\{(\gamma, a, b) \mid \gamma \in \Gamma, a \in A_\gamma, b \in B_{A_\gamma}\}$$

which are isomorphic in a canonical way. The families model therefore supports the type formers Π and Σ . It is also clear that the families model supports \sqcap -types if \mathcal{V} has small products, and \sqcup if \mathcal{V} has small coproducts that distribute over \otimes (Frobenius reciprocity).

The model also supports Id -types. For any $A \in Fam(\mathbf{Sets})_\Gamma$, let $Id_A := A \times A$, and $r_A : A \rightarrow A \times A$ be the diagonal map. Then $\Gamma.A.A^+.Id$ is isomorphic to $\Gamma.A \times \Gamma.A$, so for any commutative square:

$$\begin{array}{ccc} \Gamma.A & \xrightarrow{f} & \Delta.C \\ \downarrow r_A & & \downarrow \pi_C \\ \Gamma.A.A^+.Id & \xrightarrow{g} & \Delta \end{array}$$

we get a lift $\Gamma.A.A^+.Id \rightarrow \Delta.B$ by $f \circ \pi_1$, where $\pi_1 : \Gamma.A \times \Gamma.A \rightarrow \Gamma.A$ is a projection from the product. To deal with the operators M and L , we make use of the following lemma:

Lemma 6.4. An adjunction

$$\begin{array}{ccc} & F & \\ \mathcal{C} & \xleftarrow{\quad} & \mathcal{D} \\ & G & \end{array}$$

induces a fiber adjunction between the corresponding fibrations:

$$\begin{array}{ccc} & \mathbf{F} & \\ & \downarrow \perp & \\ \text{Fam}(\mathcal{C}) & \xrightarrow{\quad \mathbf{G} \quad} & \text{Fam}(\mathcal{D}) \\ & \downarrow p \quad \downarrow q & \\ & \mathbf{Sets} & \end{array}$$

Proof. Let $\mathbf{G} : \text{Fam}(\mathcal{C}) \rightarrow \text{Fam}(\mathcal{D})$ and $\mathbf{F} : \text{Fam}(\mathcal{D}) \rightarrow \text{Fam}(\mathcal{C})$ be the induced cartesian functors, defined pointwise in the obvious way. Define the vertical counit $\epsilon : \mathbf{F}\mathbf{G} \Rightarrow 1_{\text{Fam}(\mathcal{C})}$, given at every component $\{A\}_I \in \text{Fam}(\mathcal{C})_I$ by the underlying counit, $\epsilon_A = (1_I, \epsilon_{A_i})$. For any morphism $(u, \alpha) : \mathbf{F}(O, f) \rightarrow (O', g)$, any morphism $(v, \beta) : (O, f) \rightarrow \mathbf{G}(O', g)$ such that $\epsilon_{(O', g)} \circ \mathbf{F}(v\beta) = (u, \alpha)$, must satisfy $v = u$, since \mathbf{F} is constant on the function part of a morphism. But then each component of the family of morphisms, β is a morphism $\beta_o : f(o) \rightarrow G(g(o'))$ fitting into the following diagram:

$$\begin{array}{ccc} F(f(o)) & \xrightarrow{F(\beta_o)} & FG(g(o')) \\ & \searrow \alpha_o & \downarrow \epsilon_{g(o')} \\ & & g(o') \end{array}$$

so by the universal property of the counit, each such β_o is uniquely defined. \square

As a result, the families model supports the operators M and L whenever \mathcal{V} is a concrete category such that the free object generated by the singleton is the unit of the monoidal structure.

When it comes to universes, we need to be careful so as to not run into size issues. The simplest scenario is when the category \mathcal{V} is small. Then we define $\mathcal{L} \in \text{Fam}(\mathbf{Sets})_\Gamma$ as $(\Gamma, \lambda\gamma. \text{Ob}(\mathcal{V}))$. Since any section of $\pi(\mathcal{L}) : \Gamma.\mathcal{L} \rightarrow \Gamma$ consists of a selection of morphisms $f_\gamma : \mathbf{1} \rightarrow \text{Ob}(\mathcal{V})$, this defines an object (Γ, f) of $\text{Fam}(\mathcal{V})$ where $f(\gamma) = f_\gamma(\star)$. Of course, this universe will be closed under all purely linear type formers that are supported by \mathcal{V} , since these are constructed pointwise. However, we cannot construct a cartesian universe in the same fashion, since \mathbf{Set} is a large category. We need to restrict ourselves to a certain subset of the cumulative hierarchy of all sets.

Definition 6.5. For an ordinal α , the set \mathcal{V}_α , is defined by transfinite induction as follows:

- $V_0 = \emptyset$
- $V_{\alpha+1} = \mathcal{P}V_\alpha$
- $V_\beta = \bigcup_{\alpha < \beta} (V_\alpha)$ (β is a limit).

The objects of \mathbf{Set} consists of all of the sets such that they are elements of V_α , for some α .

Definition 6.6. For any ordinal α , the define the subcategory \mathbf{Set}_α of \mathbf{Set} , whose objects are sets X such that $X \in V_\alpha$.

The idea is to select a large enough cardinal κ , such that \mathbf{Set}_κ can be thought of as a model of ZFC itself. This is the notion of a Grothendieck universe. First, recall the notion of an inaccessible cardinal:

Definition 6.7. A cardinal, κ , is said to be:

- **Uncountable**, if it is larger than the smallest infinite set.
- A **strong limit**, if for any $\lambda < \kappa$, we have $2^\lambda < \kappa$.
- **Regular**, if it is not the union of a family of sets of size $< \kappa$ indexed by a set of size $< \kappa$.
- **Inaccessible**, if it is uncountable, a strong limit and regular.

It is important to keep in mind that the existence of inaccessible cardinals is independent of ZFC. But assuming their existence allows us to form a set theoretic universe which can model ZFC itself.

Definition 6.8. A **Grothendieck universe** is a set V_κ where κ is an inaccessible cardinal. With respect to such a universe, a set X is small if $X \in V_\kappa$, otherwise it is large.

It is now easy to see that the category \mathbf{Set}_κ corresponding to a Grothendieck universe has limits and colimits for all κ -small diagrams. For clarity, we will refer to the category consisting of both small and large sets by **SETS** with respect to this universe. By enlarging the families model and make **SETS** the new base category, we can construct cartesian universes in the following fashion:

Let $\mathcal{U} \in \text{Fam}(\mathbf{SETS})_\Gamma$ be defined by $(\Gamma, \lambda\gamma. V_\kappa)$. Any section $t : \Gamma \rightarrow \Gamma.\mathcal{U}$ defines a Γ indexed collection (Γ, t_γ) of sets in V_κ , which gives an interpretation of $El(t)$. Following closely the proof of theorem 6.2, paying attention to size, we get the following result:

Corollary 6.9. For any function $u : S \rightarrow S'$ such that for each $s \in S'$, the subset $u^{-1}(s)$ is small, the functor $u^* : \text{Fam}(\mathbf{SETS})_{S'} \rightarrow \text{Fam}(\mathbf{SETS})_S$ has left and right adjoints, Σ and Π . Furthermore, for each family (S', f) in $\text{Fam}(\mathbf{SETS})_S$, each member of the family $\Sigma(\Gamma, f) \in \text{Fam}(\mathbf{SETS})_S$ is small.

By this argument, we find that the enlarged families model supports cartesian universes that are closed under Π and Σ . Since the projections $\pi_U : \Gamma.U \rightarrow \Gamma$ associated to universes are such that the sets $\pi_U(\gamma)$ are small, for sections $t : \Gamma \rightarrow \Gamma.U$ and $s : \Gamma.El(t) \rightarrow \Gamma.El(t).U$, define $\hat{\Pi}_{El(t)}v$ as the section which associates each γ to the image of the member $El(v)_\gamma$ under functor Π . The case for Σ is completely analogous.

Note that this procedure of enlarging the base category can also be used to account for cases where the category \mathcal{V} is large, provided the existence of enough inaccessible cardinals.

6.1.1 Concrete examples

To summarize, these are the conditions imposed in order to support all of the type formers described in section 4 using the families model:

- A metatheory in which we can construct a Grothendieck universe (in ZFC, at least one inaccessible cardinal).
- A symmetric monoidal closed category \mathcal{V} , with small products and coproducts, which distribute over the monoidal structure.
- An adjunction $L \dashv M$ between \mathcal{V} and \mathbf{Sets} , such that $L\{*\}$ is isomorphic to the unit of the monoidal structure of \mathcal{V} .

Some concrete choices for \mathcal{V} that fulfill these conditions are:

- The category **AbGroups** of abelian groups with the monoidal structure given by the tensor product of abelian groups. Here $L \dashv M$ arises from the free functor on abelian groups.
- More generally, for any commutative ring R , the category $R\text{-}\mathbf{Mod}$ of modules over R with the free functor adjunction
- The category **CGTop**_{*}, of pointed compactly generated topological spaces, with the smash product as monoidal structure. The functor M is here the forgetful functor which both forgets the base point and the topology, which has a left adjoint given by the discrete topology, and then taking the coproduct with the point to create a pointed space. The unit of **CGTop**_{*} is the two point discrete set S^0 , which is precisely the image of the point in the adjunction above.

6.1.2 Syntactic enriched categories

As an in depth example construction in the families model of linear dependent type theory, recall the notion of an enriched category in as defined in 3.12. Our goal is to construct a syntactic analogue of this construction in the families model, such that the interpretation of this is an enriched category. There are two ways of doing this. The first is a meta construction, in which we ask for the existence of certain types and judgemental equalities to hold, whereas the second, internal definition, is a type **Enr-Cat**, whose terms are interpreted as enriched categories. The type **Enr-Cat** allows us to formulate theorems about enriched categories directly in the syntax, but requires more involved type formers, such as universes, and the following type, with which we can reason (non-linearly) about the hom-sets of \mathcal{V} :

$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash [A, B] \text{ type}} \text{ hom-F}$		$\frac{\Gamma \vdash f : [A, B] \quad \Gamma; \Xi \vdash a : A}{\Gamma; \Xi \vdash f(a) : B} \text{ hom-E}$	
$\frac{\Gamma; x : A \vdash t : B}{\Gamma \vdash [\lambda x.t] : [A, B]} \text{ hom-I}$		$\frac{\Gamma; x : A \vdash t : B \quad \Gamma; \Xi \vdash a : A}{\Gamma; \Xi \vdash [\lambda x.t](a) \equiv t[a/x] : B} \text{ hom-C}$	

In the presence of linear function types (\multimap) and the M -functor, this type can be just as well be constructed as $(A \multimap B)_M$. However, adding it as a primitive allows us to talk about the hom-sets of \mathcal{V} even when it is not closed. Instead, the semantics of $[-, -]$ will be an object in \mathcal{T} which “represents” the morphisms of \mathcal{L} :

Definition 6.10. A model of linear dependent type theory supports $[-, -]$ -types, if, for all objects $A, B \in \mathcal{L}_\Gamma$, there exists an object $T[A, B] \in \mathcal{T}_\Gamma$ such that $\mathcal{T}_\Gamma(X, T[A, B]) \cong \mathcal{L}_{\Gamma.X}(A, B)$.

Definition 6.11 (meta-theoretic enriched categories). A meta-theoretic \mathcal{V} -enriched category in a context Γ consists of the following data:

- A (cartesian) type $\Gamma \vdash A$ type,
- for any $x, y : A$, a linear type $\Gamma, x, y : A \vdash B_{x,y}$ linear,
- terms $\Gamma, x, y, z : A; g : B_{y,z}, f : B_{x,y} \vdash M(g, f) : B_{x,z}$,
- $\Gamma, x : A; \cdot \vdash j_x : B_{x,x}$ and
- judgemental equalities $\Gamma, x, y : A; f : B_{x,y} \vdash f \equiv M(j_y \otimes f)$ and $\Gamma, x, y : A; f : B_{x,y} \vdash f \equiv M(f \otimes j_x)$.

The interpretation of this in the families model are precisely \mathcal{V} -enriched categories. The underlying set of objects will be $[[A]]$, the interpretation of B is a function $B : [[A]] \times [[A]] \rightarrow \mathcal{V}$, which for each pair $x, y \in A$ assigns an object of \mathcal{V} .

Definition 6.12 (Internal enriched categories). The type **Enr-cat** is defined as

$$\text{Enr-Cat} := \Sigma_{A:\mathcal{U}} \Sigma_{B:\Pi_{x,y:A} \mathcal{L}} \Sigma_{M:\Pi_{x,y,z:A} [B_{y,z} \otimes B_{x,y}, B_{x,z}]} \Sigma_{j:\Pi_{x:A} [I, B_{x,x}]} [\lambda f.f] = [\lambda f.M_{x,x,y}(f \otimes j_x)] \times [\lambda f.f] = [\lambda f.M_{x,x,y}(j_y \otimes f)]$$

Since $\mathbf{Fam}(\mathbf{Set})$ is an extensional model, whenever the $a =_A b$ is inhabited, we have $[[a]] = [[b]]$. Furthermore, since the equalities p , q and r are all identifying morphisms in the image of the faithful functor M , we have:

$$[[f \circ 1_x]] \equiv [[1_y \circ f]] \equiv [[f]]$$

and

$$[[h \circ (g \circ f)]] \equiv [[(h \circ g) \circ f]]$$

for all $f : B_{x,y}$, $g : B_{y,z}$ and $h : B_{z,w}$, demonstrating that composition is unital and associative in the enriched category. For example, we may choose \mathcal{V} to be the category of groups equipped with the usual tensor product of groups. Here the L functor forms the free abelian group of a set, and the construction above will yield an Ab-enriched category.

6.2 Diagrams in monoidal categories

Expanding on the set indexed families example, for any category \mathcal{C} , there is a fibration $\text{cod} : \mathbf{Diag}(\mathcal{C}) \rightarrow \mathbf{Cat}$, where the total category $\mathbf{Diag}(\mathcal{C})$ has functors $J : \mathcal{D} \rightarrow \mathcal{C}$ into \mathcal{C} as objects, and whose morphisms between $J : \mathcal{D} \rightarrow \mathcal{V}$ and $J' : \mathcal{C} \rightarrow \mathcal{V}$ consist of functors $F : \mathcal{C} \rightarrow \mathcal{D}$ such that $J \circ F = J'$. In other words, the fibers of $\mathbf{Diag}(\mathcal{C})$ are functor categories, which we write $[\Gamma, \mathcal{C}]$, for any small category Γ . Any functor $F : \mathcal{A} \rightarrow \mathcal{B}$ in the base induces a canonical lift $F^* : [\mathcal{B}, \mathcal{C}] \rightarrow [\mathcal{A}, \mathcal{C}]$ simply given by precomposition.

Again, when \mathcal{C} has a terminal object \top such that the collections $\mathcal{C}(\top, A)$ are small for any $A \in \mathcal{C}$, then this forms a comprehension category,

$$\begin{array}{ccc} \mathbf{Diag}(\mathcal{C}) & \xrightarrow{\pi} & \mathbf{Cat}^{\rightarrow} \\ \downarrow \text{dom} & \swarrow \text{cod} & \\ \mathbf{Cat} & & \end{array}$$

where the functor π is defined as follows. Given any diagram $A : \Gamma \rightarrow \mathcal{C}$, we let the category $\Gamma.A$ be the category whose objects are pairs (γ, t_γ) where $\gamma \in \Gamma$ and $t_\gamma : 1 \rightarrow A(\gamma)$. Morphisms $(\gamma, t_\gamma) \rightarrow (\gamma', t_{\gamma'})$ consists of pairs (u, α) where $u : \gamma \rightarrow \gamma'$ is a functor and $\alpha A(u) \circ t_\gamma \Rightarrow t_{\gamma'}$ is a natural transformation. The comprehension functor $\pi : \mathbf{Diag}(\mathcal{C}) \rightarrow \mathbf{Cat}^{\rightarrow}$ sends the diagram $A : \Gamma \rightarrow \mathcal{C}$ to obvious forgetful functor:

$$\begin{aligned} \pi_A : \Gamma.A &\rightarrow \Gamma \\ \pi(\gamma, a_\gamma) &= \gamma \end{aligned}$$

When \mathcal{C} is any symmetric monoidal category \mathcal{V} , there is an obvious symmetric monoidal structure on each fiber $[\Gamma, \mathcal{V}]$, given pointwise:

$$\begin{aligned} J \otimes J'(x) &= J(x) \otimes J'(x) : [\Gamma, \mathcal{V}] \\ J \otimes J'(f) &= J(f) \otimes J'(f) : J(x) \otimes J'(x) \rightarrow J(y) \otimes J'(y) \end{aligned}$$

Lifts are simply given by precomposition, and we confirm that for $f : \Delta \rightarrow \Gamma$, and $A : \Gamma \rightarrow \mathcal{C}$, $\pi(f^*)$ is the pullback:

$$\begin{array}{ccc} \Delta.A\{f^*\} & \xrightarrow{q} & \Gamma.A \\ \downarrow \pi_{A\{f^*\}} & & \downarrow \pi_A \\ \Delta & \xrightarrow{f} & \Gamma \end{array}$$

where $q(\delta, a_{f(\delta)}) = (f(\delta), a_{f(\delta)})$. Since reindexing functors are strictly monoidal, this yields a split monoidal fibration. Restricting the base to groupoids instead of categories, and setting $\mathcal{C} = \mathbf{Gpd}$ we get a model of linear dependent type theory which expands the groupoid model by Hofmann and Streicher [7]:

$$\begin{array}{ccc} \mathbf{Diag}(\mathcal{V}) & & \mathbf{Diag}(\mathbf{Gpd}) \xrightarrow{\pi} \mathbf{Gpd}^{\rightarrow} \\ & \searrow \text{dom} & \downarrow \text{dom} \swarrow \text{cod} \\ & & \mathbf{Gpd} \end{array}$$

Since \top is the groupoid $\mathbf{1}$ consisting of a single object, we will equate the functor $t_\gamma : 1 \rightarrow A(\gamma)$ with an object a_γ of $A(\gamma)$, and the natural transformation $\alpha : A(u) \circ t_\gamma \Rightarrow t_{\gamma'}$, with a morphism $\alpha_\gamma : A(u)(a_\gamma) \rightarrow a_{\gamma'}$.

As shown in [7], this model supports Π and Σ , and importantly provides an example of *Id*-types where there might be more terms of $x =_A x$ than $\text{refl}(x)$, which sets the stage for the homotopy interpretation of dependent type theory. Here, the interpretation of identity type Id_A is the functor $\text{Id}_A : \Gamma.A.A^+ \rightarrow \mathbf{Gpd}$ which sends an object $(\gamma, a_\gamma, b_\gamma)$ to the discrete groupoid $\delta \text{Hom}_{A(\gamma)}(\gamma)(a, b)$, and a morphism $(u, \alpha, \beta) : (\gamma, a_\gamma, b_\gamma) \rightarrow (\gamma', a_{\gamma'}, b_{\gamma'})$ to the functor $\alpha^{-1} \circ A(u)(-) \circ \beta : \delta(\text{Hom}_A(a, b)) \rightarrow \delta(\text{Hom}_A(a', b'))$, which sends $f : a \rightarrow b$ to the composite $\alpha^{-1} \circ A(u)(f) \circ \beta : a' \rightarrow A(u)a \rightarrow A(u)b \rightarrow b'$.

We see that the extended context $\Gamma.A.A^+.Id_A$ is equivalent to $\Gamma.A^\rightarrow$, and we define $r_A : (\Gamma.A) \rightarrow (\Gamma.A^\rightarrow)$ to be the functor sending morphisms:

$$(u, \alpha) : (\gamma, a_\gamma) \rightarrow (\gamma', a'_{\gamma'})$$

to squares:

$$\begin{array}{ccc} (\gamma, a_\gamma) & \xrightarrow{(u, \alpha)} & (\gamma', a'_{\gamma'}) \\ \downarrow (1_\gamma, 1_{a_\gamma}) & & \downarrow (1_{\gamma'}, 1_{a'_{\gamma'}}) \\ (\gamma, a_\gamma) & \xrightarrow{(u, \alpha)} & (\gamma', a'_{\gamma'}) \end{array}$$

Theorem 6.13. *For any indexed groupoid A in Γ , the construction Id_A described above forms support for Id -types in the diagrams model.*

Proof. That Id_A satisfies condition (1) and (2) of definition 5.14 is proved in [7]. It remains to show that condition (3) is satisfied by this construction. Let $C, \Xi \in \mathcal{L}_{\Gamma.A.A^+.Id_A}$, and assume we have sections $M, N : \Gamma \rightarrow \Gamma.A$ and $P : \Gamma \rightarrow \Gamma.Id_A\{M\}\{N^+\}$. We are given a natural transformation $c : \Xi\{r_A\} \rightarrow C\{r_A\}$, between between the two functors $\Xi \circ r_A : \Gamma.A \rightarrow \Gamma.A.A^+.Id_A \rightarrow \mathcal{V}$ and $C \circ r_A : \Gamma.A \rightarrow \Gamma.A.A^+.Id_A \rightarrow \mathcal{V}$ and need to display a natural transformation $\hat{c}_{[M, N, P]}$ between:

$$\Xi \circ P^+ \circ N^+ \circ M : \Gamma \rightarrow \Gamma.A \rightarrow \Gamma.A.A^+ \rightarrow \Gamma.A.A^+.Id \rightarrow \mathcal{V}$$

to

$$C \circ P \circ N^+ \circ M : \Gamma \rightarrow \Gamma.A \rightarrow \Gamma.A.A^+ \rightarrow \Gamma.A.A^+.Id_A \rightarrow \mathcal{V}$$

such that $\hat{c}_{[M, M, refl(M)]} = c\{M\}$. The key point to observe is that there is always an isomorphism $(\gamma, P_\gamma : M_\gamma \rightarrow N_\gamma) \cong (\gamma, 1_{M_\gamma} : M_\gamma \rightarrow M_\gamma)$ given by the commutative diagram:

$$\begin{array}{ccc} (\gamma, M_\gamma) & \xrightarrow{1_M} & (\gamma, M_\gamma) \\ \downarrow P & & \downarrow 1_M \\ (\gamma, N_\gamma) & \xrightarrow{P^{-1}} & (\gamma, M_\gamma) \end{array}$$

in the groupoid $\Gamma.A^\rightarrow$, giving rise to a natural isomorphism $\phi : r_A \circ M \cong P^+ \circ N^+ \circ M$. We define $\hat{c}[M, N, P]$ as the composite:

$$\hat{c}[M, N, P] := \Xi \phi \circ c\{M\} \circ C_{\phi^{-1}} : \Xi \circ P^+ \circ N^+ \circ M \rightarrow \Xi \circ r_A \circ M \rightarrow C \circ r_A \circ M \rightarrow C \circ P^+ \circ N^+ \circ M$$

. Notice that when $M \equiv N$ and $P = refl(M)$, ϕ is the identity natural transformation. \square

As in the families model, limits and colimits are constructed pointwise, and preserved by precomposition, so the model supports $\&$, \top , \oplus , 0 , if \mathcal{V} has binary products, terminal object, binary coproducts and initial object respectively.

The situation for \multimap requires a bit of care. Under what conditions is the functor category $[\mathcal{C}, \mathcal{V}]$ monoidal closed with respect to the pointwise tensor product? For all $F \in [\mathcal{C}, \mathcal{V}]$, we need to find a right adjoint to the functor $- \otimes F : [\mathcal{C}, \mathcal{V}] \rightarrow [\mathcal{C}, \mathcal{V}]$. In other words for all functors G, H , we need

For $F, G, H : \mathcal{C} \rightarrow \mathcal{V}$ a natural transformation $\eta : F \otimes G \rightarrow H$, a natural transformation is for all $g : z \rightarrow x$ and $f : x \rightarrow y$ in \mathcal{C} , a diagram:

$$Fx \otimes Gx \xrightarrow{Fg \otimes Gf} Fz \otimes asd$$

Naively, $F, G, H : \mathcal{C} \rightarrow \mathcal{V}$ a natural transformation $\eta : F \otimes G \rightarrow H$ gives rise to a collection of maps $\hat{\eta}_A : F(A) \rightarrow [G(A), H(A)]$, but a priori these maps do not constitute a natural transformation $F \rightarrow [G, H]$.

Instead, notice that a natural transformation $\eta : F \rightarrow G$, is for each morphism $f : A \rightarrow B$ in \mathcal{C} , a pair of morphisms η_A and η_B such that:

$$\begin{array}{ccc} F(A) & \xrightarrow{Ff} & F(B) \\ \downarrow \eta_A & & \downarrow \eta_B \\ G(A) & \xrightarrow{Gf} & G(B) \end{array}$$

or equivalently, an “element” of $[F(A), G(A)] \times [F(B), G(B)]$ such that the postcomposing the first component with Gf is the same as precomposing the second component with Ff :

$$[F(A), G(A)] \times [F(B), G(B)] \xrightarrow[\pi_2(Ff_*)]{\pi_1(Gf^*)} [F(A), G(B)]$$

So if \mathcal{V} has all limits, a natural candidate for the internal hom of the functor category becomes the equalizer:

$$[F, G](x) = Eq(\Pi_{x \in \mathcal{C}} [Fx, Gx] \xrightarrow[Gf_*]{Gf^*} \Pi_{f : x \rightarrow y} [Fx, Gy])$$

also known as the end $[F, G](x) = \int_{x \in \mathcal{C}} [Fx, Gx]$ of the functor $[F-, G-] : \mathcal{C}^{op} \times \mathcal{C} \rightarrow \mathcal{V}$.

Theorem 6.14. *If \mathcal{V} has internal homs and is complete, $[\mathcal{C}, \mathcal{V}]$ has internal homs, defined as above.*

Proof. For any, $F, G, H \in [\mathcal{C}, \mathcal{V}]$, we need to display bijection

$$[\mathcal{C}, \mathcal{V}](H \otimes F, G) \cong [\mathcal{C}, \mathcal{V}](H, \int_{x \in \mathcal{C}} [Fx, Gx])$$

natural in G and H . Since $[F, G]$ is a constant functor, a map on the right hand side consists of, for every map $f : x \rightarrow y \in \text{Mor}(\mathcal{C})$, of commutative diagrams of the following form:

$$\begin{array}{ccc} Hx & \xrightarrow{Hf} & Hy \\ & \searrow \eta_x & \downarrow \eta_y \\ & & \int_{x \in \mathcal{C}} [Fx, Gx] \end{array}$$

which will determine two unique morphisms, $\hat{\eta}_x : Hx \rightarrow \Pi_{x \in \mathcal{C}} [Fx, Gx]$ and $\hat{\eta}_y : Hy \rightarrow \Pi_{x \in \mathcal{C}} [Fx, Gx]$, such that the following diagram commutes:

$$\begin{array}{ccc} Hx & \xrightarrow{Hf} & Hy \\ & \searrow \hat{\eta}_x & \downarrow \hat{\eta}_y \\ & & \Pi_{x \in \mathcal{C}} [Fx, Gx] \end{array} \xrightarrow[\pi_2(Ff_*)]{\pi_1(Gf^*)} \Pi_{f:x \rightarrow y} [Fx, Gy]$$

projecting out yields a unique collection of morphisms $\pi_x \hat{\eta}_x : Hx \rightarrow [Fx, Gx]$ such that the following diagram commutes:

$$\begin{array}{ccc} Hx & \xrightarrow{Hf} & Hy \\ \downarrow \pi_x \hat{\eta}_x & & \downarrow \pi_y \hat{\eta}_y \\ [Fx, Gx] & & [Fy, Gy] \\ & \searrow [1_{Fx}, Gf] & \downarrow [Ff, 1_{Gy}] \\ & & [Fx, Gy] \end{array}$$

which by the tensor-hom adjunction in \mathcal{V} corresponds precisely to a diagram:

$$\begin{array}{ccc} Hx \otimes Fx & \xrightarrow{Hf \otimes Ff} & Hy \otimes Fy \\ \downarrow \tilde{\eta}_x \pi_x & & \downarrow \tilde{\eta}_y \pi_y \\ Gx & \xrightarrow{Gf} & Gy \end{array}$$

, determining a unique natural transformation $H \otimes F \rightarrow G$. This yields a bijection of homsets natural in all variables. \square

Given any functor $p : \mathcal{D} \rightarrow \mathcal{C}$ in the base, we have:

$$p^*([G, F]) = \lambda x. \int_{p(x) \in \mathcal{C}} [G(p(x)), F(p(x))] = \int_{x \in \mathcal{D}} [G(p(x)), F(p(x))] = [p^*G, p^*F]$$

so we get that the diagrams model supports \multimap if \mathcal{V} is monoidal closed and complete.

Definition 6.15. For any functor $p : \mathcal{A} \rightarrow \mathcal{B}$ in the base, a left or right adjoint to the induced functor $p^* : [\mathcal{B}, \mathcal{V}] \rightarrow [\mathcal{A}, \mathcal{V}]$ is called a **left or right Kan extension** of p .

So, in order for the diagrams model to support \sqcap - and \sqcup -types, we need left and right Kan extensions of every functor between small groupoids.

In order to interpret M and L in this model, the same argument as in lemma 6.4 can be used to show that an adjunction between \mathcal{V} and **Gpd** induces a fiber adjunction between the respective diagram categories. Also, since the composition of

expa

two adjunctions is again an adjunction, we get that whenever the functor $\mathcal{V}(I, -) : \mathcal{V} \rightarrow \mathbf{Set}$ has a left adjoint F , then there exists an adjunction between $\mathbf{Diag}(\mathcal{V})$ and $\mathbf{Diag}(\mathbf{Gpd})$, arising out of:

$$\begin{array}{ccc} \mathcal{V} & \xrightleftharpoons[\mathcal{V}(I, -)]{F} & \mathbf{Set} \\ & \perp & \\ \mathbf{Set} & \xrightleftharpoons[\delta]{\pi_0} & \mathbf{Gpd} \end{array}$$

where π_0 is the functor sending a groupoid to its set of connected components.

Theorem 6.16. *There are models in which M is not faithful.*

Proof. Let \mathcal{V} to be \mathbf{Gpd} so that $L = \delta\pi_0$ and $M = \delta\mathbf{Gpd}(1, -)$. This induces a fiber adjunction $L \dashv M$ where $[L$ is strong monoidal] but M is not faithful. \square

6.2.1 Universes in the diagrams model

As long as \mathbf{Gpd} is large enough to be able to contain the core of \mathcal{V} as an object, we let the linear universe type \mathcal{L} be given by this object.

6.3 Higher linear dependent types

A key feature of the groupoid model is that it provides a model of dependent type theory where there might be nontrivial terms of the identity type. A natural question to ask is whether this higher dimensional feature of type theory can be extended to the linear dependent case. Can we, for example, construct a model where the total category \mathcal{L} of linear types is a 2-category, so that for terms $\Gamma; \Xi \vdash f : A$ and $\Gamma; \Xi \vdash g : A$, the interpretation of $\hat{f}_M = (\otimes_{\Xi \multimap A} \hat{g}_M)$ is an object representing the 2-cells between $[[f]]$ and $[[g]]$?

Such a model would provide the necessary setting for the following linear analogue to the univalence axiom:

$$\frac{\begin{array}{l} \Gamma; \cdot \vdash f : A \multimap B \\ \Gamma; \cdot \vdash g : B \multimap A \\ \Gamma \vdash p : (g \circ f)_M =_{(A \multimap A)_M} (id_A)_M \\ \Gamma \vdash q : (f \circ g)_M =_{(B \multimap B)_M} (id_B)_M \end{array}}{\Gamma \vdash ua(f, g, p, q) : A =_{\mathcal{L}} B} \text{ L-ua}$$

with computation rules:

and

Clearly, for such a model we cannot take M to be factored through \mathbf{Set} , since any groupoid in its image would be discrete. But if the objects of \mathcal{V} are categories, we can let M be given by taking the maximal subgroupoid of these. Instead, we would like the objects of \mathcal{V} be categories, and M be the functor which sends a category to its underlying maximal groupoid. Obvious candidates are \mathbf{Cat} or \mathbf{Gpd} , but we shall go for a more “linear” choice (i.e. non cartesian) option:

Definition 6.17. Let the **higher dimensional model of linear dependent type theory** be given by the diagrams model where \mathcal{V} is the 2-category of small symmetric monoidal groupoids, symmetric monoidal functors and monoidal natural transformations:

$$\begin{array}{ccc} \mathbf{Diag}(\mathbf{SMGpd}) & & \mathbf{Diag}(\mathbf{Gpd}) \xrightarrow{\pi} \mathbf{Gpd}^{\rightarrow} \\ & \searrow \text{cod} & \downarrow \text{cod} \quad \swarrow \text{dom} \\ & & \mathbf{Gpd} \end{array}$$

Recall that the category of small monoidal categories carries a natural symmetric monoidal structure, as briefly outlined in 3.15 and carefully described in [15]. We only restrict ourselves to symmetric monoidal **groupoids** instead of categories to make sure no information gets lost when applying M .¹

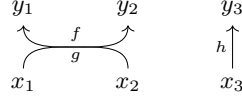
The monoidal structure is preserved while restricting the functor to subcategory of symmetric monoidal groupoids.

For two symmetric monoidal categories \mathcal{A} and \mathcal{B} , category $[\mathcal{A}, \mathcal{B}]$ consisting of monoidal functors and monoidal transformations between them carries a natural monoidal structure [15], and serves as the internal hom of \mathcal{A} and \mathcal{B} . Since \mathbf{SMGpd} is complete, with limits inherited from \mathbf{Gpd} equipped with a pointwise monoidal structure, theorem 6.14 gives us that this model supports \multimap type formers.

Lemma 6.18. There is a functor $F : \mathbf{Gpd} \rightarrow \mathbf{SMGpd}$ which constructs the free symmetric monoidal groupoid of any groupoid, i.e. is a left adjoint to the forgetful functor $U : \mathbf{SMGpd} \rightarrow \mathbf{Gpd}$, forgetting the monoidal structure.

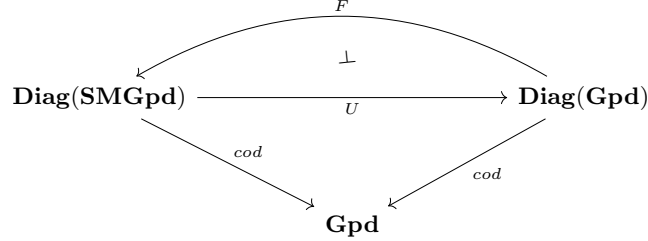
¹Technically, the structure on \mathbf{SMCat} is not quite symmetric monoidal, as the associators, unitors and symmetry functors are only invertible up to higher homotopy. However, if one applies these homotopies whenever necessary, one does get a model of linear dependent type theory.

Proof. Construct the free symmetric monoidal groupoid FC of a small category \mathcal{C} by letting the objects of FC be finite words (x_1, x_2, \dots, x_n) of the objects of \mathcal{C} . Morphisms between words are best thought of as string diagrams, where each string stems from a morphism in \mathcal{C} :



The composition of such morphisms are just the composition of the corresponding string diagrams, composing the labels of strings along the way. The monoidal structure is simply given by concatenation of words, where the empty word corresponds to the unit. From this construction, it should be clear that given any symmetric monoidal groupoid \mathcal{D} , a functor between groupoids $G : \mathcal{C} \rightarrow \mathcal{D}$ induces a functor \hat{G} which maps words $(x_1, x_2, \dots, x_n) \mapsto G(x_1) \otimes G(x_2) \otimes \dots \otimes G(x_n)$, which forms an adjunction $F \dashv U$. \square

This adjunction lifts by lemma 6.4 to a fiber adjunction:



Furthermore, it is clear from the definition of F and the unit I of \mathbf{SMGpd} , defined at ?? that the image of $\mathbf{1}$ under F is precisely I , so the higher dimensional model supports M and L by the fiber adjunction $F \dashv U$.

Armed with these results, we can prove an important first result of the higher dimensional model, answering

Lemma 6.19. For terms $;\Xi \vdash a : A$ and $;\Xi \vdash a' : A$, the interpretation of the type $\cdot \vdash \hat{a}_M = (\otimes \Xi \multimap A)_M \hat{a}'_M$ is a groupoid whose objects are natural transformations between the symmetric monoidal functors $a : \otimes \Xi \multimap A$ and $a' : \otimes \Xi \multimap A$, where \hat{a} and \hat{a}' are the lambda abstracted versions of a and a' .

Proof. \square

Theorem 6.20. *The higher dimensional model supports L-ua.*

Proof. The two generalized elements $g \circ f : I \rightarrow [A, A]$ and $f \circ g : I \rightarrow [B, B]$ correspond by the adjunction to functors $(g \circ f)_M : 1 \rightarrow M[A, A]$, and $(f \circ g)_M : 1 \rightarrow M[B, B]$ which, since M is simply a forgetful functor, determines two monoidal endofunctors GF and FG of A and B . That the types $(g \circ f)_M = (id_A)_M$ and $(f \circ g)_M = (id_B)_M$ are inhabited means that there are natural isomorphisms $\epsilon : GF \rightarrow 1_{A_M}$ and $\eta : FG \rightarrow 1_{B_M}$, witnessing an equivalence of categories $A \cong B$.

It remains to show that η and ϵ are monoidal, to make this an equivalence of monoidal categories \square

6.3.1 Univalent linear type theory

Equipped with the new linear univalence axiom, we are able to prove equalities between types that are easily shown to be isomorphic.

Theorem 6.21. *Assuming L-ua, for all linear types A, B, C , cartesian D in Γ , and cartesian E in $\Gamma.D$, the following types are inhabited:*

1. $\Gamma \vdash A \multimap B \multimap C =_{\mathcal{L}} A \otimes B \multimap C$
2. $\Gamma \vdash (\Sigma_{x:D} E)_L =_{\mathcal{L}} \sqsubset_{x:D} E_L$
3. $(A \& B)_{LM} =_{\mathcal{L}} A_{LM} \otimes B_{LM}$

Proof. 1. We construct maps $h : (A \multimap B \multimap C) \multimap (A \otimes B) \multimap C$

$$\begin{array}{c} \multimap\text{-E} \frac{\Gamma; f : A \multimap B \multimap C \vdash f : A \multimap B \multimap C \quad \Gamma; y : A \vdash y : A}{\Gamma; f : A \multimap B \multimap C, y : A, z : B \vdash f(y) : B \multimap C} \\ \multimap\text{-E} \frac{\Gamma; f : A \multimap B \multimap C, y : A, z : B \vdash f(y) : B \multimap C \quad \Gamma; z : B \vdash z : B}{\Gamma; f : A \multimap B \multimap C, y : A, z : B \vdash f(y)(z) : C} \\ \frac{\Gamma; f : A \multimap B \multimap C, x : A \otimes B \vdash \text{let } x \text{ be } (y, z) \text{ in } f(y)(z) : C}{\Gamma; f : A \multimap B \multimap C \vdash \lambda x. \text{let } x \text{ be } (y, z) \text{ in } f(y)(z) : A \otimes B \multimap C} \multimap\text{-I} \\ \frac{\Gamma; \cdot \vdash \lambda f. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } f(a)(b) : (A \multimap B \multimap C) \multimap (A \otimes B \multimap C)}{\Gamma; \cdot \vdash h : (A \multimap B \multimap C) \multimap (A \otimes B) \multimap C} \multimap\text{-I} \end{array}$$

and $h^{-1} : (A \otimes B \multimap C) \multimap (A \multimap B \multimap C)$:

$$\begin{array}{c}
\frac{\Gamma; \beta : B \vdash \beta : B \quad \Gamma; \alpha : A \vdash \alpha : A}{\Gamma; \alpha : A, \beta : B \vdash \alpha \otimes \beta : A \otimes B} \quad \Gamma; g : A \otimes B \multimap C \vdash g : A \otimes B \multimap C \\
\frac{\Gamma; g : A \otimes B \multimap C, \alpha : A, \beta : B \vdash g(\alpha \otimes \beta) : C}{\Gamma; g : A \otimes B \multimap C, \alpha : A \vdash g(\alpha \otimes \beta) : B \multimap C} \\
\frac{\Gamma; g : A \otimes B \multimap C \vdash \lambda \alpha. \lambda \beta. g(\alpha \otimes \beta) : A \multimap B \multimap C}{\Gamma; \cdot \vdash \lambda g. \lambda \alpha. \lambda \beta. g(\alpha \otimes \beta) : (A \otimes B \multimap C) \multimap (A \multimap B \multimap C)} \multimap\text{-I}
\end{array}$$

and show that they are mutually inverse. $(h \circ h^{-1}) : (A \otimes B \multimap C) \multimap (A \otimes B \multimap C)$ reduces to:

$$\begin{aligned}
(h \circ h^{-1}) &\equiv \lambda \gamma. (\lambda f. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } f(a)(b)) (\lambda g. \lambda \alpha. \lambda \beta. g(\alpha \otimes \beta)) \gamma \equiv \\
&\quad \lambda \gamma. (\lambda f. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } f(a)(b)) (\lambda \alpha. \lambda \beta. \gamma(\alpha \otimes \beta)) \equiv \\
&\quad \lambda \gamma. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } (\lambda \alpha. \lambda \beta. \gamma(\alpha \otimes \beta))(a)(b) \equiv \\
&\quad \lambda \gamma. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } \gamma(a \otimes b) \equiv \\
&\quad \lambda \gamma. \text{let } (a \otimes b) \text{ be } (a, b) \text{ in } \gamma(a \otimes b) \equiv \\
&\quad id_{(A \otimes B \multimap C) \multimap (A \otimes B \multimap C)}
\end{aligned}$$

and $(h^{-1} \circ h) : (A \multimap B \multimap C) \multimap (A \multimap B \multimap C)$ reduces to:

$$\begin{aligned}
(h^{-1} \circ h) &\equiv \lambda \gamma. (\lambda g. \lambda \alpha. \lambda \beta. g(\alpha \otimes \beta)) (\lambda f. \lambda x. \text{let } x \text{ be } (a, b) \text{ in } f(a)(b)) \gamma \equiv \\
&\quad \lambda \gamma. (\lambda g. \lambda \alpha. \lambda \beta. g(\alpha \otimes \beta)) (\lambda x. \text{let } x \text{ be } (a, b) \text{ in } \gamma(a)(b)) \equiv \\
&\quad \lambda \gamma. \lambda \alpha. \lambda \beta. (\lambda x. \text{let } x \text{ be } (a, b) \text{ in } \gamma(a)(b))(\alpha \otimes \beta) \equiv \\
&\quad \lambda \gamma. \lambda \alpha. \lambda \beta. \text{let } (\alpha \otimes \beta) \text{ be } (a, b) \text{ in } \gamma(a)(b) \equiv \\
&\quad \lambda \gamma. \lambda \alpha. \lambda \beta. \gamma(\alpha)(\beta) \equiv \\
&\quad id_{(A \multimap B \multimap C) \multimap (A \multimap B \multimap C)}
\end{aligned}$$

so we inhabit both $(h \circ h^{-1})_M =_{(A \otimes B \multimap C) \multimap (A \otimes B \multimap C)} id_M$ and $(h^{-1} \circ h)_M =_{(A \otimes B \multimap C) \multimap (A \otimes B \multimap C)} id_M$ with **refl**, and then apply the linear univalence axiom to get the desired equality.

2. By Theorem 4.9, there is a linear equivalence $(\Sigma_{x:D} E)_L \cong_{\square_{x:D}} E_L$. The linear univalence axiom gives rise to an equality between these types.
3. We construct a function $f : (A \& B)_{LM} \multimap A_{LM} \otimes B_{LM}$ by:

$$f \equiv \lambda p. \text{let } p \text{ be } x \text{ in } \text{fst}(\sigma(x))_{LM} \otimes \text{snd}(\sigma(x))_{LM}$$

with inverse $f^{-1} : A_{LM} \otimes B_{LM} \multimap (A \& B)_{LM}$:

$$f^{-1} \equiv \lambda q. \text{let } q \text{ be } y, z \text{ in } \left((\text{let } \alpha \text{ be } y \text{ in } \sigma(\alpha)), (\text{let } \beta \text{ be } z \text{ in } \sigma(\beta)) \right)_{LM}$$

which are easily seen to be mutually inverse. □

7 Ideas

H-spaces generalized to the tensor product?

7.1 Dependent ordered type theory

If one were to drop exchange as a structural rule, this should correspond to dropping the requirement of the semantic category being symmetric. A particular construction which might be interesting to explore in this setting would be simplicial sets equipped with the join. Would this work?

References

- [1] A mixed linear and non-linear logic: Proofs, terms and models (preliminary report).
- [2] P Benton. A mixed linear and non-linear logic: Proofs, terms and models. In *Computer Science Logic*, pages 121–135. Springer, 1995.
- [3] M. Calvo, A. M. Cegarra, and B. A. Heredia. Structure and classification of monoidal groupoids. *Semigroup Forum*, 87(1):35–79, Aug 2013.
- [4] Ross Duncan et al. *Types for quantum computing*.
- [5] Martin Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *International Workshop on Computer Science Logic*, pages 427–441. Springer, 1994.

- [6] Martin Hofmann. Syntax and semantics of dependent types. In *Extensional Constructs in Intensional Type Theory*, pages 13–54. Springer, 1997.
- [7] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. *Twenty-five years of constructive type theory (Venice, 1995)*, 36:83–111, 1998.
- [8] Bart Jacobs. Comprehension categories and the semantics of type dependency. *Theoretical Computer Science*, 107(2):169–207, 1993.
- [9] Max Kelly. *Basic concepts of enriched category theory*, volume 64. CUP Archive, 1982.
- [10] Neelakantan R Krishnaswami, Pierre Pradic, and Nick Benton. Integrating linear and dependent types. In *ACM SIGPLAN Notices*, volume 50, pages 17–30. ACM, 2015.
- [11] Daniel R Licata, Michael Shulman, and Mitchell Riley. A fibrational framework for substructural and modal logics (extended version). 2017.
- [12] Peter LeFanu Lumsdaine and Michael A Warren. The local universes model: an overlooked coherence construction for dependent type theories. *ACM Transactions on Computational Logic (TOCL)*, 16(3):23, 2015.
- [13] Paul-André Mellies. Categorical semantics of linear logic.
- [14] Lucius Gregory Meredith. Linear types can change the blockchain. *arXiv preprint arXiv:1506.01001*, 2015.
- [15] Vincent Schmitt. Tensor product for symmetric monoidal categories. *arXiv preprint arXiv:0711.0324*, 2007.
- [16] Michael Shulman. Framed bicategories and monoidal fibrations. *Theory and Applications of Categories*, 20(18):650–738, 2008.
- [17] Matthijs Vákár. Syntax and semantics of linear dependent types. *CoRR*, abs/1405.0033, 2014.
- [18] Marek Zawadowski. Lax monoidal fibrations. *Models, Logics, and Higher-Dimensional Categories: A Tribute to the Work of Mihály Makkai (CRM Proceedings 53, 2011)*, pages 341–424, 2011.