Models of linear dependent type theory

Martin Lundfall

December 4, 2017

Abstract

In this paper, we construct a type theory which deals with non-linear, ordinary dependent types (which we will call *cartesian*), and *linear types*, where both constructs may depend on terms of cartesian types. In the interplay between the cartesian and linear types we introduce the new type formers $\sqcap_{x:A} B$ and $\sqsubset_{x:A}\ B$, akin to Π and Σ, but where the dependent type B, (and therefore the resulting construct) is a linear type. These can be seen as internalizing quantification over linear propositions. We also introduce the modalities M and L, transforming linear types into cartesian types and vice versa.

We interpret the theory in a split comprehension category $\pi : \mathcal{T} \to \mathcal{C}^{\to}$ [Jac93], accompanied by a split monoidal fibration (Definition 4.3), $q : \mathcal{L} \to \mathcal{C}$. Here $\mathcal{C}$ models a category of contexts, so that for any $\Gamma \in \mathcal{C}$, the fiber $\mathcal{T}_\Gamma$ is the category of cartesian types which can be formed in the context Γ, while the fiber $\mathcal{L}_\Gamma$ is a monoidal category of linear types in Γ. In this setting, the type formers $\sqcap_{x:A}$ and $\sqsubset_{x:A}$ are understood as right and left adjoints of the monoidal reindexing functor $\pi_A^* : \mathcal{L}_\Gamma \to \mathcal{L}_{\Gamma.A}$ corresponding to the weakening projection $\pi_A : \Gamma.A \to \Gamma$ in $\mathcal{C}$. The operators M and L give rise to a fiberwise adjunction $L \dashv M$ between $\mathcal{L}$ and $\mathcal{T}$, where we understand the traditional exponential modality as the comonad $! = LM$.

We provide two concrete examples of models, the *set-indexed families model* and the *diagrams model*. In the former, cartesian types are interpreted in the familiar way, as sets indexed by their context set Γ, and linear types are interpreted as Γ-indexed family of objects of a symmetric monoidal category $\mathcal{V}$. The latter model extends the groupoid model of dependent type theory [HS98] to accomodate linear types. Here, cartesian types over a context Γ are interpreted as a family of groupoids indexed over the groupoid Γ, while linear types are interpreted as diagrams over groupoids, $A : \Gamma \to \mathcal{V}$ in any symmetric monoidal category $\mathcal{V}$. We show that the *diagrams model* can under certain conditions model a linear analogue of the univalence axiom, and provide some discussion on the higher-dimensional nature of linear dependent types.

Contents

1 Introduction & summary of results

Lately, there has been an increasing interest in combining linear and dependent types [Sch14], [KPB15], [VÍ15], [McB16]. The idea is that such a theory would inherit the higher-order nature of dependent types, while maintaining a careful account of how assumptions are used in a derivation. It is not completely clear, however, what the synthesis looks like, since in dependent type theory, variables may appear in both terms and types, but linear type theory only allows each variable to appear freely exactly once. Here, we take an approach inspired by [KPB15] and [VÍ15], in which we distinguish between non-linear, dependent types (which we call *cartesian*), and linear types, and circumvent the issue by only allowing cartesian terms to appear in types (both cartesian and linear).

The theory splits contexts into two parts, divided by a semicolon, where the first part contains cartesian assumptions, for which weakening and contraction is admissible, while the second part contains linear assumptions, for which only exchange is allowed. We introduce two new type formers, $\sqcap_{x:A} B$ and $\sqsubset_{x:A}\ B$, akin to Π and Σ, but where the dependent type B (and therefore the resulting construct) is a linear. The traditional ! modality is deconstructed as a comonad arising from the adjoint pair $L \dashv M$, where L is a functor (or modality) sending cartesian types into linear, and M sends linear types to cartesian. We have $\Pi_{x:A} B_M \cong (\sqcap_{x:A} B)_M$, for linear B, and, assuming a few additional rules, a linear equivalence $(\Sigma_{x:A} C)_L \cong \sqsubset_{x:A} C_L$ for cartesian C.

Compared to ordinary depdenent type theory, we get additional elimination and computation rules for both Σ and Id-types when eliminating into a linear type.

We postulate the existence of two universes, L and U, containing codes of linear and cartesian types, respectively and assumed to be closed under all type formers.

We develop categorical semantics for the theory, using traditional ZFC as a metatheory (assuming some inaccesible cardinals to support universes in Section 5.1.1). Our starting point is a *comprehension category* [Jac93], $\pi : \mathcal{T} \to \mathcal{C}$ equipped with a *split monoidal fibration* $q : \mathcal{L} \to \mathcal{C}$ over the same base. A split monoidal fibration has just enough structure to make the fibers over a context $\mathcal{L}_\Gamma$ into monoidal categories, and reindexing functors (strict) monoidal functors. The traditional linear type formers $\&, \oplus, 0, \top, \multimap$ correspond to the existence of binary products and coproducts, initial and terminal object and internal homs in each fiber, such that these are preserved under reindexing. The new type formers $\sqcap_{x:A} B$ and $\sqsubset_{x:A}\ B$ correspond to right and left adjoints to the reindexing functor $\pi_A : \mathcal{L}_{\Gamma.A} \to \mathcal{L}_\Gamma$, while the modalities L and M give rise to a fiber adjunction between $\mathcal{L}$ and $\mathcal{T}$. The new rules for Σ are automatically satisfied by the semantic interpretation of Σ_A as a left adjoint to the reindexing functor $\pi_A^* : \mathcal{T}_\Gamma \to \mathcal{T}_{\Gamma.A}$. The new rules for Id-types impose an additional condition on the semantic interpretation of Id, which are always fulfilled if our identity types are extensional.

We consider two concrete models, the *families model*, in which cartesian types consist of families of sets, indexed by their context set Γ, and a linear type in the context Γ is a Γ-indexed family of objects in a given symmetric monoidal category $\mathcal{V}$. Examples of suitable $\mathcal{V}$ supporting all type formers present in our syntax are $\mathbf{AbGrp}$, $\mathbf{GCTop}_*$, $\mathbf{Vect}_F$, i.e. the category of abelian groups, the category of compact generated, pointed topological space and the category of vector spaces over a field F, respectively.

Generalizing the families model, we get the *diagrams model*, in which contexts are interpreted as groupoids, and cartesian types over a groupoid Γ are diagrams in $\mathbf{Gpd}$ over Γ, and linear types over Γ are diagrams in a given symmetric monoidal category $\mathcal{V}$ over Γ. Just as the groupoid model [HS98] can be shown to support a univalent universe, we construct a linear analogue of the univalence axiom and show that it holds in the diagrams model if the adjunction $L \dashv M$ factors through sets.

We will be assuming a certain familiarity with basic category theory concepts like limits, adjoints and monoidal categories. For a good introduction to these notions, the reader is directed to [Awo10].

The structure of this paper is as follows: after reviewing some theory on fibrations, monoidal categories, higher and enriched categories in the Section 2, we outline the syntax of our linear dependent type theory in Section 3. Section 4 first defines a minimal categorical framework (*the semantic core*) necessary for interpreting the structural rules of the theory, dealing with weakening, context extension and substitution once and for all, and then proceeds by specifying the additional conditions imposed in order to support various type formers.

We then turn our direction to more concrete models in Section 5, and reexamine these conditions in these particular settings. We end with some discussion on the particularities of linear dependent type theory: the new elimination rules it introduces and its identity types and possible higher dimensional nature.

As our theory is both semantically and syntactically motivated, emerging from the dialectic between its semantic interpretation and its independent (syntactic) utility as a deductive system, a reader familiar with both category theory and type theory might benefit from jumping between sections to understand each indivual type former both from a semantic and syntactic perspective.

2 Preliminaries

2.1 Dependent type theory

Dependent types are types in which terms from other types may appear freely. The canonical example is the type $Vec(n)$, of vectors of length $n : \mathbb{N}$. This general class of deductive frameworks was pioneered by Per Martin Löf [ML84] as a foundation for constructive mathematics. For a thorough introduction to the syntax and semantics of dependent types, the reader may refer to [Hof97]. Here, we only present three short intuitions on how dependent type theory can be understood:

From a syntactic perspective, a *type* is a syntactic object whose meaning and function stems from the ways in which it interacts with the derivation rules of a particular *type theory*. A definition of this kind is the best we can hope for if

we want to view type theory as a foundational system of mathematical reasoning, as it does not rely on any predefined semantic framework. Taking a foundational view, the question to ask is not *what is a type?* but rather, for any other mathematical construction X *how do I express* X*, type theoretically?*.

Relaxing the foundational viewpoint, assuming the words *sets*, *propositions* and *logic* already have meaning to us, we can understand type theory as an alternative approach to higher order logic, in which we carry along "proof terms", or witnesses, for all proven propositions. In this setting, a first approximation of what a type is can be as the simultaneous generalization of a proposition and a set. This generalization allows us to understand the Π-type as both a universal quantifier and as an implication, and the Σ-type as both an existential quantifier and $\wedge$. This makes type theory a higher order system of deduction with penchant for a computational interpretation on the nature of proofs.

A third way to understand dependent type theory is as a programming language, forming a unified framework for computation and their specification. This connection goes under the name of the *Curry–Howard correspondence*, or the paradigm of "proposition as types". The computation rules of our theory provides the dynamics of our language by way of reducing terms to express them in a canonical form, while our typing rules ensure that the output has the appropriate format.

2.2 Linear type theory

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

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

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 modeling 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 2.1. 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 changing the antecedent, which does not make sense under the resource interpretation of linear logic.

Just as inuitionistic 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.

Semantically, the generalization of a linear logic to linear type theory corresponds to the generalization of symmetric monoidal preorders to symmetric monoidal categories.

Interest in linear type theory stems from disparate sources. From the perspective of (classical) computer science, it can be used for modeling state and storage, with linear variables denoting resources like pointers or files [KPB15], or even as a theoretical description of the resource handling of a blockchain [Mer15]. In quantum physics, linear types respect the no cloning theorem of quantum states, and can be used to model quantum computation [D$^+$06]. From a semantic perspective, linear type theory can be interpreted in a symmetric monoidal closed categories, generalizing the structure of a cartesian closed categories in which (non-dependent) type theories are interpreted.

2.3 Fibrations

A general heuristic for modeling a mathematical object E dependening on another object B is to specify a *projection* morphism $p : E \to B$ 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 $p : E \to I$ 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 $p : E \to B$ to be able to *lift* certain structure of the base B into *fibers* $E_b = p^{-1}(b)$ for $b \in B$.

2.3.1 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 $p : E \to B$ 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 2.2 (Cartesian arrow)**.** Given a functor $p : E \to B$, an arrow $f : e' \to e''$ of E is said to be **cartesian** with respect to p if for every $h : e \to e''$ and $\alpha : p(e) \to p(e')$ such that $p(f)\alpha = p(h)$, there is a unique arrow $\hat{\alpha} : e \to e'$ such that $p(\hat{\alpha}) = \alpha$ and $f\hat{\alpha} = h$.

The situation is illustrated in the following diagrams:

Definition 2.3 (Grothendieck fibration)**.** A functor $p : E \to B$ is a **Grothendieck fibration** if, for every $e \in E$, $b \in B$ and arrow $f : b \to p(e)$ in B, there exists a cartesian arrow $\hat{f}$ in E such that $p(\hat{f}) = f$.

We will refer to a cartesian arrow $\hat{f}$ for which $p(\hat{f}) = f$ as a **lift** of f. Although lifts are not uniquely determined, their domain will determined up to unique isomorphism. A basic review of Grothendieck fibrations particularly relevant to the semantics of dependent type theory can be found in [Jac93]. We will simply repeat the basic notions that will 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 $p : E \to B$ induces for every $u : b \to b'$ in B a *reindexing functor* $u^* : E_{b'} \to E_b$, defined by sending every object to the domain of a lift of u. Such functors will be unique up to unique natural isomorphism. In general, for two compatible morphisms u and v in the base, the reindexing functor of a composition is not identical to the composition of reindexing functors – only naturally isomorphic. In other words, we have $u^* \circ v^* \cong (u \circ v)^*$, but not functoriality on the nose.

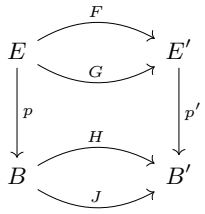
Although the definition of a fibration merely stipulates the existence of cartesian arrows, we often consider fibrations *equipped* with a collection of liftings $\{u^*\}_{u \in B}$, called a *cleavage*. Fibrations equipped with a cleavage such that the equalities $1^*_\Gamma = 1_{\mathcal{L}_\Gamma}$ and $u^* \circ v^* = (u \circ v)^*$ holds on the nose for all reindexing functors are called **split**.

Definition 2.4 (Cartesian functor)**.** Let $p : E \to B$ and $q : E' \to B$ be fibrations over the same base. A functor $F : E \to E'$ 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 **Fib** whose morphisms from fibrations $p : E \to B$ and $q : E' \to B'$ are pairs of functors (F, G) where $F : E \to E'$ and $G : B \to B'$ such that $G \circ p = q \circ F$ and F preserves cartesian morphisms. In fact, the functor $\mathbf{Fib} \to \mathbf{Cat}$ 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 fibred natural transformation, **Fib** forms a 2-category.

Definition 2.5 (Fibred 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:



a **Fibred natural transformation** (λ, λ') between (F, H) and (G, J) is a pair of natural transformations $\lambda : G \to F$ and $\lambda' : A \to B$ 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, fibred natural transformations between the parallel morphisms (F, A) and (G, A) in **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 fibred natural transformation $\lambda : F \to G$ 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 2.6. Let $p : E \to B$ and $q : D \to B$ be fibrations over the same base and $F : E \to D$ and $G : D \to E$ cartesian functors with respect to these. F is called a **fibred 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**.

2.3.2 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 2.7. A map $p : G \to H$ in **Grpd** is a **fibration of groupoids** if for every $g \in G$ and $f : p(g) \to h$ in H, there exists an object g' and map $\hat{f} : g \to g'$ in G such that $p(g') = h$ and $p(\hat{f}) = f : p(g) \to p(g')$.

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 2.8. A **monoidal fibration** is a functor $\Phi\colon E \to B$ such that

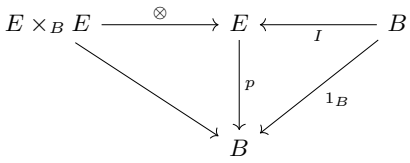
- Φ is a Grothendieck fibration
- E and B are monoidal categories and Φ is a strict monoidal functor, and
- the tensor product of E preserves cartesian arrows.

In particular, when B is a cartesian monoidal category, an arrow $f : b \to p(e)$ induces a strong monoidal functor $f^* : B_e \to B_{f^*e}$ between the fibers [Shu08].

A weaker structure is that of a lax monoidal fibration [Zaw11], which 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.

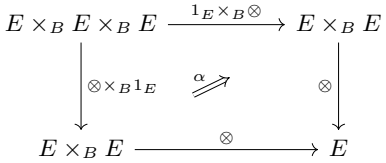
Definition 2.9. A **lax monoidal fibration** is a fibration $p : E \to B$ along with

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

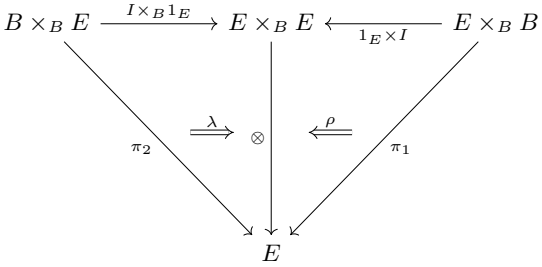


.

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



and



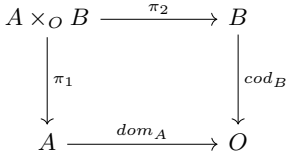
3. such that α, λ and ρ satisfies the pentagon and triangle identities in each fiber.
4. for every $b \in B$, $\rho_{I_b} = \lambda_{I_b}^{-1} : I_b \otimes I_b \to I_b$

These conditions are sufficient for each fiber to be a monoidal category and for the induced functors between fibers to be (lax) monoidal [Zaw11].

Example 2.10. An example of a fibration that is lax monoidal but not monoidal is the fibration **Gph** $\to$ **Set** taking a directed graph[1] defined by $(V, E, dom, cod : E \to V)$ 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:



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

[1] Specifically, a directed multigraph with loops allowed, also known as a quiver, defined by the *domain* and *codomain* functions from the edge set to the vertex set

2.4 Enriched and higher categories

The concept of a category can be generalized in a few different directions. In exploring semantics of linear dependent 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 2.11. Recall that a **monoidal functor** $F : \mathcal{C} \to \mathcal{D}$ is a functor equipped with a natural transformation $\mu : F \times F \to F$ with components $\mu_{A,B} : \mathcal{F}(A) \otimes_{\mathcal{D}} \mathcal{F}(B) \to \mathcal{F}(A \otimes_{\mathcal{C}} B)$ and a morphism $I_{\mathcal{D}} \to \mathcal{F}(I_{\mathcal{C}})$. Satisfying the following naturality conditions:

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

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

   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:

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

   and

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

   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 2.12. 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 $M_{a,b,c} : V_{b,c} \otimes V_{a,b} \to V_{a,c}$ and
4. an identity element $j_a : I \to V_{a,a}$

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

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

$$\begin{array}{ccccc}
I \otimes V_{a,b} & \xrightarrow{l} & V_{a,b} & \xleftarrow{r} & V_{a,b} \otimes I \\
\downarrow{\scriptstyle j_b \otimes id} & \nearrow {\scriptstyle M_{a,b,b}} & & \nwarrow {\scriptstyle M_{a,a,b}} & \downarrow{\scriptstyle id \otimes j_a} \\
V_{b,b} \otimes V_{a,b} & & & & V_{a,b} \otimes V_{a,a}
\end{array}$$

where a, 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 $\mathcal{V} = Set$ 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, [Kel05].

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 2.13. 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 $C_{a,b}$, 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 $1_a : \mathbf{1} \to \mathcal{C}_{a,a}$,
- for any triple of objects $a, b, c \in \mathcal{C}_0$, a functor $M_{a,b,c} : C_{b,c} \times C_{a,b} \to C_{a,c}$ satisfying the associativity and identity diagrams from the definition of enriched categories.

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

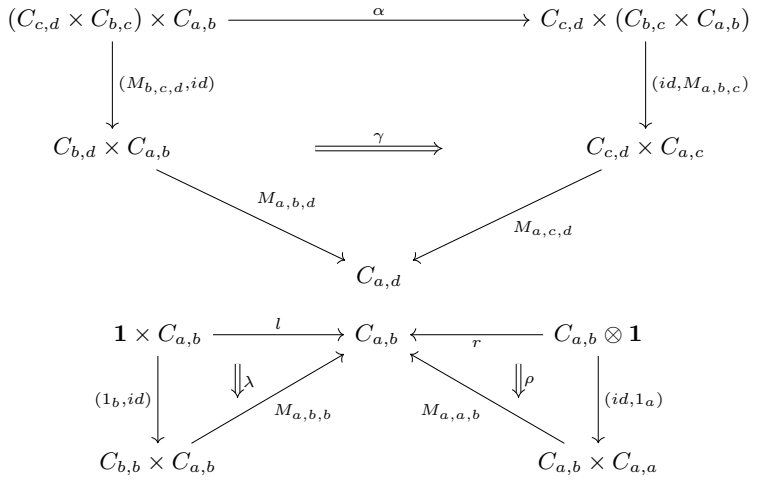
$$(\beta' \circ \beta)(\alpha' \circ \alpha) = (\beta'\alpha') \circ (\beta\alpha)$$

The road from strict 2-categories to weak 2-categories is marched to the category theoretic mantra that "it is undesirable to speak of equality between objects in a category". In the presence of 2-morphisms, the same becomes true for 1-morphisms, as we are given a way to relate 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:

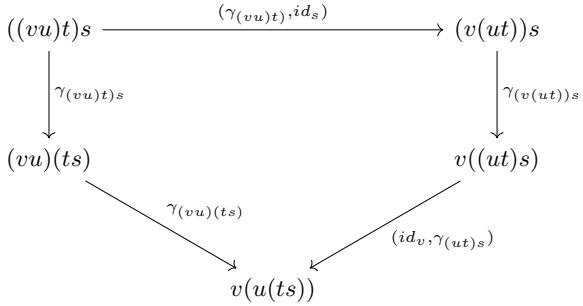
Definition 2.14. A **weak 2-category** or **bicategory** $\mathcal{C}$ is a collection of objects, 1-morphisms and 2-morphisms with composition, $M_{a,b,c} : C_{b,c} \times C_{a,b}$, and identity $1_a : \mathbf{1} \to \mathcal{C}_{a,a}$ functors as before, and for all $a, b, c, d \in \mathcal{C}_0$, natural isomorphisms:

$$\begin{aligned} \gamma : M_{a,b,d}(M_{b,c,d}, 1_{C_{a,b}}) &\implies M_{a,c,d}(1_{C_{c,d}}, M_{a,b,c})a \\ \lambda_{a,b} : l &\implies M_{a,b,b}(1_b, id) \\ \rho_{a,b} : r &\implies M_{a,a,b}(id, 1_a) \end{aligned}$$

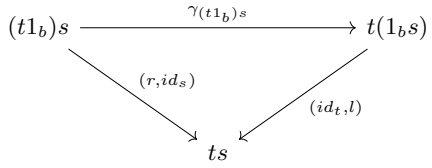
in other words, associated with the diagram of functors:



subject to the following coherence conditions. For all 1-morphisms: $s \in \mathcal{C}_{a,b}$, $t \in \mathcal{C}_{b,c}$, $u \in \mathcal{C}_{c,d}$, $v \in \mathcal{C}_{d,e}$, the following diagram in $\mathcal{C}_{a,e}$,



and the following diagram in $\mathcal{C}_{a,c}$,



commutes.

2.5 Monoidal structure of the category of symmetric monoidal categories

In section 5.3, we will consider the category of small symmetric monoidal categories, and the fact that there is a particular monoidal structure on this category itself. That is, for two symmetric monoidal categories $\mathcal{A}$ and $\mathcal{B}$, we will define a symmetric monoidal category $\mathcal{A} \otimes \mathcal{B}$. Here we 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 [Sch07].

Definition 2.15. For $\mathcal{A}, \mathcal{B} \in$ **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:

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

subject to a variety of coherence conditions, as outlined in [Sch07].

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:

$$(a_1, b) + (a_2, b) \sim (a_1 + a_2, b)$$
$$(a, b_1) + (a, b_2) \sim (a, b_1 + b_2)$$

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 $Ten :$ **SMCat** $\times$ **SMCat** $\to$ **SMCat**.

Definition 2.16. 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 functors Ten and I do not technically form a monoidal structure on **SMCat**. The associativity, left and right unitor and symmetries are all functors, $A : (\mathcal{A} \otimes \mathcal{B}) \otimes \mathcal{C} \to \mathcal{A} \otimes (\mathcal{B} \otimes \mathcal{C})$ equipped with corresponding functors, $A' : \mathcal{A} \otimes (\mathcal{B} \otimes \mathcal{C}) \to (\mathcal{A} \otimes \mathcal{B}) \otimes \mathcal{C}$, but A and A' are not "on the nose" inverses to each other. Rather, the roundabouts $A \circ A'$ and $A' \circ A$ are naturally isomorphic to the identity. One might call this a lax symmetric monoidal 2-categorical structure on **SMCat**.

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

As a final note, which will become important in section 5.3, 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.

3 Syntax

The particular linear dependent type theory under consideration is a simplified version of the work of Krishnaswami in [KPB15], with slight syntactic changes motivated by our semantic interpretation. Types are either *cartesian*, in which case we simply write $\Gamma \vdash A$ type, or *linear*, written $\Gamma \vdash A$ linear. When making typing judgments 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.

3.1 Structural rules

We will be dealing with the following judgments:

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

Figure 1: Judgments of linear dependent type theory

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

$$\frac{}{\vdash \cdot \text{ ctxt}} \text{ CI-Base} \qquad \frac{}{\vdash \cdot; \cdot \text{ ctxt}} \text{ CM-Base}$$

$$\frac{\Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A \text{ ctxt}} \text{ C-int-ext}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \vdash \Gamma, \Delta \text{ ctxt}}{\vdash \Gamma, x : A, \Delta \text{ ctxt}} \text{ C-weak-1}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \vdash \Gamma, \Delta; \Xi \text{ ctxt}}{\vdash \Gamma, x : A, \Delta; \Xi \text{ ctxt}} \text{ C-weak-2}$$

$$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma \vdash A \text{ linear}}{\vdash \Gamma; \Xi, x : A \text{ ctxt}} \text{ C-lin-ext}$$

$$\frac{\Gamma; \Xi, x : A, y : B, \Xi' \vdash t : A'}{\Gamma; \Xi, y : B, x : A, \Xi' \vdash t : A'} \text{ Lin-exch}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, \Gamma'; \Xi \vdash t : A'}{\Gamma, x : A, \Gamma'; \Xi \vdash t : A'} \text{ Weak-L}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, \Gamma' \vdash \mathcal{J}}{\Gamma, x : A, \Gamma' \vdash \mathcal{J}} \text{ Weak-I}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma, x : A, \Gamma' \vdash \mathcal{J}}{\Gamma, \Gamma'[M/x] \vdash \mathcal{J}[M/x]} \text{ Int-subst-1}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma, x : A, \Gamma'; \Xi \vdash t : A'}{\Gamma, \Gamma'[M/x]; \Xi[M/x] \vdash t : A'[M/x]} \text{ Int-subst-2}$$

$$\frac{\Gamma; \Xi, x : A \vdash t : B \quad \Gamma; \Xi' \vdash M : A}{\Gamma; \Xi, \Xi' \vdash t[M/x] : B} \text{ Lin-subst}$$

$$\frac{\Gamma, x : A, \Gamma' \text{ ctxt}}{\Gamma, x : A, \Gamma' \vdash x : A} \text{ Int-var}$$

$$\frac{\vdash \Gamma; x : A \text{ ctxt}}{\Gamma; x : A \vdash x : A} \text{ Lin-var}$$

Figure 2: Structural rules

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

3.2 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.

$$\frac{\Gamma \vdash A \ \text{type} \quad \Gamma, x : A \vdash B \ \text{type}}{\Gamma \vdash \Pi_{x:A} B \ \text{type}} \ \Pi\text{-F}$$

$$\frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x.b : \Pi_{x:A} B} \ \Pi\text{-I}$$

$$\frac{\Gamma \vdash t : \Pi_{x:A} B \ \text{type} \quad \Gamma \vdash M : A}{\Gamma \vdash t(M) : B[M/x]} \ \Pi\text{-E}$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash M : A}{\Gamma \vdash \lambda x.b(M) \equiv b[M/x] : B[M/x]} \ \Pi\text{-C}$$

$$\frac{\Gamma \vdash A \ \text{type} \quad \Gamma, x : A \vdash B \ \text{type}}{\Gamma \vdash \Sigma_{x:A} B \ \text{type}} \ \Sigma\text{-F}$$

$$\frac{\vdash \Gamma \ \text{ctxt}}{\Gamma \vdash 1 \ \text{type}} \ 1\text{-F}$$

$$\frac{\Gamma \ \text{ctxt}}{\Gamma \vdash \star : 1} \ 1\text{-I}$$

$$\frac{\Gamma, x : 1 \vdash C \ \text{type} \quad \Gamma \vdash c : C[\star/x] \quad \Gamma \vdash M : 1}{\Gamma \vdash \hat{c}[M] : C[M/x]} \ 1\text{-E}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B[M/x]}{\Gamma \vdash (M, N) : \Sigma_{x:A} B} \ \Sigma\text{-I}$$

$$\frac{\begin{array}{c}\Gamma, t : \Sigma_{x:A} B \vdash C \ \text{type} \\ \Gamma, x : A, y : B \vdash c : C[(x, y)/t] \\ \Gamma \vdash s : \Sigma_{x:A} B\end{array}}{\Gamma \vdash \hat{c}[s] : C[s/t]} \ \Sigma\text{-E}_1$$

$$\frac{\begin{array}{c}\Gamma, t : \Sigma_{x:A} B \vdash C \ \text{linear} \\ \Gamma, x : A, y : B; \Xi \vdash c : C[(x, y)/t] \\ \Gamma \vdash s : \Sigma_{x:A} B\end{array}}{\Gamma; \Xi[pr_1(s)/x][pr_2(s)/y] \vdash \hat{c}[s] : C[s/t]} \ \Sigma\text{-E}_2$$

$$\frac{\Gamma \vdash \hat{c}[(a, b)] : C[(a, b)/t]}{\Gamma \vdash \hat{c}[(a, b)] \equiv c[(a, b)/t] : C[(a, b)/t]} \ \Sigma\text{-C}_1$$

$$\frac{\Gamma; \Xi \vdash \hat{c}[(a, b)] : C[(a, b)/t]}{\Gamma; \Xi \vdash \hat{c}[(a, b)] \equiv c[(a, b)/t] : C[(a, b)/t]} \ \Sigma\text{-C}_2$$

$$\frac{\Gamma \vdash \hat{c}[\star] : C[\star/x]}{\Gamma \vdash \hat{c}[\star] \equiv c[\star/x] : C[\star/x]} \ 1\text{-C}$$

Figure 3: (Cartesian) dependent sum and product types

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : A}{\Gamma \vdash M =_A N \ \text{type}} \ =\text{-F}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash refl(M) : M =_A M} \ =\text{-I}$$

$$\frac{\begin{array}{c}\Gamma, x, y : A, p : x =_A y \vdash C \ \text{type} \\ \Gamma, z : A \vdash c : C[z/x, z/y, refl(z)/p] \\ \Gamma \vdash M : A \quad \Gamma \vdash N : A \\ \Gamma \vdash P : M =_A N\end{array}}{\Gamma \vdash R^{Id}_{[x,y,p]}(c, M, N, P) : C[M/x, N/y, P/p]} \ =\text{-E}_1$$

$$\frac{\begin{array}{c}\vdash \Gamma, x, y : A, p : x =_A y; \Xi \text{ctxt} \\ \Gamma, x, y : A, p : x =_A y \vdash C \ \text{linear} \\ \Gamma, z : A; \Xi[z/x, z/y, refl(z)/p] \vdash c : C[z/x, z/y, refl(z)/p] \\ \Gamma \vdash M : A \\ \Gamma \vdash N : A \\ \Gamma \vdash P : M =_A N\end{array}}{\Gamma; \Xi[M/x, N/y, P/p] \vdash R^{Id}_{[x,y,p]}(c, M, N, P) : C[M/x, N/y, P/p]} \ =\text{-E}_2$$

$$\frac{\Gamma \vdash R^{Id}_{[x,y,p]}(c, M, M, refl(M)) : C[M/x, M/y, refl(M)/p]}{\Gamma \vdash R^{Id}_{[x,y,p]}(c, M, M, refl(M)) \equiv c[M/z] : C[M/x, M/y, refl(M)/p]} \ =\text{-C}_1$$

$$\frac{\Gamma; \Xi[M/x, M/y, refl(M)/p] \vdash R^{Id}_{[x,y,p]}(c, M, M, refl(M)) : C[M/x, M/y, refl(M)/p]}{\Gamma; \Xi[M/x, M/y, refl(M)/p] \vdash R^{Id}_{[x,y,p]}(c, M, M, refl(M)) \equiv c[M/z] : C[M/x, M/y, refl(M)/p]} \ =\text{-C}_2$$

Figure 4: (Cartesian) identity type

3.3 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 $\Xi :\equiv x_1 : A_1, x_2 : A_2, \ldots x_n : A_n$ and $x_1 \otimes x_2 \otimes \ldots x_n : A_1 \otimes A_2 \otimes \ldots A_n$.

$$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash A \otimes B \text{ linear}} \otimes\text{-F} \qquad \frac{}{\Gamma \vdash I \text{ linear}} \text{I-F}$$

$$\frac{\Gamma; \Xi \vdash a : A \quad \Gamma; \Xi' \vdash b : B}{\Gamma; \Xi, \Xi' \vdash (a, b) : A \otimes B} \otimes\text{-I} \qquad \frac{}{\Gamma; \cdot \vdash * : I} \text{I-I}$$

$$\frac{\Gamma; \Xi' \vdash t : A \otimes B \quad \Gamma; \Xi, x : A, y : B \vdash c : C}{\Gamma; \Xi, \Xi' \vdash \text{let } x, y \text{ be } t \text{ in } c : C} \otimes\text{-E} \qquad \frac{\Gamma; \Xi \vdash c : C \quad \Gamma; \Xi' \vdash t : I}{\Gamma; \Xi, \Xi' \vdash \text{ let } * \text{ be } t \text{ in c : C}} \text{I-E}$$

$$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (a, b) \text{ in } c : C}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (a, b) \text{ in } c \equiv c[a/x][b/y] : C} \otimes\text{-C} \qquad \frac{\Gamma; \Xi \vdash \text{ let } * \text{ be } * \text{ in } c : C}{\Gamma; \Xi \vdash \text{ let } * \text{ be } * \text{ in } c \equiv c : C} \text{I-C}$$

Figure 5: Linear $\otimes$ and I type formers

The typing rules for the remaining linear rules are standard.

$$\frac{\Gamma \vdash A \ \text{linear} \quad \Gamma \vdash B \ \text{linear}}{\Gamma \vdash A \multimap B \ \text{linear}} \ \multimap\text{-F}$$

$$\frac{\Gamma; \Xi, x : A \vdash b : B}{\Gamma; \Xi \vdash \lambda x.b : A \multimap B} \ \multimap\text{-I}$$

$$\frac{\Gamma; \Xi \vdash f : A \multimap B \quad \Gamma; \Xi' \vdash a : A}{\Gamma; \Xi, \Xi' \vdash f(a) : B} \ \multimap\text{-E}$$

$$\frac{\Gamma; \Xi \vdash \lambda x.b(a) : B}{\Gamma; \Xi \vdash \lambda x.b(a) \equiv b[a/x] : B} \ \multimap\text{-C}$$

$$\frac{\Gamma \vdash A \ \text{linear} \quad \Gamma \vdash B \ \text{linear}}{\Gamma \vdash A \& B \ \text{linear}} \ \&\text{-F}$$

$$\frac{\Gamma; \Xi \vdash a : A \quad \Gamma; \Xi \vdash b : B}{\Gamma; \Xi \vdash \langle a, b \rangle : A \& B} \ \&\text{-I}$$

$$\frac{\Gamma; \Xi \vdash t : A \& B}{\Gamma; \Xi \vdash \text{fst}(t) : A} \ \&\text{-E}_1$$

$$\frac{\Gamma; \Xi \vdash t : A \& B}{\Gamma; \Xi \vdash \text{snd}(t) : B} \ \&\text{-E}_2$$

$$\frac{\Gamma; \Xi \vdash \text{fst}(\langle a, b \rangle) : A}{\Gamma; \Xi \text{fst}(\langle a, b \rangle) \equiv a : A} \ \&\text{-C}_1$$

$$\frac{\Gamma; \Xi \vdash \text{snd}(\langle a, b \rangle) : B}{\Gamma; \Xi \vdash \text{snd}(\langle a, b \rangle) \equiv b : B} \ \&\text{-C}_2$$

$$\frac{\Gamma \vdash A \ \text{linear} \quad \Gamma \vdash B \ \text{linear}}{\Gamma \vdash A \oplus B \ \text{linear}} \ \oplus\text{-F}$$

$$\frac{\Gamma; \Xi \vdash a : A}{\Gamma \Xi \vdash \text{inl}(a) : A \oplus B} \ \oplus\text{-I}_1$$

$$\frac{\Gamma; \Xi \vdash b : B}{\Gamma \Xi \vdash \text{inr}(b) : A \oplus B} \ \oplus\text{-I}_2$$

$$\frac{\Gamma; \Xi, x : A \vdash c : C \quad \Gamma; \Xi, y : B \vdash d : C; \Gamma; \Xi' \vdash t : A \oplus B}{\Gamma; \Xi, \Xi' \vdash \text{case } t \text{ of inl}(x) \to c || \text{inr}(y) \to d : C} \ \oplus\text{-E}$$

$$\frac{\Gamma; \Xi \vdash \text{case inl}(a) \text{ of inl}(x) \to c || \text{inr}(y) \to d : C}{\Gamma; \Xi \vdash \text{case inl}(a) \text{ of inl}(x) \to c || \text{inr}(y) \to d \equiv c[a/x] : C} \ \oplus\text{-C}_1$$

$$\frac{\Gamma; \Xi \vdash \text{case inr}(b) \text{ of inl}(x) \to c || \text{inr}(y) \to d : C}{\Gamma; \Xi \vdash \text{case inr}(b) \text{ of inl}(x) \to c || \text{inr}(y) \to d \equiv d[b/y] : C} \ \oplus\text{-C}_2$$

$$\frac{}{\Gamma \vdash \top \ \text{linear}} \ \top\text{-F}$$

$$\frac{\vdash \Gamma; \Xi \text{ctxt}}{\Gamma; \Xi \vdash ! : \top} \ \top\text{-I}$$

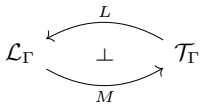
$$\frac{}{\Gamma \vdash 0 \ \text{linear}} \ 0\text{-F}$$

$$\frac{\Gamma; \Xi \vdash t : 0}{\Gamma; \Xi, \Xi' \vdash EFQ(t) : B} \ 0\text{-E}$$

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

3.4 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:



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 [Ben95], and is given an interesting new light in [LSR17], 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[2].

[2]We have not introduced universes yet, but there is a subtlety involved with how they interact with the operators M and L, which may warrant restricting the operators M and L to only act on small types. See section 4.2.7

$$\frac{\Gamma \vdash A \ \text{type}}{\Gamma \vdash A_L \ \text{linear}} \ \text{L-F}$$

$$\frac{\Gamma \vdash a : A}{\Gamma; \cdot \vdash a_L : A_L} \ \text{L-I}$$

$$\frac{\begin{array}{c}(\Gamma \vdash B \ \text{linear}) \\ (\vdash \Gamma; \Xi' \ \text{ctxt}) \\ \Gamma; \Xi \vdash y : A_L \quad \Gamma, x : A; \Xi' \vdash t : B\end{array}}{\Gamma; \Xi, \Xi' \vdash \text{let } x \text{ be } y \text{ in } t : B} \ \text{L-E}$$

$$\frac{\Gamma; \Xi \vdash \text{let } x \text{ be } s_L \text{ in } t : B}{\Gamma; \Xi \vdash \text{let } x \text{ be } s_L \text{ in } t \equiv t[s/x] : B} \ \text{L-C}$$

$$\frac{\begin{array}{c}\Gamma; y : A_L, \Xi \vdash t : B \\ (\Gamma, x : A; \Xi \vdash t[x_L/y] : B) \\ \Gamma; \Xi' \vdash a : A_L\end{array}}{\Gamma; \Xi, \Xi' \vdash \text{let } x \text{ be } a \text{ in } t[x_L/y] \equiv t[a/y] : B} \ \text{L-U}$$

$$\frac{\Gamma \vdash B \ \text{linear}}{\Gamma \vdash B_M \ \text{type}} \ \text{M-F}$$

$$\frac{\Gamma; \cdot \vdash b : B}{\Gamma \vdash b_M : B_M} \ \text{M-I}$$

$$\frac{\Gamma \vdash t : B_M}{\Gamma; \cdot \vdash \sigma(t) : B} \ \text{M-E}$$

$$\frac{\Gamma \vdash b_M : B_M}{\Gamma; \cdot \vdash \sigma(b_M) \equiv b : B} \ \text{M-C}_1$$

$$\frac{\Gamma \vdash \sigma(t)_M : B_M}{\Gamma \vdash \sigma(t)_M \equiv t : B_M} \ \text{M-C}_2$$

As a motivation for the semantic interpretation of L and M as adjoint functors given in 4.20, 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 $\lambda x.x : A \to A$, yields the identity on cartesian types, while $\lambda x.x : B \multimap B$ is the identity for linear B.

Definition 3.1. Define $\_\hat{\circ}\_ : (B \multimap C) \multimap (A \multimap B) \multimap (A \multimap C)$ and $\_ \circ \_ : (B \to C) \to (A \to B) \to (A \to C)$ by:

$$g \hat{\circ} f = \lambda x.g(f(x))$$
$$g \circ f = \lambda x.g(f(x))$$

Theorem 3.2. *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*

$$\texttt{fmap} : (A \multimap B)_M \to A_M \to B_M$$

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

Proof. Define:

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

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

Proof. Define:

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

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

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

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 4.20, we will not prove the complete statement here. We omit the naturality conditions and only prove the following:

Theorem 3.4. *There are terms t_1, t_2, t_3, t_4, relating the operators M and L so that the judgments:*

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

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:

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

□

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" $\epsilon : LM \to 1$ satisfying the following universal property:
For any $f : L(A) \to B$, there is a unique morphism $g : A \to B_M$, such that $\epsilon_B \circ L(g) = f$.
Translated into the syntax of our type theory, the statement becomes the following:

Theorem 3.5 ($L \dashv M$)**.** *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 $\epsilon : LM \to 1_{\mathcal{L}_\Gamma}$ 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 judgment:

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

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 3.6. *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 $f : A_M \times B_M \to (A\&B)_M$ by:

$$f :\equiv \lambda x.\Big(\sigma(\mathrm{pr}_1(x)), \sigma(\mathrm{pr}_2(x))\Big)_M$$

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

$$g :\equiv \lambda y.\Big(\big(\mathrm{fst}(\sigma(y))\big)_M, \big(\mathrm{snd}(\sigma(y))\big)_M\Big)$$

□

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. Furthermore, we can compare terms $\Gamma; \Xi \vdash t : A$ and $\Gamma; \Xi \vdash t' : A$ of the same type in the same context by repeated currying by forming the identity type [3] $\lambda\xi.t =_{\Xi \multimap A} \lambda\xi.t'$.

If we add extensionality to our theory so that propositional equality implies judgmental equality, then $a_M =_{A_M} b_M$ implies $\Gamma; \cdot \vdash a \equiv b : A$.

3.4.1 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.

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ linear}}{\Gamma \vdash \sqcap_{x:A} B \text{ linear}} \sqcap\text{-F} \qquad \frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ linear}}{\Gamma \vdash \sqsubset_{x:A} B \text{ linear}} \sqsubset\text{-F}$$

$$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma, x : A; \Xi \vdash b : B}{\Gamma; \Xi \vdash \lambda x.b : \sqcap_{x:A} B} \sqcap\text{-I} \qquad \frac{\Gamma \vdash s : A \quad \Gamma; \Xi \vdash b : B[s/x]}{\Gamma; \Xi \vdash (s, b) :\sqsubset_{x:A} B} \sqsubset\text{-I}$$

$$\frac{\Gamma; \Xi \vdash t : \sqcap_{x:A} B \quad \Gamma \vdash a : A}{\Gamma; \Xi \vdash t(a) : B[a/x]} \sqcap\text{-E}$$

$$\frac{\vdash \Gamma; \Xi' \text{ ctxt} \quad \Gamma \vdash C \text{ linear} \quad \Gamma; \Xi \vdash t :\sqsubset_{x:A} B \quad \Gamma, x : A; \Xi', y : B \vdash c : C}{\Gamma; \Xi, \Xi' \vdash \text{let } x, y \text{ be } t \text{ in c : C}} \sqsubset\text{-E}$$

$$\frac{\Gamma; \Xi \vdash \lambda x.b(a) : \sqcap_{x:A} B}{\Gamma; \Xi \vdash \lambda x.b(a) \equiv b[a/x] : B[a/x]} \sqcap\text{-C}$$

$$\frac{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (s, t) \text{ in c : C}}{\Gamma; \Xi \vdash \text{let } x, y \text{ be } (s, t) \text{ in c} \equiv c[s/x][t/y] : C} \sqsubset\text{-C}$$

How are we to understand these types? Either in their own right (motivated by examples), as linear analogues of Π and Σ (motivated by theorem 3.7 and 3.8), or as the type theoretic analogue of quantification in linear logic. Since the free variables occurring in linear type B in the context Γ are precisely the variables of Γ, the constructs $\sqcap$ and $\sqsubset$ can be understood as a universal or existential quantification over a variable. An example of a $\sqcap$ type, consider for any linear type A, the n-fold tensor product $A^n :\equiv A \otimes A \ldots A$, which we can define by induction over the natural numbers (using universes) via:

$$A^0 :\equiv I$$
$$A^{\mathrm{suc}\,(n)} :\equiv A \otimes A^n$$

with this construct, recall the example of burning hydrogen given in 2.1, now expressed as a linear function $burn : O_2 \otimes H_2 \otimes H_2 \multimap H_2O \otimes H_2O$. With a dependent, linear function type, we can generalize this process to the function:

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

[3] abusing notation to treat Ξ as a single type, instead of a list of types

An example of the $\sqsubset$ type comes from Krishnaswami's treatment of linear dependent logic as a way to model imperative programs [KPB15]. Here, a primitive type of memory locations, $\Gamma \vdash \mathit{Loc}$ 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 $[x \mapsto A]$:

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

Given a location $a : \mathit{Loc}$ 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:

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

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:

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

with terms introduced by:

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

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

Theorem 3.7. *For all $\Gamma \vdash A$ type and $\Gamma, x : A \vdash B$ linear, there is an isomorphism:*

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

Proof. Construct the function

$$h :\equiv \lambda f.\big(\lambda x.\sigma(f(x))\big)_M : \Pi_{x:A} B_M \to (\sqcap_{x:A} B)_M$$

and in the other direction:

$$h^{-1} :\equiv \lambda g.\lambda y.\big(\sigma(g)(y)\big)_M : (\sqcap_{x:A} B)_M \to \Pi_{x:A} B_M$$

which are mutually inverse:

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

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

□

We would like to show a similar result relating Σ and $\sqsubset$, but for this we need two additional rules. First, we assume the following uniqueness rules for Σ and $\sqsubset$ [4]:

$$\frac{\Gamma \vdash p : \Sigma_{x:A} B}{\Gamma \vdash (\mathrm{pr}_1(p), \mathrm{pr}_2(p)) \equiv p} \ \Sigma\text{-U}$$

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

Second, suppose we are given terms:

$$\begin{aligned}
\Gamma; \Xi, y : B \vdash e : C \\
\Gamma, x : A; \Xi' \vdash u : B \\
\Gamma; \Xi'' \vdash t : A_L
\end{aligned}$$

[4] The former is provable as a propositional identity, see for example [Pro13, Corollary 2.7.3]. Perhaps it is possible to obtain a similar result for $\sqsubset$, using the "surrogate equality" described in the end of Section 3.4

using the elimination rule for the L operator, we can either first combine the second and third line followed by a substitution into the first to get:

$$\Gamma; \Xi, \Xi', \Xi'' \vdash e[\text{let } x \text{ be } t \text{ in } u/y] : C$$

or weaken the cartesian context of the first line, substitute u for y and then apply L-E to get:

$$\Gamma; \Xi, \Xi', \Xi'' \vdash \text{let } x \text{ be } t \text{ in } e[u/y] : C$$

The rule Nat_L postulates that these terms are equal:

$$\frac{\begin{array}{c}\Gamma; \Xi, y : B \vdash e : C \\ \Gamma, x : A; \Xi' \vdash u : B \\ \Gamma; \Xi'' \vdash t : A_L\end{array}}{\Gamma; \Xi, \Xi', \Xi'' \vdash e[\text{let } x \text{ be } t \text{ in } u/y] \equiv \text{let } x \text{ be } t \text{ in } e[u/y] : C}\ \text{Nat}_L$$

Theorem 3.8. *Assuming the Nat_L and the uniqueness rules for Σ and $\sqcap$, there is a linear isomorphism:*

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

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

$$j :\equiv \lambda f.\text{let } p \text{ be } f \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big) : (\Sigma_{x:A} B)_L \multimap \sqcap_{x:A} B_L$$

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

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

and see that:

$$\begin{aligned}
j \circ j^{-1} :\equiv \lambda\alpha.\lambda f.\text{let } p \text{ be } f \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\big(\lambda q.\text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)\big)\alpha &\equiv && (\multimap\text{-C}) \\
\lambda\alpha.\lambda f.\text{let } p \text{ be } f \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\big(\text{let } x, y \text{ be } \alpha \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)\big) &\equiv && (\multimap\text{-C}) \\
\lambda\alpha.\text{let } p \text{ be } f \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)[\big(\text{let } x, y \text{ be } \alpha \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)\big)/f] &\equiv && (\text{Nat}_L) \\
\lambda\alpha.\text{let } x, y \text{ be } \alpha \text{ in } \Big(\text{let } p \text{ be } (\text{let } z \text{ be } y \text{ in } (x, z))_L \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\Big) &\equiv && (\text{Nat}_L) \\
\lambda\alpha.\text{let } x, y \text{ be } \alpha \text{ in } \Big(\text{let } p \text{ be } (x, \text{let } z \text{ be } y \text{ in } z)_L \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\Big) &\equiv && (\text{L-C} + \Sigma\text{-C}) \\
\lambda\alpha.\text{let } x, y \text{ be } \alpha \text{ in } \Big(x, \big(\text{let } z \text{ be } y \text{ in } z\big)_L\Big) &\equiv && (\text{Nat}_L) \\
\lambda\alpha.\text{let } x, y \text{ be } \alpha \text{ in } \Big(x, (\text{let } z \text{ be } y \text{ in } z_L)\Big) &\equiv && (\text{L-U}) + (\sqcap\text{-U}) \\
\lambda\alpha.\alpha. &&&
\end{aligned}$$

and in the other direction:

$$\begin{aligned}
j^{-1} \circ j :\equiv \lambda\alpha.\lambda q.\text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)\big(\lambda f.\text{let } p \text{ be } f \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\big)\alpha &\equiv && (\multimap\text{-C}) \\
\lambda\alpha.\lambda q.\text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)\big(\text{let } p \text{ be } \alpha \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\big) &\equiv && (\multimap\text{-C}) \\
\lambda\alpha.\lambda q.\text{let } x, y \text{ be } q \text{ in } (\text{let } z \text{ be } y \text{ in } (x, z)_L)[\big(\text{let } p \text{ be } \alpha \text{ in } \Big(\text{pr}_1(p), \big(\text{pr}_2(p)\big)_L\Big)\big)/q] &\equiv && (\text{Nat}_L) \\
\lambda\alpha.\text{let } p \text{ be } \alpha \text{ in } \Big(\text{let } x, y \text{ be } (\text{pr}_1(p), \text{pr}_2(p)_L) \text{ in } \big(\text{let } z \text{ be } y \text{ in } (x, z)_L\ \big)\ \Big) &\equiv && (\sqcap\text{-C}) \\
\lambda\alpha.\text{let } p \text{ be } \alpha \text{ in } \Big(\big(\text{let } z \text{ be } \text{pr}_2(p)_L \text{ in } (\text{pr}_1(p), z)_L\ \big)\ \Big) &\equiv && (\text{L-C}) \\
\lambda\alpha.\text{let } p \text{ be } \alpha \text{ in } \Big(\big(\text{pr}_1(p), \text{pr}_2(p)\big)_L\ \Big) &\equiv && (\Sigma\text{-U}) + (\text{L-U}) \\
\lambda\alpha.\alpha. &&&
\end{aligned}$$

□

As outlined in section 4.2, the semantic interpretation of the type formers Π, $\sqcap$ and Σ, $\sqcap$ are as left and right adjoints to reindexing functors respectively, which can by lemma 5.2 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.

3.5 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. This suggests that there is no need for more than one linear universe. To simplify the semantics, we also only assume one cartesian and one linear universe, rather than an infinite cumulative hierarchy, where $El(U_i) : U_{i+1}$, even though this might be more appealing from a syntactic point of view. 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*, [Hof97], or Krishnaswami's *Integrating Linear and Dependent types*, [KPB15].

$$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash U \text{ type}}\ \mathcal{U}\text{-F} \qquad \frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash L \text{ type}}\ \text{L-F}$$

$$\frac{\Gamma \vdash t : U}{\Gamma \vdash El(t) \text{ type}}\ \text{U-El-F} \qquad \frac{\Gamma \vdash t : L}{\Gamma \vdash El(t) \text{ linear}}\ \text{L-El-F}$$

$$\frac{\Gamma \vdash t : L \quad \Gamma \vdash s : L}{\Gamma \vdash t \hat{\otimes} s : L}\ \text{L-}\otimes$$

$$\frac{\Gamma \vdash t : U \quad \Gamma, x : El(t) \vdash v : U}{\Gamma \vdash \hat{\Pi}_{x:El(t)} v : U}\ \text{U-}\Pi \qquad \frac{\Gamma \vdash t : L \quad \Gamma \vdash s : L}{El(t \hat{\otimes} s) \equiv El(s) \otimes El(t)}\ \text{L-}\otimes\text{-Ty}$$

$$\frac{\Gamma \vdash t : U \quad \Gamma, x : El(t) \vdash v : U}{\Gamma \vdash El(\hat{\Pi}_{x:El(t)} v) \equiv \Pi_{x:El(t)} El(v)}\ \text{U-}\Pi\text{-Ty} \qquad \frac{\Gamma \vdash t : U \quad \Gamma, x : El(t) \vdash s : L}{\Gamma \vdash \hat{\sqcap}_{x:El(t)} s : L}\ \text{L-}\sqcap$$

$$\frac{\Gamma \vdash t : U \quad \Gamma, x : El(t) \vdash s : L}{\Gamma \vdash El(\hat{\sqcap}_{x:El(t)} s) \equiv \sqcap_{x:El(t)} El(s)}\ \text{L-}\sqcap\text{-Ty}$$

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

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

or

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

respectively. It also allows us to formulate the type of "pointed linear types":

$$\Gamma \vdash \sqsubset_{A:\mathcal{L}} A \text{ linear}$$

4 Semantics

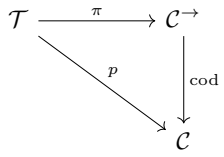
4.1 Structural semantic core

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 [Jac93], consisting of a base category of contexts equipped with a fibration of cartesian types over them, and a lax symmetric monoidal fibration consisting of linear types.

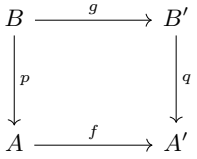
Definition 4.1. A **comprehension category** consists of a commutative diagram of functors



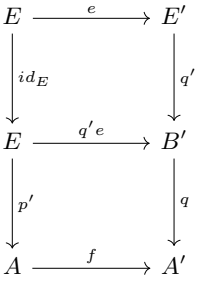
where $B^{\to}$ is the arrow category of B and $cod : B^{\to} \to B$ denotes the codomain functor, such that:

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

Notice that by the second condition, cartesian morphisms in $\mathcal{T}$ are mapped to pullback squares in $\mathcal{C}$:
A cartesian morphism $(p, q) : f \to g$ in $\mathcal{C}^{\to}$ is a commutative square in $\mathcal{C}$

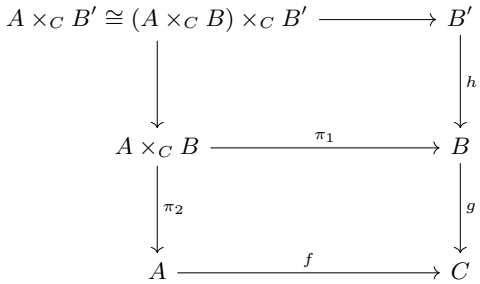


such that for any E', E, $q' : E' \to B'$, $e : E \to E'$ and $p' : E \to A$ as in the following diagram:



there is a unique arrow $u : E \to B$ 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 cod : $\mathcal{C}^{\to} \to \mathcal{C}$ 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 4.2. 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" [Hof95]. 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.



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 one reason why we ask for our comprehension category to come equipped with a collection of liftings such that the fibration is *split*. In fact, every comprehension category is equivalent to a split one, and if a full comprehension category supports various type formers, then these will also be supported in the corresponding split version. [LW15]

In dealing with linear dependent type theory, we run into another problem of similar nature. Equipped with a (lax) monoidal fibration, $q : \mathcal{L} \to \mathcal{C}$, 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 $[[\Xi]] = [[A_1]] \otimes [[A_2]] \cdots \otimes [[A_n]]$ in the monoidal category $\mathcal{L}_\Gamma$. The weakening of a linear context Ξ, as given by the rule:

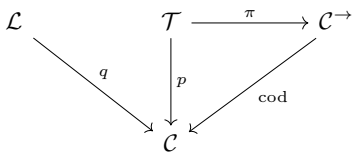
$$\frac{\vdash \Gamma; \Xi \text{ ctxt} \quad \Gamma \vdash A \text{ type}}{\vdash \Gamma, x : A; \Xi \text{ ctxt}} \text{ C-Weak}$$

will be interpreted as the image of the object $[[\Xi]] \in \mathcal{L}_\Gamma$ under the functor $\pi_A^* : \mathcal{L}_\Gamma \to \mathcal{L}_{\Gamma.A}$ induced by the projection morphism $\pi_A : \Gamma.A \to \Gamma$ 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 4.3. A **split monoidal fibration** is a lax monoidal fibration (Definition 2.9) $q : \mathcal{L} \to \mathcal{C}$, equipped with a cleavage $\{\hat{u}\}_{u \in \mathcal{C}}$ such that the induced reindexing functors u^* are strict monoidal, $u^* v^* = (vu)^*$ and $1_\Gamma^* = 1_{\mathcal{L}_\Gamma}$ for all compatible morphisms u and v.

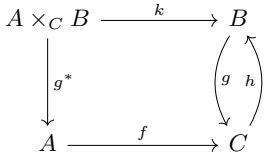
Notice that this condition is quite strong, as the induced functors of a lax monoidal fibration are generally only lax monoidal. This disqualifies the graph fibration 2.10 from constituting a model of our type theory. However, it is reasonable to expect every lax monoidal fibration where the induced functors are strong to be equivalent to a split monoidal fibration. Equipped with these notions, we define the general structure of the structural core of our type theory:

Definition 4.4. A **model for linear dependent type theory** consists of a split comprehension category $p : \mathcal{T} \to \mathcal{C}$ and a split monoidal fibration $q : \mathcal{L} \to \mathcal{C}$, as illustrated in the following picture:



Before showing how this structure supports the structural rules of the theory, the following lemma will come in handy:

Lemma 4.5. The pullback of an arrow, g, has a section if g does. In other words, given a pullback of g:



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 $u : A \to A \times_C B$ such that $hfu = k$ and $g^* u = 1_A$, making u a section of g^*. □

Below we outline how to interpret the structural rules of Figure 2 in this semantic framework.[5]

Theorem 4.6. *A split comprehension category $p : \mathcal{T} \to \mathcal{C}$ together with a split monoidal fibration $q : \mathcal{L} \to \mathcal{C}$ 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 $\pi([[A]]) : [[\Gamma, A]] \to [[\Gamma]]$.
- Linear terms $b : B$ in $\Gamma; \Xi$ to morphisms $[[b]] : [[\Xi]] \to [[B]]$.

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 $\mathcal{L}_{\mathbf{1}}$.
- Case of C-int-ext: By the induction hypothesis, we are given A in $\mathcal{T}_\Gamma$ 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 $[[\Xi]] \otimes [[A]] \in \mathcal{L}_{[[\Gamma]]}$.

[5] Due to the recursive nature of dependent type theory, the set of well-defined types and the set of well-defined sequents cannot be independently defined. Technically, when defining our interpretation of a judgment $\mathcal{J}$ by induction over the structural rules, we are actually defining a function which maps a proof tree, φ, of $\mathcal{J}$ to a semantic object. Therefore, in order for Theorem 4.6 to constitute a actual proof of soundness, we would need to show that a judgment $\mathcal{J}$ is interpreted the same way regardless of how it has been derived. The standard approach to this is to define a *partial* interpretation function for raw syntax, i.e. expressions that are not necessarily well typed, and then show that this function is total on the domain of well typed expressions. See [Str91] for details.

- Case of C-weak-1: Given $A, \Delta \in \mathcal{T}_\Gamma$, we send Δ through the functor $\pi_A^* : \mathcal{T}_\Gamma \to \mathcal{T}_{\Gamma.A}$ induced by lifting the morphism $\pi_A : \Gamma.A \to \Gamma$. The resulting object $\pi_A^*(\Delta)$ will be sent to the context $p(\pi_A^*(\Delta)) = \Gamma.A.\Delta\{\pi_*\}$ 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:

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

  so by lifting q to the monoidal functor $q^* : \mathcal{L}_{\Gamma.\Delta} \to \mathcal{L}_{\Gamma.A.\pi^*(\Delta)}$. 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 $\mathcal{L}_\Gamma$, 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 $\mathcal{L}_\Gamma$. Applying exchange to a judgment corresponds to composing with this isomorphism.
- Case of Weak-L. For any $A \in \mathcal{T}_\Gamma$ and morphism $t : \Xi \to A$ in $\mathcal{L}_{\Gamma,\Gamma'}$, we can transfer t along the functor $q_{A,\Gamma}^* : \mathcal{L}_{\Gamma,\Gamma'} \to \mathcal{L}_{\Gamma.A.\Gamma'\{\pi_A\}}$ induced by the map $q_{A,\Gamma'}$ arising from the following pullback diagram:

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

  to yield a morphism $q_{A,\Gamma'}^*(t) : q_{A,\Gamma'}^*(\Xi) \to q_{A,\Gamma'}^*(A')$.
- Case of Weak-I. The morphism $q_{A,\Gamma'}$ above also induces a functor $q_{A,\Gamma'}^* : \mathcal{T}_{\Gamma,\Gamma'} \to \mathcal{T}_{\Gamma.A.\Gamma'.\{\pi_A\}}$.
- For Int-subst-1, the judgment $\mathcal{J}$ 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:

$$M : \Gamma \to \Gamma.A$$

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

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

  where $q_{(M,\Gamma')}$ is given by the structure of the comprehension functor. We then lift $q_{(M,\Gamma')}$ to obtain a cartesian arrow $\hat{q}_{(M,\Gamma')} : \Gamma.\Gamma'\{M^*\}.B\{q_{(M,\Gamma')}^*\} \to \Gamma.A.\Gamma'.B$ which is sent to the following pullback diagram:

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

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 $\Gamma.\Gamma'[M/x] \vdash B[M/x]$ type.

Now suppose there is a section $b : \Gamma.A.\Gamma' \to \Gamma.A.\Gamma'.B$ 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 4.5, 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 $q^*_{(M,\Gamma')} : \mathcal{L}_{\Gamma.A.\Gamma'} \to \mathcal{L}_{\Gamma.\Gamma'\{M^*\}}$ 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 $t : \Xi \otimes A \to B$ and $M : \Xi' \to A$ we get a morphism $t \circ (id_\Xi \otimes M) : \Xi \otimes \Xi' \to B$. Precomposing with the isomorphism $\bigotimes \Xi, \Xi' \cong \Xi \otimes \Xi'$ yields the desired morphism $\bigotimes \Xi, \Xi' \to B$.
- Case of Int-var. For any $A \in \mathcal{T}_\Gamma$, $\Gamma.A$ together with its identity map forms a cone to the cospan:

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

so by the universal mapping property of the pullback, there is a unique morphism $v_A : \Gamma.A \to \Gamma.A.A\{\pi_A\}$ such that $\pi_{A\{\pi_A\}} v_A = 1_{\Gamma.A}$. This map is thus the interpretation of the judgment $\Gamma, x : A \vdash x : A$.

- 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 $\mathcal{L}_\Gamma$.

□

4.2 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 $\pi : \mathcal{T} \to \mathcal{C}^\to$ is full and faithful. This allows us to think of homsets $\mathcal{T}_\Gamma(A, B)$ as living over the base category projections $\mathcal{C}_\Gamma(\pi_A : \Gamma.A \to \Gamma, \pi_B : \Gamma.B \to \Gamma)$.

Definition 4.7. A **model of LDTT** is a full, split comprehension category $\pi : \mathcal{T} \to \mathcal{C}^\to$ and a split monoidal fibration $q : \mathcal{L} \to \mathcal{C}$.

It will sometimes be useful to also assume that the comprehension category has a *unit*, as this will give rise to an isomorphism $\mathcal{C}_{/\Gamma}(\Gamma.A, \Gamma.B) \cong \mathcal{T}_{\Gamma.A}(1, \pi_A^*(B))$ [Jac93].

Definition 4.8. A fibration $p : \mathcal{T} \to \mathcal{C}$ **admits a terminal object** if there is a functor $\mathbf{1} : \mathcal{C} \to \mathcal{T}$, such that for every $\Gamma \in \mathcal{C}$, $\mathbf{1}(\Gamma)$ is a terminal object of the fiber $\mathcal{T}_\Gamma$, and for every $f : \Delta \to \Gamma$, the canonical map $\alpha : f^*(\mathbf{1}(\Gamma)) \to \mathbf{1}(\Delta)$ is an isomorphism.

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

Definition 4.9. A comprehension category $\pi : \mathcal{T} \to \mathcal{C}^\to$ **has a unit** if the fibration $p : \mathcal{T} \to \mathcal{C}$ admits a terminal object functor $\mathbf{1}$ which is left adjoint to the functor $s(\pi) : \mathcal{T} \to \mathcal{C}$, 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 $\eta_A : \Gamma.A \cong \Gamma.A.1$.

4.2.1 Basic linear types

Definition 4.10. A model of LDTT **supports I-types** if, for every Γ, there is exists an object $[[I]] \in \mathcal{L}_\Gamma$, equipped with a morphism $[[*]] : I \to [[I]]$, such that for every pair of maps $t : \Xi' \to [[I]]$ and $c : \Xi \to C$, there exists a map $[[\text{let } t \text{ be } * \text{ in } c]] : \bigotimes \Xi, \Xi' \to C$ 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 $\rho(c \otimes [[t]])\alpha : \bigotimes \Xi, \Xi' \cong \Xi \otimes \Xi' \to C \otimes I \cong C$, where ρ is the right unitor for the monoidal structure of $\mathcal{L}_\Gamma$, and α is the isomorphism $\bigotimes \Xi, \Xi' \cong \Xi \otimes \Xi$. As for the identity $\rho(c \otimes 1_I)\alpha = c$, notice that $\alpha : \bigotimes \Xi \cong \Xi \otimes [[\cdot]] = \Xi \otimes I$ is given by ρ^{-1}, so we get $\rho(c \otimes 1_I) = c\rho$, from the naturality of ρ.

Definition 4.11. A model of LDTT **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 $a : \Xi \to A$, $b : \Xi' \to B$, there exists an arrow $[[(a \otimes b)]] : \bigotimes \Xi, \Xi' \to [[A \otimes B]]$ and for arrows $t : \Xi' \to [[A \otimes B]]$ and $c : \bigotimes \Xi \otimes A \otimes B \to C$, 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 LDTT 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 LDTT 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 co-complete, 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 [Mel09].

4.2.2 Π and Σ

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 4.12. A model of LDTT **supports Π-types** if, for all $A \in \mathcal{T}_\Gamma$, the induced functor $\pi_A^* : \mathcal{T}_\Gamma \to \mathcal{T}_{\Gamma.A}$ has a right adjoint $\Pi_A : \mathcal{T}_{\Gamma.A} \to \mathcal{T}_\Gamma$ satisfying the following Beck-Chevalley condition:

For all pullbacks in $\mathcal{C}$:

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

inducing the following functors between fibers:

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

The canonical natural transformation $f^*\Pi_{E'} \to \Pi_E q_{E,E'}^*$ 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 (Σ-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 4.13. A model of LDTT **supports Σ-types** if it satisfies the following:

1. For all $A \in \mathcal{T}_\Gamma$, the induced functor $\pi_A^* : \mathcal{T}_\Gamma \to \mathcal{T}_{\Gamma.A}$ 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: $\Sigma_E q^* \to f^* \Sigma_{E'}$ is a natural isomorphism, and
3. the induced map $pair_{A,B} : \Gamma.A.B \to \Gamma.\Sigma_A B$ is an isomorphism

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

This structure is sufficient to support new elimination rule (Σ-E$_2$):

Theorem 4.14. *If a model of LDTT supports Σ-types, then for every object $C \in \mathcal{L}_{\Gamma.\Sigma_A B}$ and morphism $c : \Xi \to C\{pair_{A,B}\}$ in $\mathcal{L}_{\Gamma.A.B}$ and section $s : \Gamma \to \Gamma.\Sigma_A B$, there exists a morphism $\hat{c}_s : \Xi\{(pr_1, pr_2\} \to C\{s\}$ such that given sections $a : \Gamma \to \Gamma.A$ and $b : \Gamma.A \to \Gamma.A.B$, then $\hat{c}_{(a,b)} = c\{ba\} : \Xi\{ba\} \to C\{ba\}$.*

Proof. The situation is illustrated in the following diagram:

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

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 $a : \Gamma \to \Gamma.A$ and $b : \Gamma.A \to \Gamma.A.B$, we have $c\{((pr_1, pr_2)(a, b))\} = c\{a\}\{b\}$:

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

□

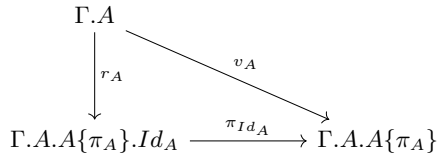
We may also speak of a model **supporting simple products**, which simply correspond to products in the fibers of $\mathcal{T}$ which are stable under reindexing functors. These are of course implied if the model supports Σ-types.

4.2.3 Identity types

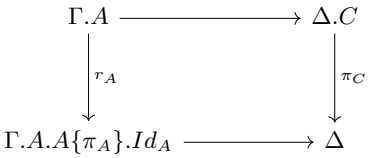
When it comes to Id-types, the situation is not as fortunate. If one wants to keep the theory intensional, we need to add condition (3) to make sure that the semantic identity types satisfy the added elimination and computation rule, $=$-E_2 and $=$-C_2 in Figure 4.

Definition 4.15 (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 $r_A : \Gamma.A \to \Gamma.A.A\{\pi_A\}.Id_A$ such that:

1. The following diagram commutes:



2. For any commutative diagram:



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

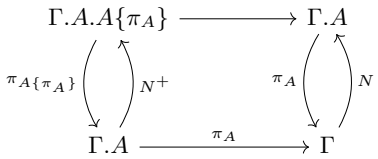
3. For any pair of objects, $C, \Xi \in \mathcal{L}_{\Gamma.A.A^+.Id}$, sections $M, N : \Gamma \to \Gamma.A$, $P : \Gamma \to \Gamma.Id_A\{N^+\}\{M\}$, and morphism $c : \Xi\{r_A\} \to C\{r_A\}$, there exists a morphism $\hat{c}_{[M,N,P]} : \Xi\{P^+\}\{N^+\}\{M\} \to C\{P^+\}\{N^+\}\{M\}$ such that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

If one were to work within an extensional type theory, where judgmental and propositional equality coincide, the third condition is always satisfied:

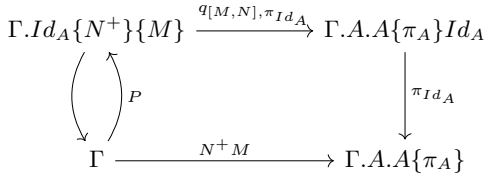
Definition 4.16. A model of linear dependent type theory supports **extensional identity types** if there are objects Id_A and morphisms r_A satisfying condition (1) and (2) such that for sections $M, N : \Gamma \to \Gamma.A$ the existence of a section $P : \Gamma \to \Gamma.Id_A\{M\}\{N^+\}$ implies $N = M$.

Theorem 4.17. *Support for extensional identity types implies support for identity types, in other words, extensional identity types always satisfy condition (3) of definition 4.15*

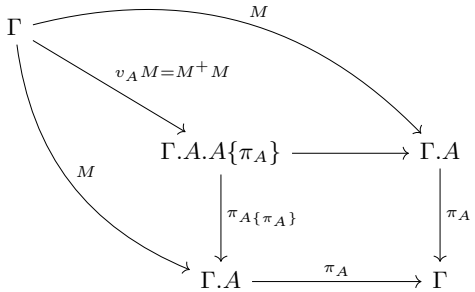
Proof. We will show that when $M = N$, the morphism $P^+N^+M : \Gamma \to \Gamma.A.A\{\pi_A\}.Id_A$ is equal to $r_AM : \Gamma \to \Gamma.A.A\{\pi_A\}.Id_A$. This implies that the morphism $\hat{c}_{[M,N,P]} : \Xi\{P^+\}\{N^+\}\{M\} \to C\{P^+\}\{N^+\}\{M\}$ of definition 4.15 really is just $c\{M\} : \Xi\{M\}\{r_A\} \to C\{M\}\{r_A\}$.
The section $N^+ : \Gamma.A \to \Gamma.A.A\{\pi_A\}$ appearing arises as in lemma 4.5 from the following pullback:



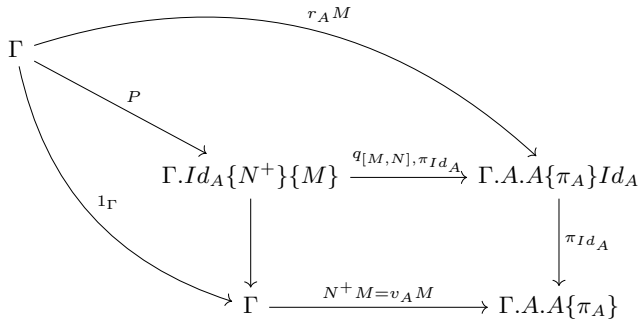
while $P^+N^+M : \Gamma \to \Gamma.A.A\{\pi_A\}.Id_A$ the composite $q_{[M,N],\pi_{Id_A}} \circ P$ in the following commutative square:



But with extensional identity types, we have $M = N$. This implies that $v_AM = M^+M$, as these are both maps from Γ into the pullback $\Gamma.A.A\{\pi_A\}$ such that $\pi_{A\{\pi_A\}}v_AM = \pi_{A\{\pi_A\}}M^+M = M$ as illustrated below:



This makes Γ a cone to the previous pullback:



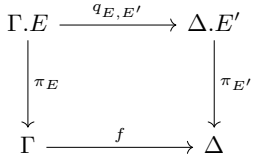
forcing $r_A M = PN^+M$. □

4.2.4 ⊓- and ⊏-types

The semantic type formers for the linear dependent $\sqcap$ and $\sqsubset$ 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 4.18. A model of LDTT **supports ⊓-types** if, for all $A \in \mathcal{T}_\Gamma$, the induced monoidal functor $\pi_A^* : \mathcal{L}_\Gamma \to \mathcal{L}_{\Gamma.A}$ has a monoidal right adjoint, $\sqcap_A$ satisfying the following Beck-Chevalley condition:

For all pullback squares in $\mathcal{C}$ of the following form:



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

Definition 4.19. It **supports ⊏-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 $\sqsubset_E\ q^* \to f^*\ \sqsubset_{E'}$ 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 $\sqsubset_A\ (\Xi\{\pi_A\} \otimes B) \to \Xi \otimes \sqsubset_A\ B$ is an isomorphism.

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

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

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

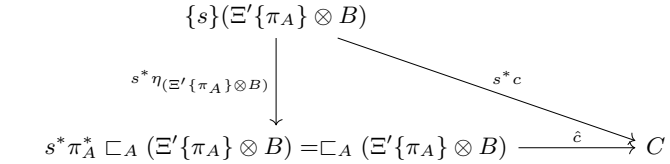
where $\eta_B : B \to \pi^*(\sqsubset_A\ B)$ is the unit of the adjunction. This map is the interpretation of the term (s, b) given by the introduction rule $\sqsubset$-I.

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

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

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 $\alpha' : \bigotimes(\Xi', \Xi) \to \Xi' \otimes \Xi$ to get a map from $\bigotimes \Xi', \Xi$ to C, which will be the interpretation of the elimination term from the rule $\sqsubset$-E.

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



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

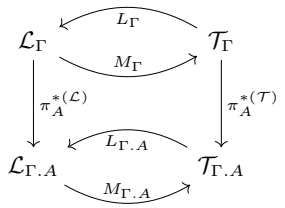
4.2.5 The operators M and L

Definition 4.20. A model of LDTT with unit **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 $p : \mathcal{T} \to \mathcal{C}$ and $q : \mathcal{L} \to \mathcal{C}$, 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:



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 4.21. *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:

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

□

4.2.6 Universes

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 4.22. A model of LDTT **supports cartesian universes**, if for every $\Gamma \in \mathcal{C}$, there exists objects $U_\Gamma \in \mathcal{T}_\Gamma$ and $El_\Gamma \in \mathcal{T}_{\Gamma.U_\Gamma}$, such that these are stable under reindexing, i.e. for any $f : \Delta \to \Gamma$ in $\mathcal{C}$, we have $f^*U_\Gamma = U_\Delta$ and $q^*_{(\pi_{U_\Gamma}, f)}(El_\Gamma) = El_\Delta$. For a section $t : \Gamma \to \Gamma.U$ to the corresponding projection, we write $El(t)$ for $t^*(El)$.

For a model that supports the type former in question, a cartesian universe is closed under:

- Π-types, (or Σ) if, for morphisms $t : \Gamma \to \Gamma.U$ and $v : \Gamma.El(t) \to \Gamma.El(t).U$, there is a section $\hat{\Pi}_{El(t)}v : \Gamma \to \Gamma.U$ 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 $A : \Gamma \to \Gamma.U$, $a, b : \Gamma \to \Gamma.El(A)$, there is a section $\hat{Id}(a, b) : \Gamma \to \Gamma.U$, 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 4.23. A model of LDTT **supports linear universes**, if, for every $\Gamma \in \mathcal{C}$, there are objects $L_\Gamma \in \mathcal{T}_\Gamma$ and $El_\Gamma \in \mathcal{L}_{\Gamma.L_\Gamma}$, stable under reindexing.

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 $\hat{I} : \Gamma \to \Gamma.L$, such that $El(\hat{I}) = I$ in $\mathcal{L}_\Gamma$.

- $\otimes$ (or $\multimap$, $\&$): For every pair of sections $a, b : \Gamma \to \Gamma.L$, 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 4.24. Given a model supporting both cartesian and linear universes, these are closed under:

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

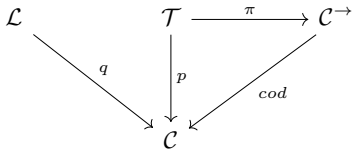
4.2.7 Universes and small type formers

We have not said anything about which types we expect to be contained in our universes. As they stand, the axioms of in section 3.5 do not postulate that there are any inhabitants of L or U to begin with. One could imagine restating the type formation rules to be defined with respect to universes, an approach exemplified by [KPB15]. As tempting as it is to think of any cartesian type being an element of U, this would imply that the code for the universe belongs to itself, which yields an inconsistency of the theory through Girard's Paradox [Hur95]. Asking for every linear type to belong to L is a more reasonable requirement, since the linear universe is a cartesian type. But doing this forces us to restrict the operators M and L so as to only act on small types.

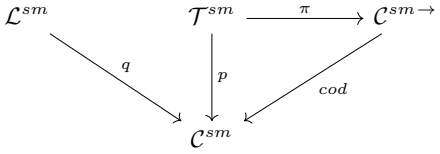
Definition 4.25. In a model that supports cartesian universes, let the category $small(\mathcal{T})$ of **small cartesian types** be the full subcategory of $\mathcal{T}$ whose objects are $El(t)$ for some section $t : \Gamma \to \Gamma.U$ in some context Γ.

Definition 4.26. Similarly, we define the category $\mathcal{L}^{sm}$ of **small linear types** to be the full subcategory of $\mathcal{L}$ whose objects are $El(t)$ for some section $t : \Gamma \to \Gamma.L$ in some context Γ.

Definition 4.27. If there is a full subcategory $\mathcal{C}^{sm}$ of $\mathcal{C}$ such that the restriction of the functors p, q and π of a model of linear dependent type theory:



to the categories $\mathcal{L}^{sm}$ and $\mathcal{T}^{sm}$ has $\mathcal{C}^{sm}$ as its codomain, so that:



is a model of linear dependent type theory, we call this a **sub-model of small types**.

We can now define what it means for a model to support small versions of various type formers.

Definition 4.28. We say that a model of linear dependent type theory **supports M and L for small types**, if, for every $\Gamma \in \mathcal{C}$, there are morphisms $L : L \iff U : M$ in $\mathcal{T}_\Gamma$ natural in Γ, inducing a fiber adjunction:

$$\mathcal{L}^{sm}(El(L(t)), El(s)) \cong \mathcal{T}^{sm}(El(t), El(M(s)))$$

such that there is an isomorphism of hom-sets:

$$\mathcal{L}^{sm}_{\Gamma.El(t)}(\pi^*_{El(t)}(El(\xi)), \pi^*_{El(t)}(El(b))) \cong \mathcal{L}^{sm}_{\Gamma}(El(Lt) \otimes El(\xi), El(b))$$

Syntactically, this corresponds to changing the type formation rules of M and L so as to not be defined for all types, but only those who are contained in the cartesian and linear universes:

$$\frac{\Gamma \vdash A : L}{\Gamma \vdash A_M : U}\ \text{M-F-U} \qquad \frac{\Gamma \vdash A : U}{\Gamma \vdash A_L : L}\ \text{L-F-U}$$

Similarly, we can define support for small $\sqcap$ and $\sqsubset$ as support for $\sqcap$ and $\sqsubset$ in a sub-model of small types.

Definition 4.29. We say that the linear universe L is **all-encompassing** if every object of $\mathcal{L}$ is also in $\mathcal{L}^{sm}$.

5 Models

5.1 Set indexed families

Our first model will be based on the standard set-theoretic interpretation of dependent type theory [Hof97]. 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 generalization of both dependent type theory and linear type theory, as outlined in [VÍ5].)

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

Definition 5.1 ($Fam(\mathcal{C})$)**.** 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 $f : S \to Ob(\mathcal{C})$. Morphisms of $Fam(\mathcal{C})$ are pairs $(u, \alpha) : (S, f) \to (S', g)$ where $u : S \to S'$ and $\alpha : S \to \mathrm{Mor}(\mathcal{C})$ such that $\alpha(s) : f(s) \to g(u(s))$ 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 $p : Fam(\mathcal{C}) \to \mathbf{Set}$, 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 $\pi : Fam(\mathcal{C}) \to \mathbf{Set}^{\to}$ as follows.

On objects, let

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

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

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

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

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

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

For a morphism $u : S \to p(S', f) = S'$ the canonical choice of a lift $u^* = (u, i) : (S, fu) \to (S', f)$ where $i : S \to \mathrm{Mor}(\mathcal{C})$ 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 $p : Fam(\mathbf{Set}) \to \mathbf{Set}$ with $\pi : Fam(\mathbf{Set}) \to \mathbf{Set}^{\to}$ 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 $u : S \to p(S', f) = S'$ 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:

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

This model trivially supports $\otimes$ and I-types (all models do). For any fibration $p : Fam(\mathcal{C}) \to \mathbf{Sets}$, 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 $f(s) \otimes g(s) \to h(s)$ is isomorphic to $f(s) \to [g(s), h(s)]$.

For the dependent types, we utilize the following result:

Theorem 5.2. *For a fibration $p : Fam(\mathcal{C}) \to \mathbf{Sets}$ and a function $u : S \to S'$ in the base, $u^* : Fam(\mathcal{C})_{S'} \to Fam(\mathcal{C})_S$ 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 $(S, f) \mapsto (S', \lambda s'. \coprod_{s \in u^{-1}(s')} f(s))$. 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 $A = \{A\}_i$ in $Fam(\mathcal{C})$. The lift of the unique map $! : p(A) \to \mathbf{1}$ in the base has a right (left) adjoint, making $\Pi_!(A)$ an object of 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 $f_i : B \to A_i$ gives rise to a unique map on the left hand side, which is precisely the universal property of the product. □

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

Theorem 5.3. *If every lift $f^* : Fam(\mathcal{C})_\Gamma \to Fam(\mathcal{C})_{\Gamma'}$ has a right adjoint R_{f^*}, defined as in Theorem 5.2 then, for all objects $A = (\Gamma, a)$ and pullbacks of the following form:*

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

there is a natural isomorphism $\eta : R_{\pi_{f^*A}} q^* \cong f^* R_{\pi_A} : Fam(\mathcal{C})_{\Gamma.A} \to Fam(\mathcal{C})_{\Gamma'}$.

Proof. Let $B = (\Gamma.A, \alpha)$ be a family of sets indexed over $\Gamma.A$. Then $R_{\pi_{f^*\{A\}}} q^*(B)$ is the family:

$$\left((\gamma' \in \Gamma)\ ,\ \lambda\gamma' \prod_{a \in f^*A(\gamma')} \alpha(f(\gamma'), a) \right)$$

while $f^* R_{\pi_A}(B)$ is the family:

$$\left((\gamma' \in \Gamma)\ ,\ \lambda\gamma' \prod_{a \in A(f(\gamma'))} \alpha(f(\gamma'), a) \right)$$

but since $f^*(A(\gamma')) = A(f(\gamma'))$ this is precisely the same thing. □

Carrying out the dual of this proof gives the corresponding Beck-Chevalley condition for left adjoints to reindexing functors. 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:

$$\{(\gamma, t) \mid \gamma \in \Gamma, t \in \coprod_{((\gamma,a),b) \in \mathrm{fst}^{-1}(\gamma,a)} \Big((\gamma, a), b\Big)$$

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 $\sqsubset$ if $\mathcal{V}$ has small coproducts that distribute over $\otimes$ (Frobenius reciprocity).

We can form extensional identity types in the families model, by interpreting propositional equality in the same way as our traditional set theoretic equality. That is, we define the family $Id_A \in Fam(\mathbf{Sets})_{\Gamma.A.A\{\pi_A\}}$ by the function:

$$Id_{A(\gamma, a \in A_\gamma, b \in A_\gamma)} = \begin{cases} \{\star\}, & \text{if } a = b \\ \emptyset, & \text{otherwise.} \end{cases}$$

This gives rise to extensional identity types, which by lemma 4.17, forms support for identity types in the families model.

To deal with the operators M and L, we make use of the following lemma:

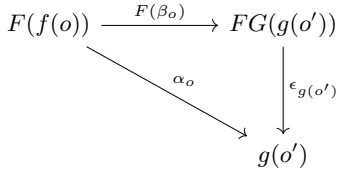
Lemma 5.4. An adjunction

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D} \quad (\perp)$$

induces a fiber adjunction between the corresponding fibrations:

$$\begin{array}{ccc} & \overset{\mathbf{F}}{\curvearrowleft} \quad (\perp) & \\ Fam(\mathcal{C}) & \xrightarrow[\mathbf{G}]{} & Fam(\mathcal{D}) \\ & {\scriptstyle p} \searrow \quad \swarrow {\scriptstyle q} & \\ & \mathbf{Sets} & \end{array}$$

Proof. Let $\mathbf{G} : Fam(\mathcal{C}) \to Fam(\mathcal{D})$ and $\mathbf{F} : Fam(\mathcal{D}) \to Fam(\mathcal{C})$ be the induced cartesian functors, defined pointwise in the obvious way. Define the vertical counit $\epsilon : \mathbf{FG} \implies 1_{Fam(\mathcal{C})}$, given at every component $\{A\}_I \in Fam(\mathcal{C})_I$ by the underlying counit, $\epsilon_A = (1_I, \epsilon_{A_i})$. For any morphism $(u, \alpha) : \mathbf{F}(O, f) \to (O', g)$, any morphism $(v, \beta) : (O, f) \to \mathbf{G}(O', g)$ 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 $\beta_o : f(o) \to G(g(o'))$ fitting into the following diagram:



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

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.

5.1.1 Universes in the families model

When it comes to universes, we need to be careful so as to not run into size issues. We want to construct universes which contains as many types as possible, without yielding any inconsistencies or collapsing the entire theory. As an example of such a collapse, consider the scenario where the category $\mathcal{V}$ is small. Then we may define our linear universe $L \in Fam(\mathbf{Sets})_\Gamma$ as the constant family

$$(\Gamma, \lambda\gamma.Ob(\mathcal{V})).$$

Since any section of $\pi(\mathcal{L}) : \Gamma.\mathcal{L} \to \Gamma$ consists of a selection of morphisms

$$f_\gamma : \mathbf{1} \to Ob(\mathcal{V}),$$

this defines an object (Γ, f) of $Fam(\mathcal{V})$ where $f(\gamma) = f_\gamma(\star)$. This universe will be closed under all purely linear type formers that are supported by $\mathcal{V}$, since these are constructed pointwise.

This may not be as desirable as it appears on first sight however, as at least in classical mathematics, a small $\mathcal{V}$ supporting $\sqcap$ in the families model is necessarily a preorder, due to the following result:

Theorem 5.5. *Assuming the law of excluded middle, if a category $\mathcal{V}$ has products indexed by the collection $Arr(\mathcal{V})$ of arrows in $\mathcal{V}$, then $\mathcal{V}$ is a preorder.*

Proof. Assume there are two distinct arrows $f, g : A \to B$ between objects $A, B \in \mathcal{V}$. We can combine these arrows to get $2^{|Arr(\mathcal{V})|}$ distinct arrows from $a \to \prod_{|Arr(\mathcal{V})|} b$, by the universal property of the product. But yields a contradiction, as this is obviously more arrows than exist in $\mathcal{V}$. By the law of the excluded middle, this double negation implies that there are no distinct arrows in $\mathcal{V}$. □

If we want $\mathcal{V}$ to not be a preorder, it is clear that we need to be more careful in distinguishing between small and large types.

In order to deal with this distinction on the semantic side, we will need to introduce some new notions:

Definition 5.6. For an ordinal α, the set $\mathcal{V}_\alpha$, 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 **Set** consists of all of the sets such that they are elements of V_α, for some α. The collection of all such sets, V is not a set itself, but a proper class.

Definition 5.7. For any ordinal α, the define the full subcategory $\mathbf{Set}_\alpha$ of **Set**, whose objects are sets X such that $X \in V_\alpha$.

The idea is to select a large enough cardinal κ, such that $\mathbf{Set}_\kappa$ 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:

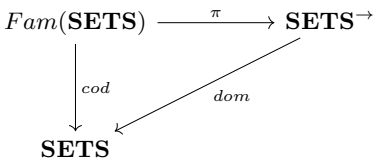
Definition 5.8. A cardinal, κ. is said to be:

- **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, and to interpret type theories with universes.

Definition 5.9. 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.

Now calling **SETS** what we previously called **Set**, its full subcategory $\mathbf{Sets}_\kappa$, consisting of all κ-small sets is closed under all the usual operations of set theory, and has all limits and colimits for all κ-small diagrams. We can now enlarge the families model to the following comprehension category:



and define the cartesian universe $\mathcal{U} \in Fam(\mathbf{SETS})_\Gamma$ by the constant family:

$$U = (\Gamma, \lambda\gamma.V_\kappa).$$

Any section $t : \Gamma \to \Gamma.U$ 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 5.2, paying attention to size, we get the following result:

Corollary 5.10. For any function $u : S \to S'$ such that for each $s \in S'$, the subset $u^{-1}(s)$ is small, the functor $u^* : Fam(\mathbf{SETS})_{S'} \to Fam(\mathbf{SETS})_S$ has left and right adjoints, Σ and Π. Furthermore, for each family (S', f) in $Fam(\mathbf{SETS})_S$, each member of the family $\Sigma(\Gamma, f) \in Fam(\mathbf{SETS})_S$ 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 $\pi_U : \Gamma.U \to \Gamma$ associated to universes are such that the sets $\pi_U(\gamma)$ are small, for sections $t : \Gamma \to \Gamma.U$ and $s : \Gamma.El(t) \to \Gamma.El(t).U$, 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.

Similarly, as long as $Ob(\mathcal{V}) \in \mathbf{SETS}$, we can define the linear universe as the family:

$$L = (\Gamma, \lambda\gamma.Ob(\mathcal{V}))$$

Where a section $t : \Gamma \to \Gamma.L$ defines a family $El(t) = (\Gamma, t_\gamma)$ where $t_\gamma \in \mathcal{V}$.

By this construction, we recover the "small" families model as sub-model of small types. Furthermore, the previous adjunction $L \dashv M$, induces, for every Γ, morphisms $L : L \iff U : M$ in $\mathcal{T}_\Gamma$, natural in Γ such that this induces an adjunction of $El(L(-)) \dashv El(M(-))$, which forms support for M and L for small types.

5.1.2 Concrete examples

To summarize, these are the conditions imposed in order to support all of the type formers described in section 3 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 **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-**Mod** of modules over R with the free functor/forgetful functor adjunction
- The category $\mathbf{CGTop}_*$, 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 $\mathbf{CGTop}_*$ is the two point discrete set S^0, which is precisely the image of the point in the adjunction above.

5.1.3 Syntactic enriched categories

As an in depth example of a construction in the families model, recall the notion of an enriched category in as defined in 2.12. Our goal is to form a syntactic construction whose interpretation of this is an enriched category. There are two ways of doing this. The first we will call a "meta construction", in which we ask for the existence of certain types and judgmental 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 $\mathcal{V}$:

$$\frac{\Gamma \vdash A \text{ linear} \quad \Gamma \vdash B \text{ linear}}{\Gamma \vdash [A, B] \text{ type}} \text{ hom-F} \qquad \frac{\Gamma \vdash f : [A, B] \quad \Gamma; \Xi \vdash a : A}{\Gamma; \Xi \vdash f(a) : B} \text{ hom-E}$$

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

In the presence of linear function types ($\multimap$) and the M-functor, this type is equivalent to $(A \multimap B)_M$. 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 5.11. 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 5.12 (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
- judgmental 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 $B : [[A]] \times [[A]] \to \mathcal{V}$, which for each pair $x, y \in A$ assigns an object of $\mathcal{V}$.

Definition 5.13 (Internal enriched categories)**.** The type **Enr-cat** is defined as

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

Since $Fam(\mathbf{Set})$ 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

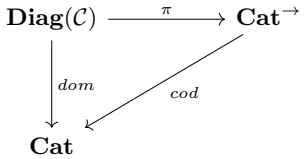
$$[[h \circ (g \circ f)]] \equiv [[(h \circ g) \circ f]]$$

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.

5.2 Diagrams in monoidal categories

Expanding on the set indexed families example, for any category $\mathcal{C}$, there is a fibration $cod : \mathbf{Diag}(\mathcal{C}) \to \mathbf{Cat}$, with total category $\mathbf{Diag}(\mathcal{C})$ begin the category of diagrams in $\mathcal{C}$ whose objects are functors $J : \mathcal{D} \to \mathcal{C}$, and whose morphisms between $J : \mathcal{D} \to \mathcal{V}$ and $J' : \mathcal{C} \to \mathcal{V}$ consist of functors $F : \mathcal{C} \to \mathcal{D}$ equipped with a natural transformation $J \circ F \implies J'$. 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 $F : \mathcal{A} \to \mathcal{B}$ in the base induces a canonical lift $F^* : [\mathcal{B}, \mathcal{C}] \to [\mathcal{A}, \mathcal{C}]$ 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,



where the functor π is defined as follows. Given any diagram $A : \Gamma \to \mathcal{C}$, we let the category $\Gamma.A$ be the **Grothendieck construction** for A, in other words, the category whose objects are pairs (γ, t_γ) where $\gamma \in \Gamma$ and $t_\gamma : \top \to A(\gamma)$. Morphisms $(\gamma, t_\gamma) \to (\gamma', t'_{\gamma'})$ consists of morphisms $u : \gamma \to \gamma'$ such that $A(u) \circ t_\gamma = t'_{\gamma'}$. If $\mathcal{C}$ is a 2-category, this can be weakened so that morphisms $(\gamma, t_\gamma) \to (\gamma', t'_{\gamma'})$ are pairs (u, α), where $u : \gamma \to \gamma'$ and α is a 2-cell $\alpha : A(u) \circ t_\gamma \implies t'_{\gamma'}$. The comprehension functor $\pi : \mathbf{Diag}(\mathcal{C}) \to \mathbf{Cat}^\to$ sends the diagram $A : \Gamma \to \mathcal{C}$ to obvious forgetful functor:

$$\pi_A : \Gamma.A \to \Gamma$$
$$\pi(\gamma, a_\gamma) = \gamma$$

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:

$$J \otimes J'(x) = J(x) \otimes J'(x) : [\Gamma, \mathcal{V}]$$
$$J \otimes J'(f) = J(f) \otimes J'(f) : J(x) \otimes J'(x) \to J(y) \otimes J'(y)$$

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

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

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 [HS98]:

$$\begin{array}{ccccc} \mathbf{Diag}(\mathcal{V}) & & \mathbf{Diag}\,(\mathbf{Gpd}) & \xrightarrow{\pi} & \mathbf{Gpd}^{\to} \\ & \searrow^{dom} & \downarrow{\scriptstyle dom} & \swarrow_{cod} & \\ & & \mathbf{Gpd} & & \end{array}$$

Since $\top$ is the groupoid $\mathbf{1}$ consisting of a single object, we will equate the functor $t_\gamma : 1 \to A(\gamma)$ with an object a_γ of $A(\gamma)$, and the natural transformation $\alpha : A(u) \circ t_\gamma \implies t'_{\gamma'}$ with a morphism $\alpha_\gamma : A(u)(a_\gamma) \to a'_{\gamma'}$.

As shown in [HS98], 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 $refl(x)$, which sets the stage for the homotopy interpretation of dependent type theory. Here, the interpretation of identity type Id_A is the functor $Id_A : \Gamma.A.A^+ \to \mathbf{Gpd}$ which sends an object $(\gamma, a_\gamma, b_\gamma)$ to the discrete groupoid $\delta \operatorname{Hom}_{A(\gamma)}(\gamma)(a, b)$, and a morphism $(u, \alpha, \beta) : (\gamma, a_\gamma, b_\gamma) \to (\gamma', a'_{\gamma'}, b'_{\gamma'})$ to the functor $\alpha^{-1} \circ A(u)(-) \circ \beta : \delta(\operatorname{Hom}_A(a, b)) \to \delta(\operatorname{Hom}_A(a', b'))$, which sends $f : a \to b$ to the composite $\alpha^{-1} \circ A(u)(f) \circ \beta : a' \to A(u)a \to A(u)b \to b'$.

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

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

to squares:

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

The following special case will become important:

Lemma 5.14. If a functor $A : \Gamma \to \mathbf{Gpd}$ takes values in discrete groupoids only, then for any two sections $M : \Gamma \to \Gamma.A$ and $N : \Gamma \to \Gamma.A$, if there exists a section $P : \Gamma \to \Gamma.Id_A\{M\}\{N^+\}$ then $M = N$.

Proof. The groupoid $\Gamma.Id_A\{M\}\{N^+\}$ has as objects

$$\big(\gamma, M_\gamma \in A(\gamma), N_\gamma \in A(\gamma)), f : M_\gamma \to N_\gamma\big)$$

where $M_\gamma \in A(\gamma)$ is determined by the section $M : \Gamma \to \Gamma.A$. If a section $P : \Gamma \to \Gamma.Id_A\{M\}\{N^+\}$ exists, it must for each γ, pick out a morphism $f : M_\gamma \to N_\gamma$. But since $A(\gamma)$ is a discrete groupoid, f can only be the identity morphism, and we must have $M_\gamma = N_\gamma$, for all $\gamma \in \Gamma$. Since every $A(\gamma)$ is discrete, this implies that $M = N$. □

Theorem 5.15. *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 4.15 is proved in [HS98]. 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 $M, N : \Gamma \to \Gamma.A$ and $P : \Gamma \to \Gamma.Id_A\{M\}\{N^+\}$. We are given a natural transformation $c : \Xi\{r_A\} \to C\{r_A\}$, between between the two functors $\Xi \circ r_A : \Gamma.A \to \Gamma.A.A^+.Id_A \to \mathcal{V}$ and $C \circ r_A : \Gamma.A \to \Gamma.A.A^+.Id_A \to \mathcal{V}$ and need to display a natural transformation $\hat{c}_{[M,N,P]}$ between:

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

to

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

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

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

in the groupoid $\Gamma.A^{\to}$, 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:

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

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

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 $- \otimes F : [\mathcal{C}, \mathcal{V}] \to [\mathcal{C}, \mathcal{V}]$. In other words for all functors G, H, we need

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

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

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

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

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

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:

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

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

$$[F, G](x) = Eq(\Pi_{x \in \mathcal{C}}[Fx, Gx] \underset{Ff_*}{\overset{Gf^*}{\rightrightarrows}} \Pi_{f:x \to y}[Fx, Gy])$$

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

Theorem 5.16. *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

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

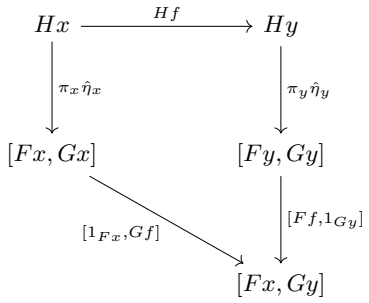
natural in G and H. Since $[F, G]$ is a constant functor, a map on the right hand side consists of, for every map $f : x \to y \in \mathrm{Mor}(\mathcal{C})$, of commutative diagrams of the following form:

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

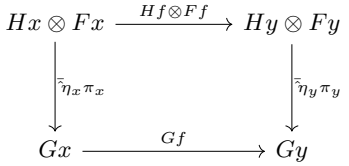
which will determine two unique morphisms, $\hat{\eta}_x : Hx \to \Pi_{x \in \mathcal{C}}[Fx, Gx]$ and $\hat{\eta}_y : Hy \to \Pi_{x \in \mathcal{C}}[Fx, Gx]$, such that the following diagram commutes:

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

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



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



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

Given any functor $p : \mathcal{D} \to \mathcal{C}$ 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 5.17. For any functor $p : \mathcal{A} \to \mathcal{B}$ in the base, a left or right adjoint to the induced functor $p^* : [\mathcal{B}, \mathcal{V}] \to [\mathcal{A}, \mathcal{V}]$ is called a **left or right Kan extension** of p.

Here is a general fact about Kan extensions:

Theorem 5.18. *Left (right) Kan extensions along $p : \mathcal{A} \to \mathcal{B}$ between two arbitrary small categories $\mathcal{A}$ and $\mathcal{B}$ exists if and only if $\mathcal{V}$ has all colimits (limits).*

Proof. Assume all left Kan extensions exist, and consider the diagram $J : \mathcal{A} \to \mathcal{V}$. Denote the left Kan extension along the functor $! : \mathcal{A} \to \mathbf{1}$ to the terminal category $Lan_! : [\mathcal{A}, \mathcal{V}] \to [\mathbf{1}, \mathcal{V}]$. Since functors in the image of $!^* : [\mathbf{1}, \mathcal{V}] \to [\mathcal{A}, \mathcal{V}]$ are functors constant at the object $v \in \mathcal{V}$, a natural transformation between functors $J \in [\mathcal{A}, \mathcal{V}]$ and $!^* v$ are equivalently cocones over J with v as a vertex. Since we have:

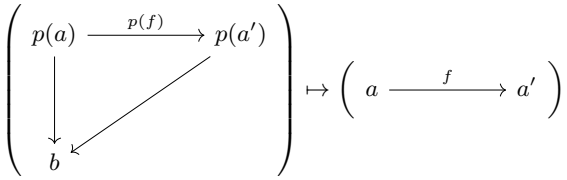
$$\hom_{[\mathbf{1}, \mathcal{V}]}(Lan_!(J), v) \cong \hom_{[\mathcal{A}, \mathcal{V}]}(J, !^* v)$$

any cocone $\eta : J \to !^* v$ gives rise to a unique morphism from the object selected by $Lan_!(J)$ to v, which is precisely the universal property of the colimit.

In the other direction, assume $\mathcal{V}$ has all colimits. Then we will, for any $p : \mathcal{A} \to \mathcal{B}$ construct a "pointwise" left Kan extension of any $F \in \mathcal{A} \to \mathcal{V}$. We define $Lan_p(F)$ to be the functor which sends $b \in \mathcal{B}$ to the colimit of the diagram

$$(p \downarrow b) \to \mathcal{A} \xrightarrow{F} \mathcal{V}$$

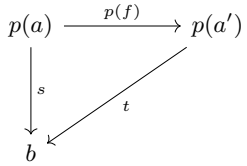
where $(p \downarrow b) \to \mathcal{A}$ is the forgetful functor illustrated in the following picture:



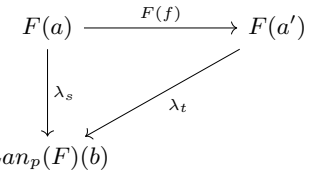
for any morphism $g : b \to b'$, there is an induces functor $(p \downarrow b) \to (p \downarrow b')$ by postcomposition, which induces a morphism of colimits:

$$\lim_{\to} \Big((p \downarrow b) \to \mathcal{A} \xrightarrow{F} \mathcal{V} \Big) \to \lim_{\to} \Big((p \downarrow b') \to \mathcal{A} \xrightarrow{F} \mathcal{V} \Big)$$

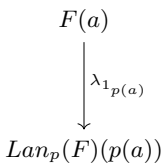
which we define to be the action of $Lan_p(F)$ on morphisms. We show that this construction gives rise to an adjunction by constructing the unit $\eta_F : F \to p^* Lan_p(F)$. For any $b \in \mathcal{B}$, denote the projections associated with any colimit of $Lan_p(F)b$ by $\lambda_{(-)}$. That is, for:



have:



Now, given any $a \in \mathcal{A}$, we have $p^* Lan_p(F)(a) = Lan_p(F)(p(a))$, the canonical projections in the case where the components of the slice $(p \downarrow p(a))$ are given by the identity:



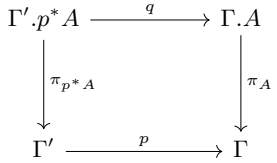
give rise to a natural transformation $\eta_F : F \to p^* Lan_p(F)$.

Leaving naturality conditions aside, it remains to show that this natural transformation satisfies the universal property of the unit. To that end, let $G \in [\mathcal{B}, \mathcal{V}]$ be a functor and $\phi : F \to p^* G$ a natural transformation. We want to display a unique natural transformation $\hat{\phi} : Lan_p(F) \to G$ such that $p^*(\hat{\phi}) \circ \eta_F = \phi$. But a natural transformation $\phi : F \to p^* G$ makes $G(p(a))$ a cocone to $(p \downarrow p(a)) \to \mathcal{A} \xrightarrow{F} \mathcal{V}$ for all $a \in \mathcal{A}$, so the universal property of the unit follows from the universal property of the colimit.

The cases for right Kan extensions and limits holds by a dual version of this argument. □

The result above ensures the existence of left and right adjoints to reindexing functors in the diagrams model as long as $\mathcal{V}$ is co-complete or complete, respectively. But since our main concern are diagrams over groupoids, it suffices to consider the case where $\mathcal{V}$ has limits and colimits for these. Similarly to the families model, the fact that our reindexing functors are given by precomposition ensures that these always satisfy the Beck-Chevalley condition:

Theorem 5.19. *If every lift $p^* : [\mathcal{B}, \mathcal{V}] \to [\mathcal{A}, \mathcal{V}]$ has a right adjoint $Ran_p : [\mathcal{A}, \mathcal{V}] \to [\mathcal{B}, \mathcal{V}]$, then, for all functors $A \in [\Gamma, \mathcal{V}]$ and pullbacks of the following form:*



there is a natural isomorphism $p^* Ran_{\pi_A} \cong Ran_{\pi_{p^*\{A\}}} q^* : [\Gamma.A, \mathcal{V}] \to [\Gamma', \mathcal{V}]$.

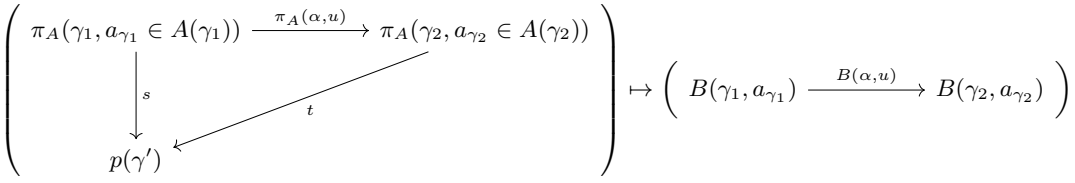
Proof. Let $B \in [\Gamma.A, \mathcal{V}]$ be a functor. Then $p^* Ran_{\pi_A}(B)$ is the functor sending $\gamma' \in \Gamma$ to the limit of:

$$(\pi_A \downarrow p(\gamma')) \to \Gamma.A \xrightarrow{B} \mathcal{V}$$

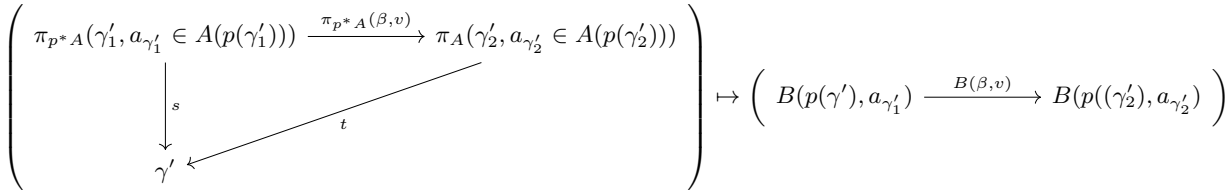
while $Ran_{\pi_{p^*A}} q^*(B)(\gamma')$ is the limit to:

$$(\pi_{p^*A} \downarrow \gamma') \to \Gamma'.p^*A \xrightarrow{q^*(B)} \mathcal{V}$$

The former functor is illustrated in the following picture:

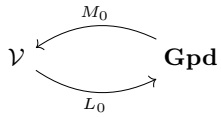


The latter is given by:



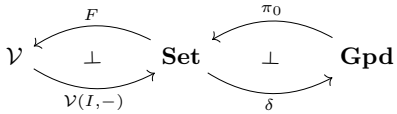
In both cases, we are looking at the limit of the image of B of all elements $(\gamma, a \in A(\gamma))$ such that $p(\gamma) \cong \gamma'$. By the universal property of the limit, these are therefore canonically isomorphic. □

In order to interpret M and L in this model, the same argument as in lemma 5.4 can be used to show that an adjunction between $\mathcal{V}$ and **Gpd** induces a fiber adjunction between the respective diagram categories. Therefore, for any diagrams model which supports $\multimap$ to support M and L, it suffices to display an adjunction



such that $L(1) \cong I$.

Remark 5.20. When $\mathcal{V}$ is a representable concrete category, we will often find that support for M and L comes "for free". Since the composition of two adjunctions is again an adjunction, we get that whenever the functor $\mathcal{V}(I,-) : \mathcal{V} \to \mathbf{Set}$ has a left adjoint F, then there exists an adjunction between $\mathbf{Diag}(\mathcal{V})$ and $\mathbf{Diag}(\mathbf{Gpd})$, arising out of:



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

Theorem 5.21. *There are models in which M is not faithful.*

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

5.2.1 Universes in the diagrams model

To support universes, assuming one inaccessible cardinal allows us to shift our perspective to from the category of small groupoids, **Gpd**, to the category **GPD** of all groupoids. Among the objects of **GPD** we find the large groupoid $\mathbf{Gpd}^{core}$, and $\mathcal{V}^{core}$, the core (i.e. maximal sub-groupoid) of the corresponding categories. This allows us to define our cartesian and linear universes in any context Γ as the functors:

$$U : \Gamma \to \mathbf{GPD}$$
$$L : \Gamma \to \mathbf{GPD}$$

which are constant at $\mathbf{Gpd}^{core}$ and $\mathcal{V}^{core}$, respectively. Any section $s : \Gamma \to \Gamma.U$ will determine a functor $\hat{s} : \Gamma \to \mathbf{Gpd}$, which will we can embed via the full subcategory embedding $\mathbf{Gpd} \to \mathbf{GPD}$ to get an interpretation of $El(s)$. Similarly, we get from each section $s : \Gamma \to \Gamma.L$, a functor $El(s) : \Gamma \to \mathcal{V}$. Restricting the type formers $\sqcap$, $\sqsubset$, M and L to small types only, this linear universe is all-encompassing.

It is easily seen that defining the linear universe via the core of $\mathcal{V}$ gives rise to the following interesting property, hinting at the possibility of a linear univalence axiom:

Corollary 5.22. For a linear universe defined as above via $\mathcal{V}^{core}$, and two sections $s, t : \Gamma \to \Gamma.L$, an isomorphism $\alpha : El(t) \cong El(s)$ gives rise to a section $p : \Gamma \to \Gamma.Id_L\{s\}\{t\}$.

5.3 Univalence in 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 setting.

In particular, we want to explore a model in which the following, linear analogue to the univalence axiom, would hold:

$$\frac{\begin{array}{c}\Gamma \vdash A : L \\ \Gamma \vdash B : L \\ \Gamma; \cdot \vdash f : El(A) \multimap El(B) \\ \Gamma; \cdot \vdash g : El(B) \multimap El(A) \\ \Gamma; \cdot \vdash h : El(B) \multimap El(A) \\ \Gamma \vdash p : (g \circ f)_M =_{(El(A) \multimap El(A))_M} (id_A)_M \\ \Gamma \vdash q : (f \circ h)_M =_{(El(B) \multimap El(B))_M} (id_B)_M\end{array}}{\Gamma \vdash ua(f)_{[g,h,p,q]} : A =_L B}\ \text{L-ua-I}$$

To define the corresponding computation rule, we first define a transport function for linear types:

Lemma 5.23. Suppose the following judgments hold:

$$\begin{array}{c}\Gamma, x : C \vdash D \ \text{ linear} \\ \Gamma \vdash a : C \\ \Gamma \vdash b : C \\ \Gamma \vdash p : a =_C b\end{array}$$

Then there is a a linear function $\Gamma; \cdot \vdash p^* : D[a/x] \multimap D[b/x]$, which we call the **transport along p**

Proof.

$$\frac{\begin{array}{c}\Gamma, x, y : C \vdash D[x/x] \multimap D[y/x] \text{ linear} \\ \Gamma, z : C; \cdot \vdash \lambda u.u : D[z/x] \multimap D[z/x] \\ \Gamma \vdash a : C \\ \Gamma \vdash b : C \\ \Gamma \vdash p : a =_C b\end{array}}{\Gamma; \cdot \vdash R^{Id}_{[x,y,p]}(\lambda u.u, a, b, p) : D[a/x] \multimap D[b/x]}$$

□

Taking $C \equiv L$, and $D \equiv El(x)$ for any $x : L$ in the procedure above yields, from judgments:

$$\begin{array}{c}\Gamma \vdash A : L \\ \Gamma \vdash B : L \\ \Gamma \vdash p : A =_L B\end{array}$$

the function $\Gamma; \cdot \vdash R^{Id}_{[x,y,p]}(\lambda u.u, A, B, p) : El(C) \multimap El(D)$. It can easily be shown that this function gives rise to a linear equivalence between $El(C)$ and $El(D)$. The two computation rules for the linear univalence axiom states that the procedure of generating an equivalence from an identity is itself an equivalence. In other words, given a linear function $f : El(A) \multimap El(B)$ which gives rise to a linear equivalence of $El(A)$ and $El(B)$, turning the equivalence f into an identity and then back into an equivalence should return f again:

$$\frac{\Gamma \vdash R^{Id}_{[x,y,p]}(\lambda u.u, A, B, ua(f)_{g,h,p,q}) : El(A) \multimap El(B)}{\Gamma \vdash R^{Id}_{[x,y,p]}(\lambda u.u, A, B, ua(f)_{g,h,p,q}) \equiv f : El(A) \multimap El(B)} \text{ L-ua-C}_1$$

and in the other direction, given an identity $p : A =_L B$, turning it to an equivalence and then back into an identity should return p:

$$\frac{\Gamma \vdash ua(R^{Id}_{[x,y,p]}(\lambda u.u, A, B, p))_{[-,-,-,-]} : A =_L B}{\Gamma \vdash ua(R^{Id}_{[x,y,p]}(\lambda u.u, A, B, p))_{[-,-,-,-]} \equiv p : A =_L B} \text{ L-ua-C}_2$$

5.3.1 Semantic justification

The semantic interpretation of the procedure of turning an identity to an equivalence above is the following:

Lemma 5.24. Given two sections $A, B : \Gamma \to \Gamma.L$, and a section $p : \Gamma \to \Gamma.Id_L\{A\}\{B^+\}$, there is an natural isomorphism $IdToEquiv(p) : El(A) \cong El(B)$ in $[\Gamma, \mathcal{V}]$.

Proof. The section $p : \Gamma \to \Gamma.Id_L\{A\}\{B^+\}$ must for every Γ, select an object $(a \in A(\gamma), b \in B(\gamma), f : a \to b)$ of the groupoid $Id_L.\{A\}\{B^+\}(\gamma) = \delta\mathcal{V}^{core}(A, B)$, and map morphisms to commutative squares, defining a natural isomorphism $El(A) \cong El(B)$ between the diagrams $El(A) : \Gamma \to \mathcal{V}$ and $El(B) : \Gamma \to \mathcal{V}$. □

Theorem 5.25. *For $L \dashv M$ interpreted as in 5.20, the linear univalence axiom holds in the diagrams model. That is, given the following data:*

- *sections: $A, B : \Gamma \to \Gamma.L$*
- *morphisms: $f : I \to [El(A), El(B)]$ and $g, h : I \to El(B)$ in $[\Gamma, \mathcal{V}]$,*
- *and sections: $p : \Gamma \to \Gamma.Id_{(M[El(A),El(A)])}\{M(g \circ f)\}\{M(id_A)\}$ and $q : \Gamma \to \Gamma.Id_{(M[El(B),El(B)])}\{M(f \circ h)\}\{M(id_B)\}$*

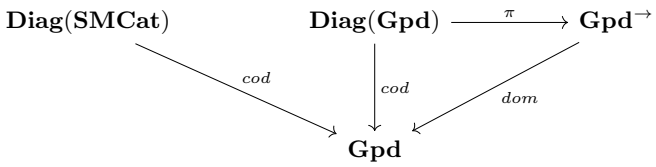
Then there is a section $ua(f)_{g,h,p,q} : \Gamma \to \Gamma.Id_L\{A\}\{B\}$ such that $IdToEquiv(ua(f)) = f$ and $ua(IdToEquiv(p)) = p$.

Proof. Since M factors through sets, the interpretation of any type in its image is a functor with values in discrete groupoids, so by lemma 5.14, the existence of a section $p : \Gamma \to \Gamma.Id_{M[El(A),El(A)]}\{(g \circ f)_M\}\{(id_{El(A)})_M\}$ implies that the sections $(g \circ f)_M : 1 \to M[A, A]$ and $(id_{El(A)})_M : 1 \to M[A, A]$ coincide, which by the adjunction $L \dashv M$ implies that $[[(g \circ f)]] = id_{El(A)}$. From the interpretation of $\multimap$ as internal hom, the syntactic composition operation coincides with the composition of the corresponding morphisms. In other words, denoting by $\hat{f}$ the transport of a map via the isomorphism $\mathcal{L}_\Gamma(I, [A, B]) \cong \mathcal{L}_\Gamma(A, B)$, we have $\hat{[[g \circ f]]} = \hat{g} \circ \hat{f}$ and $\hat{id}_{El(A)} = 1_{El(A)}$. We therefore have an isomorphism $\hat{f} : El(A) \to El(B)$ with $\hat{g} = \hat{h}$ as inverse. Selecting this isomorphism in $\mathcal{V}^{core}$ gives rise to a section $P : \Gamma \to \Gamma.Id_L\{A\}\{B\}$, which by the previous lemma can be transported back by $IdToEquiv(P) = \hat{f}$. □

So the linear univalence axiom holds in the diagrams model as long as M factors through sets, analogous to how one can prove the univalence axiom in the groupoid model for the universe which only contains discrete groupoids. This might not be completely satisfying from the perspective of homotopy type theory, as this essentially truncates any higher dimensional type into a set when transporting them via M. Instead, we could to imagine forming a model where our linear types have an inherently higher dimensional structure, which is preserved by M. One such model is given by letting the objects of $\mathcal{V}$ be categories, and M be the functor taking a category to its core, i.e. its maximal sub-groupoid. If we want $\mathcal{V}$ to have a non-cartesian monoidal structure the candidate choices **Cat** and **Gpd** won't do. As briefly outlined in in 2.15

and carefully described in [Sch07], there is a symmetric monoidal structure on **SMCat**, the category of small symmetric monoidal categories, symmetric monoidal functors and monoidal natural transformations. [6]

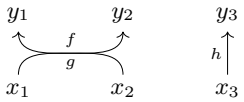
Definition 5.26. Let the **2-categorical model of LDTT** be given by the diagrams model where $\mathcal{V}$ is the 2-category of small symmetric monoidal categories, symmetric monoidal functors and monoidal natural transformations:



For two symmetric monoidal categories $\mathcal{A}$ and $\mathcal{B}$, the category $[\mathcal{A}, \mathcal{B}]$ consisting of monoidal functors and monoidal transformations between them carries a natural monoidal structure [Sch07], and serves as the internal hom of $\mathcal{A}$ and $\mathcal{B}$. Since **SMCat** is complete, with limits inherited from **Cat** equipped with a pointwise monoidal structure, we have support for $\sqcap$ and &, and theorem 5.16 gives us that this model supports $\multimap$ type formers. [7]

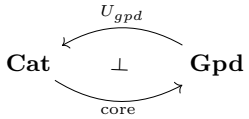
Lemma 5.27. There is a functor $F : \mathbf{Cat} \to \mathbf{SMCat}$ which constructs the free symmetric monoidal category of any groupoid, i.e. is a left adjoint to the forgetful functor $U_{smcat} : \mathbf{SMCat} \to \mathbf{Cat}$, forgetting the monoidal structure.

Proof. Construct the free symmetric monoidal category $F\mathcal{C}$ of a small category $\mathcal{C}$ by letting the objects of $F\mathcal{C}$ 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 $\mathcal{C}$:

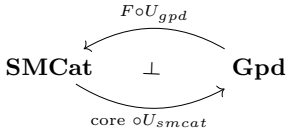


The composition of such morphisms are just the composition of the corresponding string diagrams, composing the labels of strings along the way. The monoidal structure is simply given by concatenation of words, where the empty word corresponds to the unit. From this construction, it should be clear that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

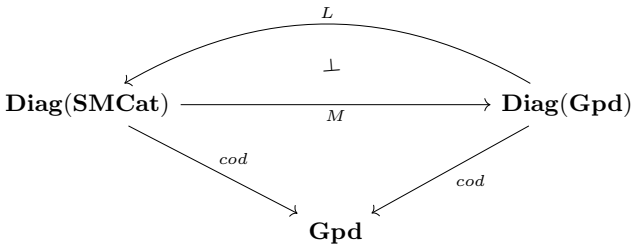
Combining this adjunction with the familiar adjunction:



where U_{gpd} is the forgetful functor and core is the functor sending a category to its underlying maximal sub-groupoid, we get an adjunction:



This adjunction lifts by lemma 5.4 to a fiber adjunction:



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

Here we have two choices for our linear universe.[8] If we want univalence to hold in this model we can let the linear universe L be the groupoid consisting of all *discrete* symmetric monoidal categories, and isomorphisms between them. This

[6] Technically, the structure on **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.

[7] Note, however, that we do not have all coproducts in **SMCat**. Therefore, we cannot support $\oplus$ or $\sqcap$. An alternative to be explored is the category **Mult**, of multicategories, which is a symmetric monoidal closed, complete and co-complete [EM09]. More on this in section 6

[8] In our syntax we only defined a single linear universe, but it can easily be extended into several universes, which may be subcategories of each other, or even an infinite hierarchy of universes

is equivalently the core of $CMon$, the category of small commutative monoids (in **Set**). $CMon$ is a full subcategory of $SMCat$, which allows us to let El be the function sending an object of L to an object of **SMCat**.

Another choice is to let $L = \mathbf{SMCat}^{core}$, making it an all-encompassing universe. This universe is not univalent, however, as one-dimensional groupoids do not carry enough higher dimensional structure to capture the weakness distinguishing an equivalence of categories and an isomorphism of categories.

5.3.2 Univalent linear type theory (syntactic)

It is not uncommon for linear logicians to state equalities like $A \multimap B \multimap C = A \otimes B \multimap C$ to mean $A \multimap B \multimap C \vdash A \otimes B \multimap C$ and $A \otimes B \multimap C \vdash A \multimap B \multimap C$. Here it is important to distinguish between linear logic and linear type theory. In linear logic, we have $A \& A \dashv\vdash A$, but in linear type theory, there is in general no equivalence $A \& A \cong A$, since there may be more than one term inhabiting A.

Theorem 5.28. *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. $\Gamma \vdash (\Sigma_{x:D} E)_L =_{\mathcal{L}} \sqcap_{x:D} E_L$
3. $(A \& B)_{LM} =_{\mathcal{L}} A_{LM} \otimes B_{LM}$

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

$$\multimap\text{-E}\,\frac{\Gamma; f : A \multimap B \multimap C \vdash f : A \multimap B \multimap C \qquad \Gamma; y : A \vdash y : A}{\multimap\text{-E}\,\dfrac{\Gamma; f : A \multimap B \multimap C, y : A, z : B \vdash f(y) : B \multimap C \qquad \Gamma; z : B \vdash z : B}{\dfrac{\Gamma; f : A \multimap B \multimap C, y : A, z : B \vdash f(y)(z) : C \qquad \Gamma; x : A \otimes B \vdash x : A \otimes B}{\dfrac{\Gamma; f : A \multimap B \multimap C, x : A \otimes B \vdash \text{let } x \text{ be } (y,z) \text{ in } f(y)(z) : \mathrm{C}}{\dfrac{\Gamma; f : A \multimap B \multimap C \vdash \lambda x.\text{let } x \text{ be } (y,z) \text{ in } f(y)(z) : A \otimes B \multimap C}{\Gamma; \cdot \vdash \lambda f.\lambda x.\text{let } x \text{ be } (a,b) \text{ in } f(a)(b) : (A \multimap B \multimap C) \multimap (A \otimes B \multimap C)}\multimap\text{-I}}\multimap\text{-I}}\otimes\text{-E}}}$$

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

$$\frac{\dfrac{\Gamma; \beta : B \vdash \beta : B \qquad \Gamma; \alpha : A \vdash \alpha : A}{\Gamma; \alpha : A, \beta : B \vdash \alpha \otimes \beta : A \otimes B} \qquad \Gamma; g : A \otimes B \multimap C \vdash g : A \otimes B \multimap C}{\dfrac{\dfrac{\Gamma; g : A \otimes B \multimap C, \alpha : A, \beta : B \vdash g(\alpha \otimes \beta) : C}{\Gamma; g : A \otimes B \multimap C, \alpha : A \vdash g(\alpha \otimes \beta) : B \multimap C}\multimap\text{-I}}{\Gamma; g : A \otimes B \multimap C \vdash \lambda\alpha.\lambda\beta.g(\alpha \otimes \beta) : A \multimap B \multimap C}\multimap\text{-I}}{\Gamma; \cdot \vdash \lambda g.\lambda\alpha.\lambda\beta.g(\alpha \otimes \beta) : (A \otimes B \multimap C) \multimap (A \multimap B \multimap C)}\multimap\text{-I}$$

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

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

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

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

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 3.8, there is a linear equivalence $(\Sigma_{x:D} E)_L \cong \sqcap_{x:D} E_L$. 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 $f^{-1} : A_{LM} \otimes B_{LM} \multimap (A\&B)_{LM}$:

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

which are easily seen to be mutually inverse.

□

6 Discussion

6.1 Equality of linear types and linear function extensionality

A basic assumption of our theory is the restriction disallowing linear terms to appear in types. This implies that we cannot form the identity type of two terms of a linear type, which could be seen as a lack of proper "internalization" of the notion of equality between linear terms. However, we can still utilize the "surrogate equality" described at the end of Section 3.4, which compares the image of the two linear terms under M. Building upon this idea, it is tempting to consider the addition of some kind of "extensionality"-style rules to make the "surrogate equality" useful in practice. Such rules should allow us to prove equalities of linear terms by proving equalities of their consituent parts. For example, suppose we are given terms:

$$\Gamma; x : A \vdash t : B$$
$$\Gamma; x : A \vdash s : B$$

and we know s and t to be equal in some weak (or propositional) sense, and therefore, we would like to say that there is a term of the type:

$$\Gamma \vdash (\lambda x.t)_M =_{(A \multimap B)_M} (\lambda x.s)_M$$

We might say that to prove such an equality, it suffices to display a proof of the type:

$$\Gamma \vdash \text{fmap } (\lambda x.t) =_{A_M \to B_M} \text{ fmap}(\lambda x.s) \equiv$$
$$\Gamma \vdash \lambda y.\big(\lambda x.t(\sigma(y))\big)_M =_{A_M \to B_M} \lambda y.\big(\lambda x.s(\sigma(y))\big)_M$$

which, in the presence of (cartesian) function extensionality reduces to a proof of:

$$\Gamma \vdash \Pi_{y:A}\, (t[(\sigma(y)/x]_M =_{B_M} s[\sigma(y)/x]_M)$$

We consider this rule to be a kind of *linear function extensionality*, even though the name does not reflect the interplay between M and L present in its formulation. For an application of this rule, consider the proof of Theorem 3.8. There we displayed a linear equivalence by showing judgmental equality between the roundabout $g \circ f$ and the identity function, but it is tempting to weaken this to a propositional equality and drop additional assumption Σ-U.

We have not investigated the semantic interpretation of linear function extensionality, nor its interplay with univalence or linear univalence. It appears that more investigation is warranted regarding equality between linear terms.

6.2 Inductive types in linear dependent logic

As we saw in the case for Σ- and Id-types, we add new elimination and computation rules for our traditional cartesian type formers for when the type we eliminate into is linear. This should remain true for inductive types. For example, adding the type of natural numbers to our theory, we would add the rules [$\mathbb{N}$-E_2], [$\mathbb{N}$-C_2-0], [$\mathbb{N}$-C_2-S] to the typical rules for the natural numbers type:

$$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash \mathbb{N} \text{ type}}\ \mathbb{N}\text{-F}$$

$$\frac{\vdash \Gamma \text{ ctxt}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-0-I} \qquad \frac{\Gamma \vdash R^{\mathbb{N}}_n(H_z, H_s, Suc(M)) : C[Suc(M)/n]}{\Gamma \vdash R^{\mathbb{N}}_n(H_z, H_s, Suc(M)) \equiv H_s[M/n, R^{\mathbb{N}}_n(H_z, H_s, M)/x] : C[Suc(M)/n]}\ \mathbb{N}\text{-C}_1\text{-S}$$

$$\frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash Suc(M) : \mathbb{N}}\ \mathbb{N}\text{-S-I}$$

$$\frac{\begin{array}{c}\Gamma, n : \mathbb{N} \vdash C \text{ linear} \\ \Gamma; \Xi \vdash H_z : C[0/n] \\ \Gamma, n : \mathbb{N}; \Xi, x : C \vdash H_s : C[Suc(n)/x] \\ \Gamma \vdash M : \mathbb{N}\end{array}}{\Gamma; \Xi \vdash R^{\mathbb{N}}_n(H_z, H_s, M) : C[M/n]}\ \mathbb{N}\text{-E}_2$$

$$\frac{\begin{array}{c}\Gamma, n : \mathbb{N} \vdash C \text{ type} \\ \Gamma \vdash H_z : C[0/n] \\ \Gamma, n : \mathbb{N}, x : C \vdash H_s : C[Suc(n)/x] \\ \Gamma \vdash M : \mathbb{N}\end{array}}{\Gamma \vdash R^{\mathbb{N}}_n(H_z, H_s, M) : C[M/n]}\ \mathbb{N}\text{-E}_1 \qquad \frac{\Gamma; \Xi \vdash R^{\mathbb{N}}_n(H_z, H_s, 0) : C[0/n]}{\Gamma; \Xi \vdash R^{\mathbb{N}}_n(H_z, H_s, 0) \equiv H_z : C[0/n]}\ \mathbb{N}\text{-C}_2\text{-0}$$

$$\frac{\Gamma; \Xi \vdash R^{\mathbb{N}}_n(H_z, H_s, Suc(M)) : C[Suc(M)/n]}{\Gamma; \Xi \vdash R^{\mathbb{N}}_n(H_z, H_s, Suc(M)) \equiv H_s[M/n, R^{\mathbb{N}}_n(H_z, H_s, M)/x] : C[Suc(M)/n]}\ \mathbb{N}\text{-C}_2\text{-S}$$

$$\frac{\Gamma \vdash R^{\mathbb{N}}_n(H_z, H_s, 0) : C[0/n]}{\Gamma \vdash R^{\mathbb{N}}_n(H_z, H_s, 0) \equiv H_Z : C[0/n]}\ \mathbb{N}\text{-C}_1\text{-0}$$

These rules allow us to define an addition operator for the linear type $\mathbb{N}_L$:

We first define a successor operation for $\mathbb{N}_L$ by:

$$LSuc : \mathbb{N}_L \multimap \mathbb{N}_L$$
$$LSuc :\equiv \lambda x.\text{let } n \text{ be } x \text{ in } Suc(n)_L$$

and define the function that applies this operation $x : \mathbb{N}$ times to any term of $\mathbb{N}_L$:

$$\frac{\dfrac{x : \mathbb{N}, n : \mathbb{N}, z : \mathbb{N}; m : \mathbb{N}_L \vdash LSuc(m) : \mathbb{N}_L \qquad y : \mathbb{N}_L \vdash y : \mathbb{N}_L}{x : \mathbb{N}, n : \mathbb{N}; y : \mathbb{N}_L, m : \mathbb{N}_L \vdash \text{let } z \text{ be } y \text{ in } LSuc(m) : \mathbb{N}} \qquad \begin{array}{c} x : \mathbb{N}; y : \mathbb{N}_L \vdash y : \mathbb{N}_L \\ x : \mathbb{N} \vdash x : \mathbb{N} \end{array}}{x : \mathbb{N}; y : \mathbb{N}_L \vdash R^{\mathbb{N}}(y, \text{let } z \text{ be } y \text{ in } LSuc(m), x) : \mathbb{N}_L}$$

allowing us to define "linear addition operator":

$$\_L + \_ : \mathbb{N}_L \multimap \mathbb{N}_L \multimap \mathbb{N}_L$$
$$\_L + \_ :\equiv \lambda n.\lambda m.\text{let } x \text{ be } n \text{ in } R^{\mathbb{N}}(y, \text{let } z \text{ be } y \text{ in } LSuc(m), x)$$

It remains future work to investigate which conditions these extra rules impose on the semantic interpretation of the natural numbers type, and also to explore more generally how to understand inductive types in the linear dependent setting.

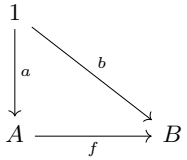
6.3 Linearity, pointedness, stability and the delooping hypothesis

In our models, we have kept the categories $\mathcal{L}$ and $\mathcal{T}$ of linear and cartesian types only related mainly via the adjunction $L \dashv M$. This has provided us with a variety of options for understanding linear and cartesian types and their relation, but at this level of generality there does not seem to be a strong conceptual (or ontological) connection between linear and cartesian types.

Inspired by the higher categorical interpretation of dependent type theory, we consider a few different ways of connecting cartesian and dependent types, both from a syntactic and semantic point of view.

6.3.1 Linear types as pointed cartesian types

For any category $\mathcal{C}$ with terminal object, we can consider the its pointed version, $\mathcal{C}_*$, consisting of objects $A \in \mathcal{C}$ together with a morphism $a : 1 \to A$, whose morphisms are commutative triangles:



This category has coproducts (called wedge sums) given by pushouts if $\mathcal{C}$ has these, products inherited from $\mathcal{C}$, and a symmetric tensor product (called the smash product) when $\mathcal{C}$ has both products and pushouts. Furthermore, when $\mathcal{C}$ has

all coproducts, the forgetful functor $\mathcal{C}^{*/} \to \mathcal{C}$ which forgets the base point has a left adjoint which is given for any object A by he coproduct $1 + A$.

This suggests that whenever we have a model for dependent type theory which supports Π and Σ, we should get a model for linear type theory by considering the category of pointed objects over any context. $\mathcal{L}_\Gamma = \mathcal{T}_\Gamma^{*/}$.

The connection gives motivation for including the following derivation rules:

$$\frac{\begin{array}{c}\vdash A \ \text{type} \\ \vdash a : A\end{array}}{\vdash (a, A) \ \text{linear}}$$

$$\frac{\begin{array}{c}\vdash a : A \\ \vdash b : B \\ \vdash f : A \to B \\ \vdash f(a) \equiv b\end{array}}{\cdot; (a, A) \vdash f : (b, B)}$$

However, under the homotopical interpretation of type theory, there might be a more interesting relationship between pointed cartesian types and linear types.

6.3.2 Linear types as looped cartesian types

From the homotopy type theory interpretation of dependent type theory, we are inclined to think of the category of cartesian types as an n-dimensional or ∞-dimensional groupoid, where the identity type provides us with the means of going "one step up the ladder" of higher dimensions. The

There are higher dimensional analogues to this process of relating pointed categories and monoidal categories, going under the homotopy theory inspired names of *looping* and *delooping*. The full theory and motivation behind this process falls outside the scope of this paper, and the curious reader is directed to [BS10]. Below we outline the main idea and discuss how this relates to linear dependent type theory.

Starting with a low dimensional example, we can for any small category $\mathcal{A}$ with a distinguished object a construct a monoid, $\Omega(A)$, the *looping of* A, whose underlying set of objects is given by $\mathrm{Ob}(\Omega(A)) = \mathcal{C}(a, a)$, and whose monoid structure is induced by the composition in $\mathcal{A}$. In the other direction, we can for any monoid M, construct a small category $\Sigma(M)$ consisting a single object $*$, whose set of morphisms is given by the underlying set of M, and whose composition is given by multiplication in M. This procedure extends to an adjoint pair $B \dashv \Omega$ of functors between the category of pointed categories and the category of monoids. In the case where the small category $\mathcal{A}$ is monoidal, the resulting monoid has two "monoid operations", which by an Eckmann–Hilton argument turn out to coincide and yield a commutative monoid.

Going one dimension higher, if $(\mathcal{C}, a)$ is a bicategory (recall definition 2.14) with a distinguished object $a \in \mathcal{C}_0$, we define its looping $\Omega(A)$ to be the category whose objects are the 1-morphisms in a, and whose morphisms are the 2-morphisms between these. In other words:

$$\Omega(A) = \mathcal{C}_{a,a}$$

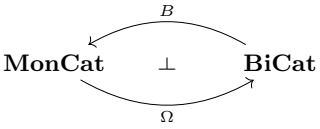
Now one can verify that the structure given from the composition $M_{a,a,a} : \mathcal{C}_{a,a} \times \mathcal{C}_{a,a} \to \mathcal{C}_{a,a}$ and identity functor $1_a : \mathbf{1} \to \mathcal{C}_{a,a}$ and the associated associators and unitors is precisely that of a monoidal category.

The dual of this procedure takes a monoidal category $\mathcal{V}$ and defines $\Sigma(\mathcal{V})$ as the bicategory consisting of a single object, $*$, to which we associate corresponding 1-morphisms and 2-morphisms by:

$$\Sigma(\mathcal{V})_{*,*} = \mathcal{V}$$

Again, the monoidal structure of $\mathcal{V}$ gives us precisely the structure needed to define the coherent composition and identity functors.

It can be shown that the process of looping and suspension can be extended to functors, which form a sort of adjunction between monoidal categories and bicategories.



By double looping and delooping, we get an adjunction between commutative monoids and pointed bicategories.

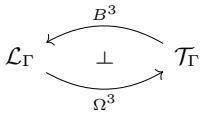
Motivated by these lower dimensional examples provides an intuition behind the *delooping hypothesis*: [9]

Hypothesis 6.1. There is an adjoint pair $B^k \dashv \Omega^k$ between k-monoidal n-categories and pointed $(n + k)$-categories.

What k-monoidal means in this context is the higher dimensional generalization of correspondence between sets, monoids, commutative monoids in the 0-dimensional case, and categories, monoidal-, braided monoidal- and symmetric monoidal categories in the 1-dimensional case.

[9] This version is a simplified form of the more general version given in [BS10]

In the context of dependent linear type theory, the hypothesis suggests that if we want the category $\mathcal{L}_\Gamma$ to be a non-discrete symmetric monoidal category, i.e. a 3-monoidal 1-category, the lowest dimension needed for a non-trivial category of cartesian types is to let $\mathcal{T}_\Gamma$ be a pointed tetracategory, i.e. weak 4-category, connected by the adjunction:



At this point, one might jump straight to the ∞-dimensional case[10], and consider all n-dimensional examples as special cases.

Roughly, this means that we are to understand a cartesian context Γ as an ∞-groupoid as in the simplicial sets model, from which we can construct the ∞-category $Sp(\Gamma)$ of *spectrum objects*, in the sense of [Lur06], if Γ admits finite limits. The category $Sp(\Gamma)$ should be a symmetric monoidal $(\infty, 1)$-category, which seems like a suitable environment to interpret "higher dimensional" linear types.

This suggests a connection between dependent linear type theory and stable homotopy theory akin to the connection between dependent type theory and to homotopy theory. This connection was hypothesized in [Sch14].

However, as the connection between linear types and cartesian types relies heavily on pointed objects in the cartesian realm, one is tempted to experiment with a reformulation of the adjunction $L \dashv M$ that reflects this.

References

[Awo10] Steve Awodey, *Category theory*, second ed., Oxford Logic Guides, vol. 52, Oxford University Press, Oxford, 2010.

[Ben95] P. N. Benton, *A mixed linear and non-linear logic: proofs, terms and models (extended abstract)*, Computer science logic (Kazimierz, 1994), Lecture Notes in Comput. Sci., vol. 933, Springer, Berlin, 1995, pp. 121–135, https://doi.org/10.1007/BFb0022251.

[BS10] John C. Baez and Michael Shulman, *Lectures on n-categories and cohomology*, Towards higher categories, IMA Vol. Math. Appl., vol. 152, Springer, New York, 2010, pp. 1–68, https://doi.org/10.1007/978-1-4419-1524-5_1.

[D+06] Ross Duncan et al., *Types for quantum computing*, Ph.D. thesis, University of Oxford, 2006.

[EM09] A. D. Elmendorf and M. A. Mandell, *Permutative categories, multicategories and algebraic K-theory*, Algebr. Geom. Topol. **9** (2009), no. 4, 2391–2441, https://doi.org/10.2140/agt.2009.9.2391.

[Hof95] Martin Hofmann, *On the interpretation of type theory in locally Cartesian closed categories*, Computer science logic (Kazimierz, 1994), Lecture Notes in Comput. Sci., vol. 933, Springer, Berlin, 1995, pp. 427–441, https://doi.org/10.1007/BFb0022273.

[Hof97] ———, *Syntax and semantics of dependent types*, Semantics and logics of computation (Cambridge, 1995), Publ. Newton Inst., vol. 14, Cambridge Univ. Press, Cambridge, 1997, pp. 79–130, https://doi.org/10.1017/CBO9780511526619.004.

[HS98] Martin Hofmann and Thomas Streicher, *The groupoid interpretation of type theory*, Twenty-five years of constructive type theory (Venice, 1995), Oxford Logic Guides, vol. 36, Oxford Univ. Press, New York, 1998, pp. 83–111.

[Hur95] Antonius J. C. Hurkens, *A simplification of Girard's paradox*, Typed lambda calculi and applications (Edinburgh, 1995), Lecture Notes in Comput. Sci., vol. 902, Springer, Berlin, 1995, pp. 266–278, https://doi.org/10.1007/BFb0014058.

[Jac93] Bart Jacobs, *Comprehension categories and the semantics of type dependency*, Theoret. Comput. Sci. **107** (1993), no. 2, 169–207, https://doi.org/10.1016/0304-3975(93)90169-T.

[Kel05] G. M. Kelly, *Basic concepts of enriched category theory*, Repr. Theory Appl. Categ. (2005), no. 10, vi+137, Reprint of the 1982 original [Cambridge Univ. Press, Cambridge; MR0651714].

[KPB15] Neelakantan R Krishnaswami, Pierre Pradic, and Nick Benton, *Integrating linear and dependent types*, ACM SIGPLAN Notices, vol. 50, ACM, January 2015, pp. 17–30, http://www.cs.bham.ac.uk/~krishnan/dlnl-paper.pdf.

[LSR17] Daniel R. Licata, Michael Shulman, and Mitchell Riley, *A Fibrational Framework for Substructural and Modal Logics*, 2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017) (Dagstuhl, Germany) (Dale Miller, ed.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 84, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 25:1–25:22, doi:10.4230/LIPIcs.FSCD.2017.25, http://drops.dagstuhl.de/opus/volltexte/2017/7740.

[Lur06] Jacob Lurie, *Stable infinity categories*, preprint, 2006, arXiv:math/0608228.

[LW15] Peter LeFanu Lumsdaine and Michael A. Warren, *The local universes model: an overlooked coherence construction for dependent type theories*, ACM Trans. Comput. Log. **16** (2015), no. 3, Art. 23, 31, https://doi.org/10.1145/2754931.

[10]The definition of a tetracategory spans multiple pages for every single coherence axiom (of which there are many)

[McB16] Conor McBride, *I got plenty o' nuttin'*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 207–233, Springer, 3 2016, doi:10.1007/978-3-319-30936-1_12.

[Mel09] Paul-André Melliès, *Categorical semantics of linear logic*, Interactive models of computation and program behavior, Panor. Synthèses, vol. 27, Soc. Math. France, Paris, 2009, pp. 1–196.

[Mer15] Lucius Gregory Meredith, *Linear types can change the blockchain*, preprint, 2015, arXiv:1506.01001.

[ML84] Per Martin-Löf, *Intuitionistic type theory*, Studies in Proof Theory. Lecture Notes, vol. 1, Bibliopolis, Naples, 1984, Notes by Giovanni Sambin.

[Pro13] The Univalent Foundations Program, *Homotopy type theory—univalent foundations of mathematics*, The Univalent Foundations Program, Princeton, NJ; Institute for Advanced Study (IAS), Princeton, NJ, 2013.

[Sch07] Vincent Schmitt, *Tensor product for symmetric monoidal categories*, preprint, 2007, arXiv:0711.0324.

[Sch14] Urs Schreiber, *Quantization via linear homotopy types*, preprint, 2014, arXiv:1402.7041.

[Shu08] Michael Shulman, *Framed bicategories and monoidal fibrations*, Theory Appl. Categ. **20** (2008), No. 18, 650–738.

[Str91] Thomas Streicher, *Semantics of type theory*, Progress in Theoretical Computer Science, Birkhäuser Boston, Inc., Boston, MA, 1991, Correctness, completeness and independence results, With a foreword by Martin Wirsing, https://doi.org/10.1007/978-1-4612-0433-6.

[VÍ5] Matthijs Vákár, *A categorical semantics for linear logical frameworks*, Foundations of software science and computation structures, Lecture Notes in Comput. Sci., vol. 9034, Springer, Heidelberg, 2015, pp. 102–116, https://doi.org/10.1007/978-3-662-46678-0_7.

[Zaw11] Marek Zawadowski, *Lax monoidal fibrations*, Models, logics, and higher-dimensional categories, CRM Proc. Lecture Notes, vol. 53, Amer. Math. Soc., Providence, RI, 2011, pp. 341–426.