Syntax and Semantics of Linear Dependent Types
Technical Report

Matthijs Vákár

Oxford, UK, January 19, 2015

Abstract

A type theory is presented that combines (intuitionistic) linear types with type dependency, thus properly generalising both intuitionistic dependent type theory and full linear logic. A syntax and complete categorical semantics are developed, the latter in terms of (strict) indexed symmetric monoidal categories with comprehension. Various optional type formers are treated in a modular way. In particular, we will see that the historically much-debated multiplicative quantifiers and identity types arise naturally from categorical considerations. These new multiplicative connectives are further characterised by several identities relating them to the usual connectives from dependent type theory and linear logic. Finally, one important class of models, given by families with values in some symmetric monoidal category, is investigated in detail.

Disclaimer and Acknowledgements

The concept of a syntax with linear dependent types is not new. Such a calculus was first considered in [1], in the context of a linear extension of the Logical Framework (LF). In terms of the semantics, the author would like to point to [2] and [3], which study very similar semantic semantic objects, although without the notion of comprehension. Finally, the author would like to thank Urs Schreiber for sparking his interest in the topic through many enthusiastic posts on the nLab and nForum.

The contribution of the present work is as follows.

1. The presentation of a syntax in a style that is very close both to the dual intuitionistic linear logic of [4] and to that of intuitionistic dependent type theory in [5]. This clarifies, from a syntactic point of view, how exactly linear dependent types fit in with the work in both traditions.

2. Adding various structural rules, that allow us to consider the more basic setting without the various type formers of the Linear Logical Framework (LLF). In the rich setting of LLF, these become admissible.

3. The addition of various type formers to the syntax (most notably, Σ-, !-, and Id-types).

4. The development of the first categorical semantics for linear dependent types - although there have been some suggestions about this on the nLab, by Mike Shulman and Urs Schreiber, in particular, as far as the author is aware, no account of this has been published. The semantics is developed in a style that combines features of the linear-non-linear adjunctions of [4] and the comprehension categories of [6]. Among other things, it shows that multiplicative quantifiers arise very naturally, as adjoints to substitution, thereby offering a new point of view on long-debated issue of quantifiers in linear logic.

Contents

Introduction

To put this work in context, the best point of departure may be Church's simply typed λ-calculus (or intuitionistic propositional type theory), [7], which according to the Curry-Howard correspondence (e.g. [8]) can be thought of as a proof calculus for intuitionistic propositional logic. To this, Lambek (e.g. [9]) added the new idea that it can also be viewed as a syntax for describing Cartesian closed categories. From this starting point, two traditions depart that are particularly relevant for us.

On the one hand, so-called dependent type theory, can be seen to extend along the Curry-Howard correspondence, to provide a proof calculus for intuitionistic predicate logic. See e.g. [10]. Various flavours of categorical semantics for this calculus have been given, but they almost all boil down to the same essential ingredients of talking about certain fibred or, equivalently, indexed Cartesian categories, usually (depending on the exact formulation of the type theory) with a notion of comprehension that relates the fibres to the base category.

On the other hand, there is a school of logic, initiated by Girard, so-called linear logic, that has pursued a further, resource sensitive, analysis of the intuitionistic propositional type theory by exposing precisely how many times each assumption is used in proofs. See e.g. [11]. Later, the essence of Girard's system was seen to be captured by a less restrictive system called (multiplicative) intuitionistic linear logic (where Girard's original system is now referred to as classical linear logic). This will be the sense in which we use the word 'linear'. A satisfactory proof calculus and categorical semantics, in terms of symmetric monoidal closed categories, are easily given. The final essential element of linear type theories, the resource modality providing the connection with intuitionistic type theories, can be given categorical semantics as a monoidal adjunction between an intuitionistic and a linear model. [4]

The question rises if this linear analysis can be extended to predicate logic. Although some work has been done in this direction, the author feels a satisfactory answer has not been given.

Although Girard's early work in linear logic already talks about quantifiers, the analysis appears to have stayed rather superficial. In particular, an account of internal quantification, or a linear variant of Martin-Löf's type theory, was missing, let alone a Curry-Howard correspondence. Later, linear types and dependent types were first combined in [1], where a syntax was presented that extends LF (Logical Framework) with linear types (that depend on terms of intuitionistic types). This has given rise to a line of work in the computer science community. See e.g. [12, 13, 14]. All the work seems to be syntactic in nature, however.

On the other hand, very similar ideas, this time at the level of categorical semantics and specific models (coming from homotopy theory, algebra, and mathematical physics), have emerged in the mathematical community lately, perhaps independently so. Relevant literature is e.g. [15, 2, 3, 16].

Although, in the past months, some suggestions have been made on the nLab and nForum, of possible connections between both lines of work, no account of the correspondence was ever published, as far as the author is aware. Moreover, the syntactic tradition seems to have stayed restricted to the situation of functional type theory, in which one has Π- and $\multimap$-types, while we are really interested in the general case of algebraic type theory, which is of fundamental importance from the point of view of mathematics where structures seldom admit internal homs. Simultaneously, in the syntactic traditions there seems to be a lack of other type formers (like Σ-, !-, and Id-types). The semantic tradition, however, does not seem to have given a sufficient account of the notion of comprehension. The present text will take some steps to close this gap in the existing literature.

The point of this paper is to illustrate how linear and dependent types can be combined straightforwardly and in great generality. Firstly, in section 1, we start with some general discussion on the matter of combining linear and dependent types. Secondly, in section 2, we present a syntax, intuitionistic linear dependent type theory (ILDTT), a natural blend of the dual intuitionistic linear logic (DILL) of [4] and dependent type theory (DTT) of e.g. [5]. Thirdly, in section 3, we present a complete categorical semantics that is the obvious combination of linear/non-linear adjunctions of e.g. [4] and indexed Cartesian categories with comprehension of e.g. [6]. Finally, in section 4, an important class of (rather discrete) models is investigated, in terms of families with values in a fixed symmetric monoidal category.

This paper forms the first report of a research programme that aims to explore the interplay between type dependency and linearity. We we will therefore end with a brief discussion of some our work in progress and planned work on related topics.

0 Preliminaries

We will start these notes with some suggestions of references for both linear type theory and dependent type theory, as having the right point of view on both subjects greatly simplifies understanding our presentation of the new material.

0.1   (Non-Linear) Intuitionistic Dependent Type Theory (DTT)

Syntax

For an introductory but thorough treatment of the syntax, we refer the reader to [5].

Categorical Semantics

There are many (more or less) equivalent kinds of categorical model of a dependent type theory. Perhaps the best definition the have in mind while reading this text is either the notion of a full (split) comprehension category of [6] or the following rephrasing of a full split comprehension category with 1- and $\times$-types[1].

Definition (Strict Indexed Cartesian Monoidal Category with Comprehension)**.** By a strict[2] indexed[3] Cartesian monoidal category with comprehension, we will mean the following data.

1. A category of intuitionistic contexts $\mathcal{C}$ with a terminal object $\cdot$.

2. A strict indexed Cartesian monoidal category $\mathcal{I}$ over $\mathcal{C}$, i.e. a contravariant functor $\mathcal{I}$ into the category CMCat of (small) Cartesian monoidal categories and strong cartesian monoidal functors

$$\mathcal{C}^{op} \xrightarrow{\mathcal{I}} \text{CMCat}.$$

   We will also write $-\{f\} := \mathcal{I}(f)$ for the action of $\mathcal{I}$ on a morphism f of $\mathcal{C}$.

3. A *comprehension schema*, i.e. for each $\Delta \in \text{ob}(\mathcal{C})$ and $A \in \text{ob}(\mathcal{I}(\Delta))$ a representation for the functor

$$x \mapsto \mathcal{I}(\text{dom}(x))(1, A\{x\}) : (\mathcal{C}/\Delta)^{op} \longrightarrow \text{Set},$$

   which induces a fully faithful *comprehension functor*: if we write the representing object $\Delta.A \xrightarrow{\mathbf{p}_{\Delta,A}} \Delta \in \text{ob}(\mathcal{C}/\Delta)$, write $a \mapsto \langle f, a \rangle$ for the isomorphism $\mathcal{I}(\Delta')(1, A\{f\}) \cong \mathcal{C}/\Delta(f, \mathbf{p}_{\Delta,A})$, and write $\mathbf{v}_{\Delta,A} \in \mathcal{I}(\Delta.A)(1, A\{\mathbf{p}_{\Delta,A}\})$ for the universal element, the comprehension functor $\mathcal{I} \longrightarrow \mathcal{C}/-$ is defined as

$$\mathcal{I}(\Delta) \longrightarrow \mathcal{C}/\Delta$$

$$A \xrightarrow{f} B \quad \longmapsto \quad \mathbf{p}_{\Delta,A} \xrightarrow{\langle \mathbf{p}_{\Delta,A}, \mathcal{I}(\mathbf{p}_{\Delta,A})(f) \circ \mathbf{v}_{\Delta,A} \rangle} \mathbf{p}_{\Delta,B}.$$

Note that the comprehension schema says that we build up the morphisms into objects that arise as lists of types in our category of contexts $\mathcal{C}$ as lists of closed terms. It will be the crucial aspect that makes sure we are in the world of internal quantification. The fact that we are demanding the comprehension functor to be fully faithful means that also the non-closed terms in $\mathcal{I}(\Delta)(A, B)$ correspond precisely with the closed terms $\mathcal{I}(\Delta.A)(I, B)$. The equivalent condition on a comprehension category is the notion of fullness of [6]. This is essential to get a precise fit with the syntax. However, we shall drop this restriction in modelling linear dependent types, as, there, having A as a linear assumption (which will be an assumption in the fibre) can really be different from having A as an intuitionistic assumption (an assumption in the base).

---

[1]This restriction avoids us having to talk about multicategories. We are sure the reader familiar with multicategories can see the obvious generalisation to the case without 1- and $\times$-types.

[2]The strictness, here, refers to the fact that $\mathcal{I}$ is a functor, rather than a pseudofunctor. This reflects the fact that term substitution in types is a strict operation in type theory, even though the non-strict version might seem more natural from the point of view of category theory. It should be noted, though, that every indexed category is equivalent to a strict one. [17]

[3]Of course, using the Grothendieck construction, we could, equivalently, use split fibred categories, here, as is in fact done in certain accounts of semantics for DTT. However, indexed categories seem closer to the syntax and are technically less involved in the strict case.

0.2 Intuitionistic Linear (Non-Dependent) Type Theory (ILTT)

The most suitable flavour of ILTT to have in mind while reading these notes is the so-called dual intuitionistic linear logic (DILL) of [4]. This will be our principal reference for both syntax and semantics. Both the syntax and semantics given in this reference are very close in intuition to our syntax and semantics of linear dependent types. For the semantics, more background is provided in [18].

1 Intuitionistic Linear Dependent Type Theory?

Although it is a priori not entirely clear what linear dependent type theory should be, one can easily come up with some guiding criteria. My gut reaction was the following.

What we want to arrive at is a version of dependent type theory without weakening and contraction rules. Moreover, we would want an exponential co-modality on the type theory that gives us back the missing structural rules.

When one first tries to write down such a type theory, however, one will run into the following discrepancy.

- The lack of weakening and contraction rules in *linear type theory* forces us to refer to each declared variable precisely once: for a sequent $x : A \vdash t : B$, we know that x has a unique occurence in t.

- In *dependent type theory*, types can have free (term) variables: $x : A \vdash B$ type, where x is a free variable in B. Crucially, we can then talk about terms of B: $x : A \vdash b : B$, where generally x may also be free in b. For almost all interesting applications we will need multiple occurences of x to construct $b : B$, at least one for B and one for b.

The question now is what it means to refer to a declared variable only once.

Do we not count occurences in types? This point of view seems incompatible with universes, however, which play an important role in dependent type theory. If we do, however, the language seems to lose much of its expressive power. In particular, it prevents us from talking about constant types, it seems.

In this paper, we will circumvent the issue *by restricting to type dependency on terms of intuitionistic types*. In this case, there is no conflict, as those terms can be copied and deleted freely.

An argument for this system, that is of a semantic rather than a syntactic nature, is the following. We shall see that if we start out with a model of linear type theory with external quantification (i.e. strict indexed symmetric monoidal category) and demand the quantification to be internal (i.e. demand it to be a linear dependent type theory) in the natural way (i.e. impose the comprehension axiom), we will see that our base category becomes a cartesian category (i.e. a model of intuitionistic type theory).

Our best attempt to give meaning on type dependency on linear types, will be through dependency on a (co-Kleisli or co-Eilenberg-Moore) category of co-algbras for !. We will briefly return to the issue later, but most of the discussion is beyond the scope of this paper.

In a linear dependent type theory, one could initially conceive of the option of an additive - denote with $\Sigma^{\&}_{x:X} A$ - and multiplicative - denote with $\Sigma^{\otimes}_{x:X} A$ - Σ-type to generalise the additive and multiplicative conjunction of intuitionistic linear logic. The modality would relate the additive Σ-type of linear type families to the multiplicative Σ-type of intuitionistic type families, through a Seely-like isomorphism:

$$!(\Sigma^{\&}_{x:X} A) = \Sigma^{\otimes}_{x:!X} !A.$$

The idea is that the linear Σ-types should simultaneously generalise the conjunctions from propositional linear logic and the Σ-types from intuitionistic dependent type theory.

Generally, though, since we are restricting to type dependency on !-ed types, the additive Σ-type (as far as one can make sense of the notion) would become effectively intuitionistic (as both terms in the introduction rule would have to depend on the same context, one of which is forced to be intuitionistic). This also ruins the symmetry in the Seely isomorphism. Moreover, we will see that the multiplicative Σ-types (and Π-types) arise naturally in the categorical semantics. We will later briefly return to the possibility of a really linear additive Σ-type and Seely isomorphisms.

For the Π-type, one would also expect to obtain two linear cousins: an additive and a multiplicative version. However, given the overwhelming preference in the linear logic literature of the multiplicative implication, we can probably safely restrict our attention to multiplicative Π-types for now.

2 Syntax of ILDTT

We assume the reader has some familiarity with the formal syntax of dependent type theory and linear type theory. In particular, we will not go into syntactic details like α-conversion, name binding, capture-free substitution of a for x in t (write $t[a/x]$), and pre-syntax. The reader can find details on all of these topics in [5].

We next present the formal syntax of ILDTT. We start with a presentation of the judgements that will represent the propositions in the language and then discuss its rules of inference: first its structural core, then the logical rules for a series of optional type formers. We conclude this section with a few basic results about the syntax.

Judgements

We adopt a notation $\Delta; \Xi$ for contexts, where Δ is 'an intuitionistic region' and Ξ is 'a linear region', similarly to [4]. The idea will be that we have an empty context and can extend an existing context $\Delta; \Xi$ with both intuitionistic and linear types that are allowed to depend on Δ.

Our language will express judgements of the following six forms.

ILDTT judgement	Intended meaning
$\vdash \Delta; \Xi$ ctxt	$\Delta; \Xi$ is a valid context
$\Delta; \cdot \vdash A$ type	A is a type in (intuitionistic) context Δ
$\Delta; \Xi \vdash a : A$	a is a term of type A in context $\Delta; \Xi$
$\vdash \Delta; \Xi \equiv \Delta'; \Xi'$ ctxt	$\Delta; \Xi$ and $\Delta'; \Xi'$ are judgementally equal contexts
$\Delta; \cdot \vdash A \equiv A'$ type	A and A' are judgementally equal types in (intuitionistic) context Δ
$\Delta; \Xi \vdash a \equiv a' : A$	a and a' are judgementally equal terms of type A in context $\Delta; \Xi$

Figure 1: Judgements of ILDTT.

Structural Rules

We will use the following structural rules, which are essentially the structural rules of dependent type theory where some rules appear in both an intuitionistic and a linear form. We present the rules per group, with their names, from left-to-right, top-to-bottom.
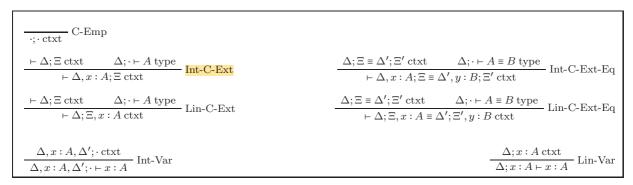
$$\frac{}{\cdot; \cdot \text{ ctxt}} \text{ C-Emp}$$

$$\frac{\vdash \Delta; \Xi \text{ ctxt} \qquad \Delta; \cdot \vdash A \text{ type}}{\vdash \Delta, x : A; \Xi \text{ ctxt}} \text{ Int-C-Ext} \qquad \frac{\Delta; \Xi \equiv \Delta'; \Xi' \text{ ctxt} \qquad \Delta; \cdot \vdash A \equiv B \text{ type}}{\vdash \Delta, x : A; \Xi \equiv \Delta', y : B; \Xi' \text{ ctxt}} \text{ Int-C-Ext-Eq}$$

$$\frac{\vdash \Delta; \Xi \text{ ctxt} \qquad \Delta; \cdot \vdash A \text{ type}}{\vdash \Delta; \Xi, x : A \text{ ctxt}} \text{ Lin-C-Ext} \qquad \frac{\Delta; \Xi \equiv \Delta'; \Xi' \text{ ctxt} \qquad \Delta; \cdot \vdash A \equiv B \text{ type}}{\vdash \Delta; \Xi, x : A \equiv \Delta'; \Xi', y : B \text{ ctxt}} \text{ Lin-C-Ext-Eq}$$

$$\frac{\Delta, x : A, \Delta'; \cdot \text{ ctxt}}{\Delta, x : A, \Delta'; \cdot \vdash x : A} \text{ Int-Var} \qquad \frac{\Delta; x : A \text{ ctxt}}{\Delta; x : A \vdash x : A} \text{ Lin-Var}$$

Figure 2: Context formation and variable declaration rules.

$$\frac{\vdash \Delta; \Xi \text{ ctxt}}{\vdash \Delta; \Xi \equiv \Delta; \Xi \text{ ctxt}} \text{ C-Eq-R} \qquad\qquad \frac{\vdash \Delta; \Xi \equiv \Delta'; \Xi' \text{ ctxt}}{\vdash \Delta'; \Xi' \equiv \Delta; \Xi \text{ ctxt}} \text{ C-Eq-S}$$

$$\frac{\vdash \Delta; \Xi \equiv \Delta'; \Xi' \text{ ctxt} \qquad \vdash \Delta'; \Xi' \equiv \Delta''; \Xi'' \text{ ctxt}}{\vdash \Delta; \Xi \equiv \Delta''; \Xi'' \text{ ctxt}} \text{ C-Eq-T}$$

$$\frac{\Delta; \Xi \vdash A \text{ type}}{\Delta; \Xi \vdash A \equiv A \text{ type}} \text{ Ty-Eq-R} \qquad\qquad \frac{\Delta; \Xi \vdash A \equiv A' \text{ type}}{\Delta; \Xi \vdash A' \equiv A \text{ type}} \text{ Ty-Eq-S}$$

$$\frac{\Delta; \Xi \vdash A \equiv A' \text{ type} \qquad \Delta; \Xi \vdash A' \equiv A'' \text{ type}}{\Delta; \Xi \vdash A \equiv A'' \text{ type}} \text{ Ty-Eq-T}$$

$$\frac{\Delta; \Xi \vdash a : A}{\Delta; \Xi \vdash a \equiv a : A} \text{ Tm-Eq-R} \qquad\qquad \frac{\Delta; \Xi \vdash a \equiv a' : A}{\Delta; \Xi \vdash a' \equiv a : A} \text{ Tm-Eq-S}$$

$$\frac{\Delta; \Xi \vdash a \equiv a' : A \qquad \Delta; \Xi \vdash a' \equiv a'' : A}{\Delta; \Xi \vdash a \equiv a'' : A} \text{ Tm-Eq-T}$$

$$\frac{\Delta; \Xi \vdash a : A \qquad \vdash \Delta; \Xi \equiv \Delta; \Xi' \text{ ctxt} \qquad \Delta; \cdot \vdash A \equiv A' \text{ Type}}{\Delta'; \Xi' \vdash a : A'} \text{ Tm-Conv}$$

$$\frac{\Delta'; \cdot \vdash A \text{ type} \qquad \vdash \Delta; \cdot \equiv \Delta'; \cdot \text{ ctxt}}{\Delta'; \cdot \vdash A \text{ type}} \text{ Ty-Conv}$$

Figure 3: A few standard rules for judgemental equality, saying that it is an equivalence relation and is compatible with typing.

$$\frac{\Delta, \Delta'; \Xi \vdash \mathcal{J} \qquad \Delta; \cdot \vdash A \text{ type}}{\Delta, x : A, \Delta'; \Xi \vdash \mathcal{J}} \text{ Int-Weak}$$

$$\frac{\Delta, x : A, x' : A', \Delta'; \Xi \vdash \mathcal{J}}{\Delta, x' : A', x : A, \Delta'; \Xi \vdash \mathcal{J}} \text{ Int-Exch} \qquad\qquad \frac{\Delta; \Xi, x : A, x' : A', \Xi' \vdash \mathcal{J}}{\Delta; \Xi, x' : A', x : A, \Xi' \vdash \mathcal{J}} \text{ Lin-Exch}$$
(if x is not free in A')

$$\frac{\Delta, x : A, \Delta'; \cdot \vdash B \text{ type} \qquad \Delta; \cdot \vdash a : A}{\Delta, \Delta'[a/x]; \cdot \vdash B[a/x] \text{ type}} \text{ Int-Ty-Subst} \qquad \frac{\Delta, x : A, \Delta'; \cdot \vdash B \equiv B' \text{ type} \qquad \Delta; \cdot \vdash a : A}{\Delta, \Delta'[a/x]; \cdot \vdash B[a/x] \equiv B'[a/x] \text{ type}} \text{ Int-Ty-Subst-Eq}$$

$$\frac{\Delta, x : A, \Delta'; \Xi \vdash b : B \qquad \Delta; \cdot \vdash a : A}{\Delta, \Delta'[a/x]; \Xi[a/x] \vdash b[a/x] : B[a/x]} \text{ Int-Tm-Subst} \qquad \frac{\Delta, x : A, \Delta'; \Xi \vdash b \equiv b' : B \qquad \Delta; \cdot \vdash a : A}{\Delta, \Delta'[a/x]; \Xi \vdash b[a/x] \equiv b'[a/x] : B[a/x]} \text{ Int-Tm-Subst-Eq}$$

$$\frac{\Delta; \Xi, x : A \vdash b : B \qquad \Delta; \Xi' \vdash a : A}{\Delta; \Xi, \Xi' \vdash b[a/x] : B} \text{ Lin-Tm-Subst} \qquad \frac{\Delta; \Xi, x : A \vdash b \equiv b' : B \qquad \Delta; \Xi' \vdash a : A}{\Delta; \Xi, \Xi' \vdash b[a/x] \equiv b'[a/x] : B} \text{ Lin-Tm-Subst-Eq}$$

Figure 4: Exchange, weakening, and substitution rules. Here, $\mathcal{J}$ represents a statement of the form B type, $B \equiv B'$, $b : B$, or $b \equiv b' : B$, such that all judgements are well-formed.

Logical Rules

We introduce some basic (optional) type and term formers, for which we will give type formation (denoted -F), term introduction (-I), term elimination (-E), term computation rules (-C), and (judgemental) term uniqueness principles (-U). We also assume the obvious rules to hold that state that the new type formers and term formers respect judgemental equality. Moreover, $\Sigma_{!x:!A}$, $\Pi_{!x:!A}$, $\lambda_{!x:!A}$, and $\lambda_{x:A}$ are name binding operators, binding free occurences of x within their scope. Preempting some theorems of the calculus, we overload some of the notation for -I and -E various type formers, in order to avoid a lot of syntactic clutter. Needless to say, uniqueness of typing can easily be restored by carrying around enough type information on the notations corresponding to the various -I and -E rules.

We demand -U-rules for the various type formers in this paper, as this allows us to give a natural categorical semantics. In practice, when building a computational implementation of a type theory like ours, one would probably drop these rules to make the system decidable, which would correspond to switching to weak equivalents of the categorical constructions presented here.[4]

$$\frac{\Delta, x:A; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash \Sigma_{!x:!A}B \text{ type}} \Sigma\text{-F}$$

$$\frac{\Delta; \cdot \vdash a:A \qquad \Delta; \Xi \vdash b:B[a/x]}{\Delta; \Xi \vdash !a \otimes b : \Sigma_{!x:!A}B} \Sigma\text{-I}$$

$$\frac{\Delta; \Xi \vdash \text{let } !a \otimes b \text{ be } !x \otimes y \text{ in } c:C}{\Delta; \Xi \vdash \text{let } !a \otimes b \text{ be } !x \otimes y \text{ in } c \equiv c[a/x, b/y]:C} \Sigma\text{-C}$$

$$\frac{\begin{array}{c}\Delta; \cdot \vdash C \text{ type} \\ \Delta; \Xi \vdash t : \Sigma_{!x:!A}B \\ \Delta, x:A; \Xi', y:B \vdash c:C\end{array}}{\Delta; \Xi, \Xi' \vdash \text{let } t \text{ be } !x \otimes y \text{ in } c:C} \Sigma\text{-E}$$

$$\frac{\Delta; \Xi \vdash \text{let } t \text{ be } !x \otimes y \text{ in } !x \otimes y : \Sigma_{!x:!A}B}{\Delta; \Xi \vdash \text{let } t \text{ be } !x \otimes y \text{ in } !x \otimes y \equiv t : \Sigma_{!x:!A}B} \Sigma\text{-U}$$

$$\frac{\Delta, x:A; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash \Pi_{!x:!A}B \text{ type}} \Pi\text{-F}$$

$$\frac{\vdash \Delta; \Xi \text{ ctxt} \qquad \Delta, x:A; \Xi \vdash b:B}{\Delta; \Xi \vdash \lambda_{!x:!A}b : \Pi_{!x:!A}B} \Pi\text{-I}$$

$$\frac{\Delta; \Xi \vdash (\lambda_{!x:!A}b)(!a):B}{\Delta; \Xi \vdash (\lambda_{!x:!A}b)(!a) \equiv b[a/x] : B[a/x]} \Pi\text{-C}$$

$$\frac{\Delta; \cdot \vdash a:A \qquad \Delta; \Xi \vdash f : \Pi_{!x:!A}B}{\Delta; \Xi \vdash f(!a) : B[a/x]} \Pi\text{-E}$$

$$\frac{\Delta; \Xi \vdash \lambda_{!x:!A}f(!x) : \Pi_{!x:!A}B}{\Delta; \Xi \vdash f \equiv \lambda_{!x:!A}f(!x) : \Pi_{!x:!A}B} \Pi\text{-U}$$

$$\frac{\Delta; \cdot \vdash a:A \qquad \Delta; \cdot \vdash a':A}{\Delta; \cdot \vdash \text{Id}_{!A}(a,a') \text{ type}} \text{Id-F}$$

$$\frac{\Delta; \cdot \vdash a:A}{\Delta; \cdot \vdash \text{refl}_{!a} : \text{Id}_{!A}(a,a)} \text{Id-I}$$

$$\frac{\begin{array}{c}\Delta, x:A, x':A; \cdot \vdash D \text{ type} \\ \Delta, z:A; \Xi \vdash d : D[z/x, z/x'] \\ \Delta; \cdot \vdash a:A \\ \Delta; \cdot \vdash a':A \\ \Delta; \Xi' \vdash p : \text{Id}_{!A}(a,a')\end{array}}{\Delta; \Xi[a/z], \Xi' \vdash \text{let } (a,a',p) \text{ be } (z,z,\text{refl}_{!z}) \text{ in } d : D[a/x, a'/x']} \text{Id-E}$$

$$\frac{\Delta; \Xi \vdash \text{let } (a,a,\text{refl}_{!a}) \text{ be } (z,z,\text{refl}_{!z}) \text{ in } d : D[a/x, a/x']}{\Delta; \Xi \vdash \text{let } (a,a,\text{refl}_{!a}) \text{ be } (z,z,\text{refl}_{!z}) \text{ in } d \equiv d[a/z] : D[a/x, a/x']} \text{Id-C}$$

$$\frac{\Delta, x:A, x':A; \Xi, z:\text{Id}_{!A}(x,x') \vdash \text{let } (x,x',z) \text{ be } (x,x,\text{refl}_{!x}) \text{ in } c[x/x', \text{refl}_{!x}/z] : C}{\Delta, x:A, x':A; \Xi, z:\text{Id}_{!A}(x,x') \vdash \text{let } (x,x',z) \text{ be } (x,x,\text{refl}_{!x}) \text{ in } c[x/x', \text{refl}_{!x}/z] \equiv c : C} \text{Id-U}$$

Figure 5: Rules for linear equivalents of some of the usual type formers from DTT: Σ-, Π-, and Id-types.

---

[4]In that case in DTT, one would usually demand some stronger 'dependent' elimination rules, which would make propositional equivalents of the -U-rules provable, adding some extensionality to the system, while preserving its computational properties. Such rules are problematic in ILDTT, however, both from a syntactic and semantic point of view and a further investigation is warranted here.

$$\frac{}{\Delta; \cdot \vdash I \text{ type}} \; I\text{-F}$$

$$\frac{}{\Delta; \cdot \vdash * : I} \; I\text{-I}$$

$$\frac{\Delta; \Xi' \vdash t : I \qquad \Delta; \Xi \vdash a : A}{\Delta; \Xi, \Xi' \vdash \text{let } t \text{ be } * \text{ in } a : A} \; I\text{-E}$$

$$\frac{\Delta; \Xi \vdash \text{let } * \text{ be } * \text{ in } a : A}{\Delta; \Xi \vdash \text{let } * \text{ be } * \text{ in } a \equiv a : A} \; I\text{-C}$$

$$\frac{\Delta; \Xi \vdash \text{let } t \text{ be } * \text{ in } * : I}{\Delta; \Xi \vdash \text{let } t \text{ be } * \text{ in } * \equiv t : I} \; I\text{-U}$$

$$\frac{\Delta; \cdot \vdash A \text{ type} \qquad \Delta; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash A \otimes B \text{ type}} \; \otimes\text{-F}$$

$$\frac{\Delta; \Xi \vdash a : A \qquad \Delta; \Xi' \vdash b : B}{\Delta; \Xi, \Xi' \vdash a \otimes b : A \otimes B} \; \otimes\text{-I}$$

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

$$\frac{\Delta; \Xi \vdash \text{let } a \otimes b \text{ be } x \otimes y \text{ in } c : C}{\Delta; \Xi \vdash \text{let } a \otimes b \text{ be } x \otimes y \text{ in } c \equiv c[a/x, b/y] : C} \; \otimes\text{-C}$$

$$\frac{\Delta; \Xi \vdash \text{let } t \text{ be } x \otimes y \text{ in } x \otimes y : A \otimes B}{\Delta; \Xi \vdash \text{let } t \text{ be } x \otimes y \text{ in } x \otimes y \equiv t : A \otimes B} \; \otimes\text{-U}$$

$$\frac{\Delta; \cdot \vdash A \text{ type} \qquad \Delta; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash A \multimap B \text{ type}} \; \multimap\text{-F}$$

$$\frac{\Delta; \Xi, x : A \vdash b : B}{\Delta; \Xi \vdash \lambda_{x:A} b : A \multimap B} \; \multimap\text{-I}$$

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

$$\frac{\Delta; \Xi \vdash (\lambda_{x:A} b)(a) : B}{\Delta; \Xi \vdash (\lambda_{x:A} b)(a) \equiv b[a/x] : B} \; \multimap\text{-C}$$

$$\frac{\Delta; \Xi \vdash \lambda_{x:A} f x : A \multimap B}{\Delta; \Xi \vdash \lambda_{x:A} f x \equiv f : A \multimap B} \; \multimap\text{-U}$$

$$\frac{}{\Delta; \cdot \vdash \top \text{ type}} \; \top\text{-F}$$

$$\frac{\Delta; \Xi \text{ ctxt}}{\Delta; \Xi \vdash \langle \rangle : \top} \; \top\text{-I}$$

$$\frac{\Delta; \Xi \vdash t : \top}{\Delta; \Xi \vdash t \equiv \langle \rangle : \top} \; \top\text{-U}$$

$$\frac{\Delta; \cdot \vdash A \text{ type} \qquad \Delta; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash A \& B \text{ type}} \; \&\text{-F}$$

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

$$\frac{\Delta; \Xi \vdash t : A \& B}{\Delta; \Xi \vdash \text{fst}(t) : A} \; \&\text{-E1}$$

$$\frac{\Delta; \Xi \vdash t : A \& B}{\Delta; \Xi \vdash \text{snd}(t) : B} \; \&\text{-E2}$$

$$\frac{\Delta; \Xi \vdash \text{fst}(\langle a, b \rangle) : A}{\Delta; \Xi \vdash \text{fst}(\langle a, b \rangle) \equiv a : A} \; \&\text{-C1}$$

$$\frac{\Delta; \Xi \vdash \text{snd}(\langle a, b \rangle) : B}{\Delta; \Xi \vdash \text{snd}(\langle a, b \rangle) \equiv b : B} \; \&\text{-C2}$$

$$\frac{\Delta; \Xi \vdash \langle \text{fst}(t), \text{snd}(t) \rangle : A \& B}{\Delta; \Xi \vdash \langle \text{fst}(t), \text{snd}(t) \rangle \equiv t : A \& B} \; \&\text{-U}$$

$$\frac{}{\Delta; \cdot \vdash 0 \text{ type}} \; 0\text{-F}$$

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

$$\frac{\Delta; \Xi \vdash t : 0}{\Delta; \Xi \vdash \text{false}(t) \equiv t : 0} \; 0\text{-U}$$

$$\frac{\Delta; \cdot \vdash A \text{ type} \qquad \Delta; \cdot \vdash B \text{ type}}{\Delta; \cdot \vdash A \oplus B \text{ type}} \; \oplus\text{-F}$$

$$\frac{\Delta; \Xi \vdash a : A}{\Delta; \Xi \vdash \text{inl}(a) : A \oplus B} \; \oplus\text{-I1}$$

$$\frac{\Delta; \Xi \vdash b : B}{\Delta; \Xi \vdash \text{inr}(b) : A \oplus B} \; \oplus\text{-I2}$$

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

$$\frac{\Delta; \Xi, \Xi' \vdash \text{case } \text{inl}(a) \text{ of } \text{inl}(x) \to c \;\|\; \text{inr}(y) \to d : C}{\Delta; \Xi, \Xi' \vdash \text{case } \text{inl}(a) \text{ of } \text{inl}(x) \to c \;\|\; \text{inr}(y) \to d \equiv c[a/x] : C} \; \oplus\text{-C1}$$

$$\frac{\Delta; \Xi, \Xi' \vdash \text{case } \text{inr}(b) \text{ of } \text{inl}(x) \to c \;\|\; \text{inr}(y) \to d : C}{\Delta; \Xi, \Xi' \vdash \text{case } \text{inr}(b) \text{ of } \text{inl}(x) \to c \;\|\; \text{inr}(y) \to d \equiv d[b/y] : C} \; \oplus\text{-C2}$$

$$\frac{\Delta; \Xi, \Xi' \vdash \text{case } t \text{ of } \text{inl}(x) \to \text{inl}(x) \;\|\; \text{inr}(y) \to \text{inr}(y) : A \oplus B}{\Delta; \Xi, \Xi' \vdash \text{case } t \text{ of } \text{inl}(x) \to \text{inl}(x) \;\|\; \text{inr}(y) \to \text{inr}(y) \equiv t : A \oplus B} \; \oplus\text{-U}$$

$$\frac{\Delta;\cdot \vdash A \text{ type}}{\Delta;\cdot \vdash\, !A \text{ type}} \text{ !-F}$$

$$\frac{\Delta;\cdot \vdash a : A}{\Delta;\cdot \vdash\, !a :\, !A} \text{ !-I} \qquad\qquad \frac{\Delta;\Xi \vdash t :\, !A \qquad \Delta, x : A;\Xi' \vdash b : B}{\Delta;\Xi,\Xi' \vdash \text{let } t \text{ be } !x \text{ in } b : B} \text{ !-E}$$

$$\frac{\Delta;\Xi \vdash \text{let } !a \text{ be } !x \text{ in } b : B}{\Delta;\Xi \vdash \text{let } !a \text{ be } !x \text{ in } b \equiv b[a/x] : B} \text{ !-C} \qquad\qquad \frac{\Delta;\Xi \vdash \text{let } t \text{ be } !x \text{ in } !x :\, !A}{\Delta;\Xi \vdash \text{let } t \text{ be } !x \text{ in } !x \equiv t :\, !A} \text{ !-U}$$

Figure 6: Rules for the usual linear type formers in each context: I-, $\otimes$-, $\multimap$-, $\top$-, $\&$-, 0-, $\oplus$-, and !-types.

Finally, we add rules that say we have all the possible commuting conversions, which from a syntactic point of view restore the subformula property and from a semantic point of view say that our rules are natural transformations (between hom-functors), which simplifies the categorical semantics significantly. We represent these schematically, following [4]. That is, if $C[-]$ is a linear (program rather than typing) context, then (abusing notation and dealing with all the let be in -constructors in one go).

$$\frac{\Delta;\Xi \vdash C[\text{let } a \text{ be } b \text{ in } c] : D}{\Delta;\Xi \vdash C[\text{let } a \text{ be } b \text{ in } c] \equiv \text{let } a \text{ be } b \text{ in } C[c] : D}$$

if $C[-]$ does not bind any free variables in a or b;

$$\frac{\Delta;\Xi \vdash C[\text{false}(t)] : D}{\Delta;\Xi \vdash C[\text{false}(t)] \equiv \text{false}(t) : D}$$

if $C[-]$ does not bind any free variables in t;

$$\frac{\Delta;\Xi \vdash C[\text{case } t \text{ of } \text{inl}(x) \to c \,\|\, \text{inr}(y) \to d] : D}{\Delta;\Xi \vdash C[\text{case } t \text{ of } \text{inl}(x) \to c \,\|\, \text{inr}(y) \to d] \equiv \text{case } t \text{ of } \text{inl}(x) \to C[c] \,\|\, \text{inr}(y) \to C[d] : D}$$

if $C[-]$ does not bind any free variables in t or x or y.

Figure 7: Commuting conversions.

Remark 1. Note that all type formers that are defined context-wise (I, $\otimes$, $\multimap$, $\top$, $\&$, 0, $\oplus$, and !) are automatically preserved under the substitutions from Int-Ty-Subst (up to canonical isomorphism[5]), in the sense that $F(A_1, \ldots, A_n)[a/x]$ is isomorphic to $F(A_1[a/x], \ldots, A_n[a/x])$ for an n-ary type former F. Similarly, for $T = \Sigma$ or Π, we have that $(T_{!y:!B}C)[a/x]$ is isomorphic to $T_{!y:!B[a/x]}C[a/x]$ and $(Id_{!B}(b, b'))[a/x]$ is isomorphic to $Id_{!B[a/x]}(b[a/x], b'[a/x])$. (This gives us Beck-Chevalley conditions in the categorical semantics.) These are the remaining naturality conditions for rules.

Remark 2. The reader can note that the usual formulation of universes for DTT transfers very naturally to ILDTT, giving us a notion of universes for linear types. This allows us to write rules for forming types as rules for forming terms, as usual. We do not choose this approach and define the various type formers in the setting without universes, as this will give a cleaner categorical semantics.

Try writing this out

Some Basic Results

As the focus of this paper is the syntax-semantics correspondence, we will only briefly state a few syntactic results. For some standard metatheoretic properties for the $\multimap, \Pi, \top, \&$-fragment of our syntax, we refer the reader to [1]. Standard techniques and some small adaptations of the system should be enough to extend the results to all of ILDTT.

Theorem 1 (Consistency). *ILDTT with all its type formers is consistent.*

Proof. We discuss a class of models in section 4. $\qquad\square$

[5]By an isomorphism of types $\Delta;\cdot \vdash A$ type and $\Delta;\cdot \vdash B$ type in context Δ, we here mean a pair of terms $\Delta; x : A \vdash f : B$ and $\Delta; y : B \vdash g : A$ together with a pair of judgemental equalities $\Delta; x : A \vdash g[f/y] \equiv x : A$ and $\Delta; y : B \vdash f[g/x] \equiv y : B$.

To give the reader some intuition for these linear Π- and Σ-types, we suggest the following two interpretations.

Theorem 2 (Π and Σ as Dependent $!(-) \multimap (-)$ and $!(-) \otimes (-)$). *Suppose we have !-types. Let $\Delta, x : A; \cdot \vdash B$ type, where x does not occur freely in B. Then, for the purposes of the type theory,*

1. *$\Pi_{!x:!A}B$ is isomorphic to $!A \multimap B$, if we have Π-types and $\multimap$-types;*

2. *$\Sigma_{!x:!A}B$ is isomorphic to $!A \otimes B$, if we have Σ-types and $\otimes$-types.*

Proof. 1. We will construct terms

$$\Delta; y : \Pi_{!x:!A}B \vdash f : !A \multimap B \quad \text{and} \quad \Delta; y' : !A \multimap B \vdash g : \Pi_{!x:!A}B$$

s.t.

$$\Delta; y : \Pi_{!x:!A}B \vdash g[f/y'] \equiv y : \Pi_{!x:!A}B \quad \text{and} \quad \Delta; y' : !A \multimap B \vdash f[g/y] \equiv y' : !A \multimap B.$$

First, we construct f.

$$\cfrac{\cfrac{\cfrac{\overline{\Delta, x : A; \cdot \vdash x : A}\ \text{Int-Var} \quad \overline{\Delta, x : A; y : \Pi_{!x:!A}B \vdash y : \Pi_{!x:!A}B}\ \text{Lin-Var}}{\Delta, x : A; y : \Pi_{!x:!A}B \vdash y(!x) : B}\ \Pi\text{-E} \quad \overline{\Delta; x' : !A \vdash x' : !A}\ \text{Lin-Var}}{\Delta; y : \Pi_{!x:!A}B, x' : !A \vdash \text{let } x' \text{ be } !x \text{ in } y(!x) : B}\ !\text{-E}}{\Delta; y : \Pi_{!x:!A}B \vdash f : !A \multimap B}\ \multimap\text{-I}$$

Then, we construct g.

$$\cfrac{\cfrac{\cfrac{\overline{\Delta, x : A; \cdot \vdash x : A}\ \text{Int-Var}}{\Delta, x : A; \cdot \vdash !x : !A}\ !\text{-I} \quad \overline{\Delta, x : A; y' : !A \multimap B \vdash y' : !A \multimap B}\ \text{Lin-Var}}{\Delta, x : A; y' : !A \multimap B \vdash y'(!x) : B}\ \multimap\text{-E}}{\Delta; y' : !A \multimap B \vdash g : \Pi_{!x:!A}B}\ \Pi\text{-I}$$

It is easily verified that $\multimap$-C, !-C, and Π-U imply the first judgemental equality:
$g[f/y'] \equiv \lambda_{!x:!A}(\lambda_{x':!A}\text{let } x' \text{ be } !x \text{ in } y(!x))(!x) \equiv \lambda_{!x:!A}\text{let } !x \text{ be } !x \text{ in } y(!x) \equiv \lambda_{!x:!A}y(!x) \equiv y.$

Similarly, Π-C, commuting conversions, !-U, and $\multimap$-U imply the second judgemental equality:
$f[g/y] \equiv \lambda_{x':!A}\text{let } x' \text{ be } !x \text{ in } (\lambda_{!x:!A}y'(!x))(!x) \equiv \lambda_{x':!A}\text{let } x' \text{ be } !x \text{ in } y'(!x) \equiv \lambda_{x':!A}y'(\text{let } x' \text{ be } !x \text{ in } !x)$
$\equiv \lambda_{x':!A}y'(x') \equiv y'.$

2. We will construct terms

$$\Delta; y : \Sigma_{!x:!A}B \vdash f : !A \otimes B \quad \text{and} \quad \Delta; y' : !A \otimes B \vdash g : \Sigma_{!x:!A}B$$

s.t.

$$\Delta; y : \Sigma_{!x:!A}B \vdash g[f/y'] \equiv y : \Sigma_{!x:!A}B \quad \text{and} \quad \Delta; y' : !A \otimes B \vdash f[g/y] \equiv y' : !A \otimes B.$$

First, we construct f.

$$\cfrac{\overline{\Delta; y : \Sigma_{!x:!A}B \vdash y : \Sigma_{!x:!A}B}\ \text{Lin-Var} \quad \cfrac{\cfrac{\cfrac{\overline{\Delta; x' : !A \vdash x' : !A}\ \text{Lin-Var} \quad \overline{\Delta; z : B \vdash z : B}\ \text{Lin-Var}}{\Delta; x' : !A, z : B \vdash x' \otimes z : !A \otimes B}\ \otimes\text{-I}}{\Delta, x : A; x' : !A, z : B \vdash x' \otimes z : !A \otimes B}\ \text{Int-Weak} \quad \cfrac{\overline{\Delta, x : A; \cdot \vdash x : A}\ \text{Int-Var}}{\Delta, x : A; \cdot \vdash !x : !A}\ !\text{-I}}{\Delta, x : A; z : B \vdash !x \otimes z : !A \otimes B}\ \text{Lin-Subst}}{\Delta; y : \Sigma_{!x:!A}B \vdash f : !A \otimes B}\ \Sigma\text{-E}$$

Then, we construct g.

$$\cfrac{\overline{\Delta; y' : !A \otimes B \vdash y' : !A \otimes B}\ \text{Lin-Var} \quad \cfrac{\cfrac{\cfrac{\overline{\Delta, x : A; \cdot \vdash x : A}\ \text{Int-Var} \quad \overline{\Delta, x : A; y : B \vdash y : B}\ \text{Lin-Var}}{\Delta, x : A; y : B \vdash !x \otimes y : \Sigma_{!x:!A}B}\ \Sigma\text{-I} \quad \overline{\Delta; x' : !A \vdash x' : !A}\ \text{Lin-Var}}{\Delta; x' : !A, y : B \vdash \text{let } x' \text{ be } !x \text{ in } !x \otimes y : \Sigma_{!x:!A}B}\ !\text{-E}}{\Delta; y' : !A \otimes B \vdash g : \Sigma_{!x:!A}B}\ \otimes\text{-E}$$

Here, the first judgemental equality follows from commuting conversions, $\otimes$-C, !-C, and Σ-U:

$g[f/y'] \equiv$ let (let y be $!x \otimes z$ in $!x \otimes z$) be $x' \otimes y$ in (let x' be $!x$ in $!x \otimes y$) $\equiv$ let y be $!x \otimes z$ in let $!x \otimes z$ be $x' \otimes y$ in (let x' be $!x$ in $!x \otimes y$) $\equiv$ let y be $!x \otimes z$ in (let x' be $!x$ in $!x \otimes y$)$[!x/x'][z/y] \equiv$ let y be $!x \otimes z$ in (let $!x$ be $!x$ in $!x \otimes z$) $\equiv$ let y be $!x \otimes z$ in $!x \otimes z \equiv y$.

The second judgemental equality follows from commuting conversions, Σ-C, !-U, and $\otimes$-U:
$f[g/y] \equiv$ let (let y' be $x' \otimes y$ in (let x' be $!x$ in $!x \otimes y$)) be $!x \otimes z$ in $!x \otimes z \equiv$ let y' be $x' \otimes y$ in let x' be $!x$ in let $!x \otimes y$ be $!x \otimes z$ in $!x \otimes z \equiv$ let y' be $x' \otimes y$ in let x' be $!x$ in $(!x \otimes y) \equiv$ let y' be $x' \otimes y$ in (let x' be $!x$ in $!x) \otimes y \equiv$ let y' be $x' \otimes y$ in $x' \otimes y \equiv y'$.
$\square$

In particular, we have the following stronger version of a special case.

Theorem 3 (! as ΣI)**.** *Suppose we have* Σ- *and* I-*types. Let* $\Delta; \cdot \vdash A$ *type. Then,* $\Sigma_{!x:!A}I$ *satisfies the rules for* $!A$*. Conversely, if we have* !- *and* I-*types, then* $!A$ *satisfies the rules for* $\Sigma_{!x:!A}I$*.*

Proof. We obtain the !-rule as follows.

$$\frac{\Delta; \cdot \vdash a : A \qquad \dfrac{}{\Delta, x : A; \cdot \vdash * : I}\; I\text{-I}}{\Delta; \cdot \vdash !a \otimes * : \Sigma_{!x:!A}I}\; \Sigma\text{-I}$$

We obtain the !-E rule as follows.

$$\frac{\Delta; \Xi \vdash t : \Sigma_{!x:!A}I \qquad \dfrac{\Delta, x : A; \Xi' \vdash c : C \qquad \dfrac{}{\Delta; y : I \vdash y : I}\; \text{Lin-Var}}{\Delta, x : A; \Xi', y : I \vdash \text{let } y \text{ be } * \text{ in } c : C}\; I\text{-E}}{\Delta; \Xi, \Xi' \vdash \text{let } t \text{ be } !x \otimes y \text{ in let } y \text{ be } * \text{ in } c : C.}\; \Sigma\text{-E}$$

It is easily seen that Σ-C and I-C imply !-C (let $!a \otimes *$ be $!x \otimes y$ in let y be $*$ in $c \equiv$ (let y be $*$ in c)$[a/x][*/y] \equiv$ let $*$ be $*$ in $c[a/x] \equiv c[a/x]$) and that I-U and Σ-U (and commuting conversions) imply !-U (let t be $!x \otimes y$ in let y be $*$ in $x \otimes * \equiv$ let t be $!x \otimes y$ in $!x \otimes$ let y be $*$ in $* \equiv$ let t be $!x \otimes y$ in $!x \otimes y \equiv t$).

The converse statement follows through a similarly trivial argument, noting that $I[a/x]$ is isomorphic to I.
$\square$

A second interpretation is that Π and Σ generalise $\&$ and $\oplus$. Indeed, the idea is that (or their infinitary equivalents) is what they reduce to when taken over discrete types. The subtlety in this result will be the definition of a discrete type. The same phenomenon is observed in a different context in section 4.

For our purposes, a discrete type is a strong sum of I (a sum with a dependent -E-rule). Let us for simplicity limit ourselves to the binary case. For us, the discrete type with two elements will be $2 = I \oplus I$, where $\oplus$ has a strong/dependent -E-rule (note that this is not our $\oplus$-E). Explicitly, 2 is a type with the following rules:

$$\frac{}{\Delta; \cdot \vdash 2 \text{ type}}\; 2\text{-F} \qquad\qquad \frac{}{\Delta; \cdot \vdash \text{tt} : 2}\; 2\text{-I1} \qquad\qquad \frac{}{\Delta; \cdot \vdash \text{ff} : 2}\; 2\text{-I2}$$

$$\frac{\Delta, x : 2; \cdot \vdash A \text{ type} \qquad \Delta; \cdot \vdash t : 2 \qquad \Delta; \Xi \vdash a_{\text{tt}} : A[\text{tt}/x] \qquad \Delta; \Xi \vdash a_{\text{ff}} : A[\text{ff}/x]}{\Delta; \Xi \vdash \text{if } t \text{ then } a_{\text{tt}} \text{ else } a_{\text{ff}} : A[t/x]}\; 2\text{-E}$$

$$\frac{\Delta; \Xi \vdash \text{if tt then } a_{\text{tt}} \text{ else } a_{\text{ff}} : A[\text{tt}/x]}{\Delta; \Xi \vdash \text{if tt then } a_{\text{tt}} \text{ else } a_{\text{ff}} \equiv a_{\text{tt}} : A[\text{tt}/x]}\; 2\text{-C1} \qquad\qquad \frac{\Delta; \Xi \vdash \text{if ff then } a_{\text{tt}} \text{ else } a_{\text{ff}} : A[\text{ff}/x]}{\Delta; \Xi \vdash \text{if ff then } a_{\text{tt}} \text{ else } a_{\text{ff}} \equiv a_{\text{ff}} : A[\text{ff}/x]}\; 2\text{-C2}$$

$$\frac{\Delta; \Xi \vdash \text{if } t \text{ then tt else ff} : 2}{\Delta; \Xi \vdash \text{if } t \text{ then tt else ff} \equiv t : 2}\; 2\text{-U}$$

Figure 8: Rules for a discrete type 2.

Theorem 4 (Π and Σ as Infinitary Non-Discrete $\&$ and $\oplus$)**.** *If we have a discrete type* 2 *and a type family* $\Delta, x : 2; \cdot \vdash A$*, then*

1. $\Pi_{!x:!2}A$ *satisfies the rules for* $A[\text{tt}/x]\&A[\text{ff}/x]$*;*

14

2. $\Sigma_{!x:!2}A$ *satisfies the rules for* $A[\text{tt}/x] \oplus A[\text{ff}/x]$.

Proof.   1. We obtain &-I as follows.

$$\frac{\Delta, x:2; \Xi \vdash a : A[\text{tt}/x] \qquad \Delta, x:2; \Xi \vdash b : A[\text{ff}/x] \qquad \dfrac{}{\Delta, x:2; \cdot \vdash x:2}\text{ Int-Var} \qquad \dfrac{}{\Delta, x:2; \cdot \vdash A\text{ type}}\,\substack{\text{Assumption}\\\text{2-E-dep}}}{\dfrac{\Delta, x:2; \Xi \vdash \text{if } x \text{ then } a \text{ else } b : A}{\Delta; \Xi \vdash \lambda_{!x:!2}\text{if } x \text{ then } a \text{ else } b : \Pi_{!x:!2}A}\text{ \Pi-I}}$$

Moreover, we obtain &-E1 as follows (similarly, we obtain &-E2).

$$\frac{\Delta; \Xi \vdash t : \Pi_{!x:!2}A \qquad \dfrac{}{\Delta; \cdot \vdash \text{tt} : 2}\,2\text{-I1}}{t(!\text{tt})}\,\Pi\text{-E}$$

The &-C-rules follow from Π-C and 2-C, e.g.
$\text{fst}\langle a, b \rangle :\equiv (\lambda_{!x:!2}\text{if } x \text{ then } a \text{ else } b)(!\text{tt}) \equiv \text{if tt then } a \text{ else } b \equiv a.$

The &-U-rules follow from Π-U and 2-U (and commuting conversions):
$\langle \text{fst}(t), \text{snd}(t) \rangle :\equiv \lambda_{!x:!2}\text{if } x \text{ then } t(!\text{tt}) \text{ else } t(!\text{ff}) \equiv \lambda_{!x:!2}t(!x) \equiv t.$

2. We obtain $\oplus$-I1 as follows (and similarly, we obtain $\oplus$-I2):

$$\frac{\dfrac{}{\Delta; \cdot \vdash \text{tt} : 2}\,2\text{-I1} \qquad \Delta; \Xi \vdash a : A[\text{tt}/x]}{\Delta; \Xi \vdash \,!\text{tt} \otimes a : \Sigma_{!x:!2}A}\,\Sigma\text{-I}$$

Moreover, we obtain $\oplus$-E as follows.

$$\frac{\Delta; \Xi' \vdash t : \Sigma_{!x:!2}A \qquad \dfrac{\dfrac{\Delta; \Xi, z:A[\text{tt}/x] \vdash c : C}{\Delta, x:2; \Xi, y:A \vdash c[y/z] : C} \quad \dfrac{\Delta; \Xi, w:A[\text{ff}/x] \vdash d : C}{\Delta, x:2; \Xi, y:A \vdash d[y/w] : C} \quad \dfrac{}{\Delta, x:2; \cdot \vdash x:2} \quad \dfrac{}{\Delta, x:2; \cdot \vdash A\text{ type}}\,\substack{\text{Assump}\\\text{2-E-dep}}}{\Delta, x:2; \Xi, y:A \vdash \text{if } x \text{ then } c[y/z] \text{ else } d[y/w] : C}\,\Sigma\text{-E}}{\Delta; \Xi, \Xi' \vdash \text{let } t \text{ be } !x \otimes y \text{ in if } x \text{ then } c[y/z] \text{ else } d[y/w] : C}$$

The $\oplus$-C-rules follow from Σ-C and 2-C, e.g.

$$\begin{aligned}
\text{case inl}(a) \text{ of inl}(z) \to c \| \text{ inr}(w) \to d : &\equiv \text{let } !\text{tt} \otimes a \text{ be } !x \otimes y \text{ in if } x \text{ then } c[y/z] \text{ else } d[y/w] \\
&\equiv \text{if tt then } c[a/z] \text{ else } d[a/w] \\
&= c[a/z].
\end{aligned}$$

The $\oplus$-U-rules follow from Σ-U and 2-U (and commuting conversions):

$$\begin{aligned}
\text{case } t \text{ of inl}(z) \to \text{inl}(z) \| \text{ inr}(w) \to \text{inr}(w) : &\equiv \text{let } t \text{ be } !x \otimes y \text{ in if } x \text{ then inl}(z)[y/z] \text{ else inr}(w)[y/w] \\
&\equiv \text{let } t \text{ be } !x \otimes y \text{ in if } x \text{ then } !\text{tt} \otimes z[y/z] \text{ else } !\text{ff} \otimes w[y/w] \\
&\equiv \text{let } t \text{ be } !x \otimes y \text{ in if } x \text{ then } !\text{tt} \otimes y \text{ else } !\text{ff} \otimes y \\
&\equiv \text{let } t \text{ be } !x \otimes y \text{ in } !(\text{if } x \text{ then tt else ff}) \otimes y \\
&\equiv \text{let } t \text{ be } !x \otimes y \text{ in } !x \otimes y \\
&\equiv t.
\end{aligned}$$

$\square$

We see that we can view Π and Σ as generalisations of & and $\oplus$, respectively.

3 Semantics of ILDTT

The idea behind the categorical semantics we present for the structural core of our syntax (with I- and $\otimes$-types) will be to take our suggested categorical semantics for the structural core of DTT (with $\top$- and $\wedge$-types) and relax the assumption of the Cartesian character of its fibres to them only being (possibly non-Cartesian) symmetric. This entirely reflects the relation between the conventional semantics of non-dependent intuitionistic and linear type systems. The structure we obtain is that of a strict indexed symmetric monoidal[6] category with comprehension.

The Σ- and Π-types arise as left and right adjoints of substitution functors along projections in the base-category and the Id-types arise as left adjoints to substitution along diagonals, all satisfying Beck-Chevalley (and Frobenius) conditions, as is the case in the semantics for DTT. The !-types boil down to having a left adjoint to the comprehension (which can be made a functor), giving a linear-non-linear adjunction as in the conventional semantics for linear logic. Finally, additive connectives arise as compatible Cartesian and distributive co-Cartesian structures on the fibres, as would be expected from the semantics of linear logic.

3.1 Tautological models of ILDTT

First, we translate the structural core of our syntax to the tautological notion of model. We will later prove this to be equivalent to the more intuitive notion of categorical model we referred to above.

Definition (Tautological model of ILDTT). With a (tautological) model $\widetilde{\mathbb{T}}$ of ILDTT, we shall mean the following.

1. We have a set ICtxt, of which the elements will be interpreted as (dependent) contexts consisting of intuitionistic types. Then, for all $\Delta \in$ ICtxt, we have a set $\mathrm{LType}(\Delta)$ of linear types and a set $\mathrm{LCtxt}(\Delta)$ of linear contexts (multisets of linear types) in the context Δ. For each $\Delta, \Xi \in \mathrm{LCtxt}(\Delta)$, and each $A \in \mathrm{LType}(\Delta)$, we have a set $\mathrm{LTerm}(\Delta, \Xi, A)$ of (linear) terms of A in context $\Delta; \Xi$. On all these sets, judgemental equality is interpreted by equality of elements, taking into account α-conversion for terms. (The means that, if we construct a model from the syntax, we devide out judgemental equality on the syntactic objects.) This takes care of the -Eq rules, the rules expressing that judgemental equality is an equivalence relation, and the rules relating typing and judgemental equality.

2. C-Emp says that ICtxt has a distinguished element $\cdot$ and that $\mathrm{LType}(\cdot)$ has a distinguished element $\cdot$.

3. Int-C-Ext says that for all $\Delta \in$ ICtxt and $A \in \mathrm{LType}(\Delta)$, we can form a context $\Delta.A \in$ ICtxt, and that we have a function $\mathrm{LCtxt}(\Delta) \longrightarrow \mathrm{LCtxt}(\Delta.A)$, introducing fake dependencies of the linear types on A. Int-Exch says that for two types in the same context, the order in which we append them to a context does not matter.

4. Lin-C-Ext says that for all $\Xi \in \mathrm{LCtxt}(\Delta)$ and $A \in \mathrm{LType}(\Delta)$, we can form a context $\Xi.A \in \mathrm{LCtxt}(\Delta)$. Lin-Exch says that the order in which we do this does not matter.

5. Int-Var says that for a context $\Delta.A.\Delta'$, we have a term $\mathrm{der}_{\Delta,A,\Delta'} \in \mathrm{LTerm}(\Delta.A.\Delta', \cdot, A)$, that - this is implicitly present in the syntax - acts as a diagonal morphism through the substitution operations of Int-Ty-Subst and Int-Tm-Subst, equating the values of two variables of type A.

6. Lin-Var says that for a context Δ and a type $A \in \mathrm{LType}(\Delta)$, we have a term $\mathrm{id}_A \in \mathrm{LTerm}(\Delta, A, A)$, that - this is implicitly present in the syntax - acts as the identity for the substitution operation of Lin-Tm-Subst.

7. Int-Weak says that we have functions (abusing notation) $\mathrm{LType}(\Delta.\Delta') \overset{\mathrm{weak}_{\Delta,A,\Delta'}}{\longrightarrow} \mathrm{LType}(\Delta.A.\Delta')$ and $\mathrm{LTerm}(\Delta.\Delta', \Xi, B) \overset{\mathrm{weak}_{\Delta,A,\Delta'}}{\longrightarrow} \mathrm{LTerm}(\Delta.A.\Delta', \Xi, B)$. We think of this as projecting away a variable $x : A$ to introduce a fake dependency.

---

[6] As said before, we can easily obtain a sound and complete semantics for only the structural core, possibly without I- and $\otimes$-types, by considering strict indexed symmetric multicategories with comprehension.

8. Int-Ty-Subst says that for $B \in \mathrm{LType}(\Delta.A.\Delta')$ and $a \in \mathrm{LTerm}(\Delta, \cdot, A)$, we have a context $\Delta.\Delta'[a/x] \in$ ICtxt and a $B[a/x] \in \mathrm{LType}(\Delta.\Delta'[a/x])$.

9. Int-Tm-Subst says that for $b \in \mathrm{LTerm}(\Delta.A, \Delta', \Xi, B)$ and $a \in \mathrm{LTerm}(\Delta, \cdot, A)$, we have $b[a/x] \in \mathrm{LTerm}(\Delta.\Delta'[a/x], \Xi[a/x], B[a/x])$.

10. Lin-Tm-Subst says that for $b \in \mathrm{LTerm}(\Delta, \Xi.A, B)$ and $a \in \mathrm{LTerm}(\Delta, \Xi', A)$, we have $b[a/x] \in \mathrm{LTerm}(\Delta, \Xi.\Xi', B)$.

For these last three substitution operations it is implicit in the syntax that they are associative.

Finally, the remarks in Int-Var and Int-Weak about diagonals and projections in formal terms mean that the morphisms coming from these rules work together to form a generalised comonoid.

It is tautological that there is a one to one correspondence between theories $\mathbb{T}$ in ILDTT and models $\widetilde{\mathbb{T}}$ of this sort.

We now define what it means for the model to support various type formers.

Definition (Semantic I- and $\otimes$-types)**.** We say a model $\widetilde{\mathbb{T}}$ supports I-types, if for all $\Delta \in$ ICtxt, we have an $I \in \mathrm{LType}(\Delta)$ and $* \in \mathrm{LTerm}(\Delta, \cdot, I)$ and whenever $t \in \mathrm{LTerm}(\Delta, \Xi, I)$ and $a \in \mathrm{LTerm}(\Delta, \Xi, A)$, we have let t be $*$ in $a \in \mathrm{LTerm}(\Delta, \Xi.\Xi', A)$, such that let $*$ be $*$ in $a = a$ and let t be $*$ in $* = t$.

Similarly, we say it admits $\otimes$-types, if for all $A, B \in \mathrm{LType}(\Delta)$, we have a $A \otimes B \in \mathrm{LType}(\Delta)$, for all $a \in \mathrm{LTerm}(\Delta, \Xi, A), b \in \mathrm{LTerm}(\Delta, \Xi', B)$, we have $a \otimes b \in \mathrm{LType}(\Delta, \Xi.\Xi', A \otimes B)$, and if $t \in \mathrm{LTerm}(\Delta, \Xi, A \otimes B)$ and $c \in \mathrm{LTerm}(\Delta, \Xi'.A.B, C)$, we have let t be $x \otimes y$ in $c \in \mathrm{LTerm}(\Delta, \Xi.\Xi', C)$, such that let $a \otimes b$ be $x \otimes y$ in $c = c$ and let t be $x \otimes y$ in $x \otimes y = t$.

Note that this defines a function $\mathrm{LCtxt}(\Delta) \xrightarrow{\otimes} \mathrm{LType}$. The C-rule precisely says that from the point of view of the (terms of the) type theory this map is an injection, while the U-rule says it is a surjection[7]. We conclude that in presence of I- and $\otimes$-types, we can faithfully describe the type theory without mentioning linear contexts, replacing them by the linear type that is their $\otimes$-product.

We will henceforth assume that our type theory has I- and $\otimes$-types, as this simplifies the categorical semantics[8] and is appropriate for the examples we are interested in.

For the other type formers, one can give a similar, almost tautological, translation from the syntax into a tautological model. We trust this to the reader when we discuss the semantic equivalent of various type formers in the categorical semantics we present next.

3.2 Categorical Semantics of ILDTT

Strict Indexed Symmetric Monoidal Categories with Comprehension

We now introduce a notion of categorical model for which soundness and completeness results hold with respect to the syntax of ILDTT in presence of I- and $\otimes$-types[9]. This notion of model will prove to be particularly useful when thinking about various (extensional) type formers.

Definition 1. By a *strict indexed symmetric monoidal category with comprehension*, we will mean the following data.

1. A category $\mathcal{C}$ with a terminal object $\cdot$.

2. A strict indexed symmetric monoidal category $\mathcal{L}$ over $\mathcal{C}$, i.e. a contravariant functor $\mathcal{L}$ into the category SMCat of (small) symmetric monoidal categories and strong monoidal functors $\mathcal{C}^{op} \xrightarrow{\mathcal{L}}$ SMCat. We will also write $-\{f\} := \mathcal{L}(f)$ for the action of $\mathcal{L}$ on a morphism f of $\mathcal{C}$.

---

[7]The precise statement that we are alluding to here would be that the multicategory of linear contexts is equivalent to the (monoidal) multicategory of linear types. Really, $\otimes$ is only part of an equivalence of categories rather than an isomorphism, i.e. it is injective on objects up to isomorphism rather than on the nose.

[8]To be precise, it allows us to give a categorical semantics in terms of monoidal categories rather than multicategories.

[9]In case we are interested in the case without I- and $\otimes$-types, the semantics easily generalises to strict indexed symmetric multicategories with comprehension.

3. A *comprehension schema*, i.e. for each $\Delta \in \mathrm{ob}(\mathcal{C})$ and $A \in \mathrm{ob}(\mathcal{L}(\Delta))$ a representation for the functor

$$x \mapsto \mathcal{L}(\mathrm{dom}(x))(I, A\{x\}) : (\mathcal{C}/\Delta)^{op} \longrightarrow \mathrm{Set}.$$

We will write its representing object[10] $\Delta.A \overset{\mathbf{p}_{\Delta,A}}{\longrightarrow} \Delta \in \mathrm{ob}(\mathcal{C}/\Delta)$ and universal element $\mathbf{v}_{\Delta,A} \in \mathcal{L}(\Delta.A)(I, A\{\mathbf{p}_{\Delta,A}\})$. We will write $a \mapsto \langle f, a \rangle$ for the isomorphism $\mathcal{L}(\Delta')(I, A\{f\}) \cong \mathcal{C}/\Delta(f, \mathbf{p}_{\Delta,A})$.

Again, the comprehension schema means that the morphisms in our category of contexts $\mathcal{C}$, into a context built by adjoining types, arise as lists of closed linear terms. Here, there is the crucial identification with intuitionistic terms of linear terms without linear assumptions: they can be freely copied and discarded.

Remark 3. Note that this notion of model reduces to a standard notion of model for intuitionistic dependent type theory in the case that the monoidal structures on the fibre categories are Cartesian: a strict indexed Cartesian monoidal category with comprehension. Indeed, these are easily seen to be equivalent to, for instance, the split comprehension categories of [6] with terminal object and Cartesian products. However, it turns we have to impose the extra requirement of fullness on the comprehension category to get an exact match with the syntax. The corresponding condition in our framework is to ask for the comprehension functor to be full and faithful. See [6] for more discussion.

Proof. First note that a morphism M of indexed symmetric monoidal categories consists of lax monoidal functors in each context $\Delta \in \mathcal{C}$ such that

We define

$$M_\Delta(A \overset{a}{\longrightarrow} B) := \mathbf{p}_{\Delta,A} \xrightarrow{\langle \mathbf{p}_{\Delta,A}, a\{\mathbf{p}_{\Delta,A}\} \circ \mathbf{v}_{\Delta,A}\rangle} \mathbf{p}_{\Delta,B}.$$

Functoriality follows from the uniqueness property of $\langle \mathrm{id}_\Delta, a \rangle$.

We define the lax monoidal structure

$$\mathrm{id}_\Delta \xrightarrow{\qquad m_\Delta^I \qquad} M_\Delta(I) = \mathbf{p}_{\Delta,I}$$

$$\mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,B\{\mathbf{p}_{\Delta,A}\}} = M_\Delta(A) \times M_\Delta(B) \xrightarrow{m_\Delta^{A,B}} M_\Delta(A \otimes B) = \mathbf{p}_{\Delta,A\otimes B},$$

where $m_\Delta^I := \langle \mathrm{id}_\Delta, \mathrm{id}_I \rangle$ and $m_\Delta^{A,B} := \langle \mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,B\{\mathbf{p}_{\Delta,A}\}}, \mathbf{v}_{\Delta,A}\{\mathbf{p}_{\Delta.A,B\{\mathbf{p}_{\Delta,A}\}}\} \otimes \mathbf{v}_{\Delta.A,B\{\mathbf{p}_{\Delta,A}\}} \rangle$.

Finally, we verify that $\mathcal{I}(f)M_\Delta = M_{\Delta'}\mathcal{L}(f)$. This follows directly from the fact that the following square is a pullback square:

---

[10]Really, $\Delta.MA \overset{\mathbf{p}_{\Delta,MA}}{\longrightarrow} \Delta$ would be a better notation, where we think of $L \dashv M$ as an adjunction inducing !, but it would be very verbose.

[11]Here, we use the axiom of choice to make a choice of pullback and make $\mathcal{I}$ really into an indexed category (or cloven fibration). Alternatively, we can avoid the axiom of choice and treat it as a more general fibration.

where $\mathbf{q}_{f,A} \coloneqq \langle f\mathbf{p}_{\Delta',A\{f\}}, \mathbf{v}_{\Delta',A\{f\}}\rangle$. We leave this verification to the reader as an exercise. Alternatively, a proof for this fact in DTT, that will transfer to our setting in its entirety, can be found in [5]. $\quad\square$

Remark 4. Note that $\mathcal{I}$ is a display map category (or, less specifically, a full comprehension category). Hence, it is a model of intuitionistic type theory. We will see that, in many ways, we can regard it as the intuitionistic content of $\mathcal{L}$.

Remark 5. We will see that this functor will give us a unique candidate for !-types: $! \coloneqq LM$, where $L \dashv M$. We conclude that, in ILDTT, the !-modality is uniquely determined by the indexing. This is worth noting, because, in propositional linear type theory, we might have many different candidates for !-types.

Moreover, it explains why we don't demand M to be fully faithful in the case of linear types. Indeed, although we have a map $\mathcal{L}(\Delta)(A,B) \xrightarrow{M_\Delta} \mathcal{I}(\Delta)(\mathbf{p}_{\Delta,A}, \mathbf{p}_{\Delta,B}) \cong \mathcal{L}(\Delta.A)(I, B\{\mathbf{p}_{\Delta,A}\})$, this is not generally an isomorphism. In fact, in presence of !-types, we will see that the right hand side is precisely isomorphic to $\mathcal{L}(\Delta)(!A, B)$ and the map is precomposition with dereliction.

Next, we prove that we have a sound interpretation of ILDTT in such categories.

Theorem 6 (Soundness). *A strict indexed symmetric monoidal category with comprehension $(\mathcal{C}, \mathcal{L}, \mathbf{p}, \mathbf{v})$ defines a model $\widetilde{\mathbb{T}}^{(\mathcal{C},\mathcal{L},\mathbf{p},\mathbf{v})}$ of ILDTT with I- and $\otimes$-types.*

Proof. We define

1. ICtxt $\coloneqq \mathrm{ob}(\mathcal{C})$
   LType$(\Delta) \coloneqq \mathrm{ob}(\mathcal{L}(\Delta))$
   LCtxt$(\Delta) \coloneqq \mathrm{free-comm.-monoid}(\mathrm{LType}(\Delta))$ (where we will write 0 and + for the operations)
   LTerm$(\Delta, \Xi, A) \coloneqq \mathcal{L}(\Delta)(\otimes\Xi, A)$

2. C-Emp: $\cdot_{\mathrm{ICtxt}} \coloneqq \cdot_{\mathcal{C}}$ and $\cdot_{\mathrm{LCtxt}(\Delta)} \coloneqq 0_{\mathrm{LCtxt}(\Delta)}$.

3. Int-C-Ext: $\Delta._{\mathrm{ICtxt}} A \coloneqq \Delta._{\mathcal{C}} A$ and LType$(\Delta) \longrightarrow$ LType$(\Delta.A) \coloneqq -\{\mathbf{p}_{\Delta,A}\}$ inducing the obvious function LCtxt$(\Delta) \longrightarrow$ LCtxt$(\Delta.A)$. We have seen how

$$
\begin{array}{ccc}
\Delta.A.B\{\mathbf{p}_{\Delta,A}\} & \xrightarrow{\;\mathbf{q}_{\mathbf{p}_{\Delta,A},B}\;} & \Delta.B \\
{\scriptstyle \mathbf{p}_{\Delta.A,B\{\mathbf{p}_{\Delta,A}\}}}\Big\downarrow & & \Big\downarrow{\scriptstyle \mathbf{p}_{\Delta,B}} \\
\Delta.A & \xrightarrow[\;\mathbf{p}_{\Delta,A}\;]{} & \Delta
\end{array}
$$

is a product in $\mathcal{C}/\Delta$ which interprets the double context extension $\Delta.A.B$ where $A, B \in$ LType(Δ). Being a Cartesian monoidal structure, this is, in particular, symmetric, so validates Int-Exch.

4. Lin-C-Ext: $\Xi.A \coloneqq \Xi + A$

5. Int-Var: $\mathrm{der}_{\Delta,A,\Delta'} \in$ LTerm$(\Delta.A.\Delta', \cdot, A)$ is defined as

$$
\mathbf{v}_{\Delta,A}\{\mathbf{p}_{\Delta.A,\Delta'}\} : I \longrightarrow A\{\mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,\Delta'}\} \in \mathcal{L}(\Delta.A.\Delta')
$$

Note that $\mathrm{der}_{\Delta,A,\Delta'}$ defines a morphism

$\Delta.A.\Delta' \xrightarrow{\mathrm{diag}_{\Delta,A,\Delta'}} \Delta.A.\Delta'.A\{\mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,\Delta'}\} \coloneqq \langle \mathrm{id}_{\Delta.A.\Delta'}, \mathrm{der}_{\Delta,A,\Delta'}\rangle$.
We will later show that this in fact behaves as a diagonal morphism on A.

6. $\mathrm{id}_A \in$ LTerm(Δ, A, A) is taken to be $\mathrm{id}_A \in \mathcal{L}(\Delta)(A, A)$. Note that this is indeed the neutral element for our semantic linear term substitution operation that we will define shortly.

19

7. The required morphisms in Int-Weak are interpreted as follows. Suppose we are given $A, \Delta' \in \mathrm{ob}(\mathcal{L}(\Delta))$. We will define a functor

$$\mathcal{L}(\Delta.\Delta'\{\mathbf{p}_{\Delta,A}\}) \xrightarrow{\mathcal{L}(\langle f,a \rangle)} \mathcal{L}(\Delta.A.\Delta'),$$

where f and a are defined as follows.

$$\Delta.A.\Delta'\{\mathbf{p}_{\Delta,A}\} \xrightarrow{f := \mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,\Delta'\{\mathbf{p}_{\Delta,A}\}}} \Delta$$

and

$$I \xrightarrow{a = \mathbf{v}_{\Delta,A.\Delta'\{\mathbf{p}_{\Delta,A}\}}} \Delta'\{f\} = \Delta'\{\mathbf{p}_{\Delta,A.\Delta'\{\mathbf{p}_{\Delta,A}\}}\} \in \mathcal{L}(\Delta.A.\Delta'\{\mathbf{p}_{\Delta,A}\}).$$

8.&9. Int-Ty-Subst and Int-Tm-Subst, along a term $\Delta; \cdot \vdash a : A$, are interpreted by the functors $\mathcal{L}(\langle \mathrm{id}_\Delta, a \rangle) = -\{\langle \mathrm{id}_\Delta, a \rangle\}$. Indeed, let $B \in \mathcal{L}(\Delta.A.\Delta')$ and $a \in \mathcal{L}(\Delta)(I, A)$. Then, we define the context $\Delta.\Delta'[a/x]$ as $\Delta.(\Delta'\{\langle \mathrm{id}_\Delta, a \rangle\})$ and the type $B[a/x]$ as $B\{\langle f, a' \rangle\}$, where

$$\Delta.\Delta'\{\langle \mathrm{id}_\Delta, a \rangle\}.I \xrightarrow{\langle f,a' \rangle} \Delta.A.\Delta'$$
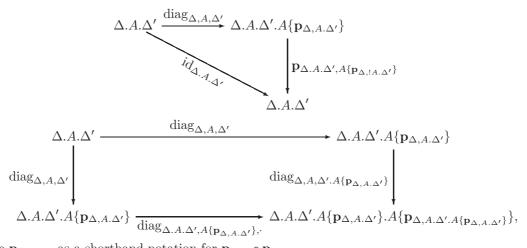
is defined from

and

$$I \xrightarrow{a' := \mathbf{v}_{\Delta,\Delta'\{\langle \mathrm{id}_\Delta, a \rangle\}}} \Delta'\{f\} = (\Delta'\{\langle \mathrm{id}_\Delta, a \rangle\})\{\mathbf{p}_{\Delta,\Delta'\{\langle \mathrm{id}_\Delta, a \rangle\}}\}.$$

10. Lin-Tm-Subst is interpreted by composition in $\mathcal{L}(\Delta)$. To be precise, given $b \in \mathcal{L}(\Delta)(\otimes \Xi \otimes A, B)$ and $a \in \mathcal{L}(\Delta)(\otimes \Xi', A)$, we define $b[a/x] \in \mathcal{L}(\Delta)(\otimes \Xi \otimes \otimes \Xi', B)$ as $b \circ \mathrm{id}_{\otimes \Xi} \otimes a$.

Note that all our substitution rules are interpreted by functors and are therefore clearly associative.

The fact that Int-Var and Int-Weak define compatible (generalised) diagonals and projections is reflected in the fact that diag and $\mathbf{p}$ obey generalised comonoid laws:

where we use $\mathbf{p}_{\Delta,A.\Delta'}$ as a shorthand notation for $\mathbf{p}_{\Delta,A} \circ \mathbf{p}_{\Delta.A,\Delta'}$.

Finally, the model clearly supports I- and $\otimes$-types. We interpret $I \in \mathrm{LType}(\Delta)$ as the unit object in $\mathcal{L}(\Delta)$ while its term $*$ is interpreted as the identity morphism. Similarly, we interpret $\otimes$ by the monoidal product on the fibres: $* := \mathrm{id}_I \in \mathcal{L}(\Delta)$, let t be $*$ in $a := t \otimes a$, $a \otimes b$ is defined as the tensor product of morphisms in $\mathcal{L}(\Delta)$, and let t be $x \otimes y$ in $c := c \circ (\mathrm{id}_{\Xi'} \otimes t)$ (ignoring associatiators and unitors, here). The C- and U-rules are immediate. □

In fact, the converse is also true: we can build a category of this sort from the syntax of ILDTT.

Theorem 7 (co-Soundness). *A tautological model $\widetilde{\mathbb{T}}$ of ILDTT with I and $\otimes$-types defines a strict indexed symmetric monoidal category with comprehension $(\mathcal{C}^{\mathbb{T}}, \mathcal{L}^{\mathbb{T}}, \mathbf{p}^{\mathbb{T}}, \mathbf{v}^{\mathbb{T}})$.*

Proof. The main technical difficulty in this proof will be that our category of contexts has context morphisms as morphisms (corresponding to lists of terms of the type theory) while the type theory only talks about individual terms. This exact difficulty is also encountered when proving completeness of the categories with families semantics for ordinary DTT. It is sometimes fixed by (conservatively) extending the the type theory to also talk about context morphisms explicitly. See e.g. [19].

1. We define $\mathrm{ob}(\mathcal{C}^{\mathbb{T}}) \coloneqq \mathrm{ICtxt}$, modulo α-equivalence, and write $\Delta.A \coloneqq \Delta, x : A$. The designated object will be $\cdot$ (from C-Emp), which will automatically become a terminal object because of our definition of a morphism of $\mathcal{C}^{\mathbb{T}}$ (context morphism). Indeed, we define morphisms in $\mathcal{C}^{\mathbb{T}}$, as follows, by induction.

   Start out defining $\mathcal{C}^{\mathbb{T}}(\Delta', \cdot) \coloneqq \{\langle\rangle\}$ and for $\Delta \in \mathrm{ICtxt}$ that are not of the form $\Delta''.A$, define $\mathcal{C}^{\mathbb{T}}(\Delta', \Delta) = \{\mathrm{id}_\Delta\}$ if $\Delta' = \Delta$ and $\mathcal{C}^{\mathbb{T}}(\Delta', \Delta) = \varnothing$ otherwise.

   Then, by induction on the length n of $\Delta = x_1 : A_1, \ldots, x_n : A_n$, we define

   $$\mathcal{C}^{\mathbb{T}}(\Delta', \Delta.A_{n+1}) \coloneqq \Sigma_{f \in \mathcal{C}^{\mathbb{T}}(\Delta', \Delta)} \mathrm{LTerm}(\Delta', \cdot, A_{n+1}[f/x]),$$

   where $A_{n+1}[f/x]$ is defined, using Int-Ty-Subst, to be the (syntactic operation of) parallel substitution (see [5], section 2.4) of the list $f_1, \ldots, f_n$ of linear terms $\Delta'; \cdot \vdash f_i : A_i[f_1/x_1, \ldots, f_{i-1}/x_{i-1}]$ that f is made up out of, for the variables $x_1, \ldots, x_n$ in Δ.

   Note that, in particular, according to Int-Var, $\mathrm{LTerm}(A_1.\ldots.A_n., \cdot, A_i)$ contains a term $\mathrm{der}_{A_1.\ldots.A_{i-1}, A_i, A_{i+1}.\ldots.A_n}$, which allows us to define, inductively,

   $$\mathbf{p}_{A_1.\ldots.A_n, \cdot} \coloneqq \langle\rangle \in \mathcal{C}^{\mathbb{T}}(A_1.\ldots.A_n, \cdot)$$

   $$\mathbf{p}_{A_1.\ldots.A_n, A_1.\ldots.A_i} \coloneqq \mathbf{p}_{A_1.\ldots.A_n, A_1.\ldots.A_{i-1}}, \mathrm{der}_{A_1.\ldots.A_{i-1}, A_i, A_{i+1}.\ldots.A_n} \in \mathcal{C}^{\mathbb{T}}(A_1.\ldots.A_n, A_1.\ldots.A_i)$$

   In particular, we define identities in $\mathcal{C}^{\mathbb{T}}$ from these: $\mathrm{id}_{A_1.\ldots.A_n} \coloneqq \mathbf{p}_{A_1.\ldots.A_n, A_1.\ldots.A_n}$. We will also use these 'projections' in 3. to define the comprehension schema.

   We define composition in $\mathcal{C}^{\mathbb{T}}$ by induction. Let $B_1.\ldots.B_m = \Delta' \xrightarrow{f = f_1, \ldots, f_n} \Delta = A_1.\ldots.A_n$ and $\Delta'' \xrightarrow{g = g_1, \ldots, g_m} \Delta'$. Then, we define, $(f_1, \ldots, f_{n-1}, f_n) \circ g \coloneqq (f_1, \ldots, f_{n-1}) \circ g, f_n[g/x]$, where $f_n[g/x]$ denotes the parallel substitution of $g = g_1, \ldots, g_m$ for the free variables $x_1, \ldots, x_m$ in f_n, using Int-Tm-Subst. Note that associativity of composition comes from the associativity of substitution that is implicit in the syntax while the identity morphism we defined clearly acts as a neutral element for our composition.

2. Define $\mathrm{ob}(\mathcal{L}^{\mathbb{T}}(\Delta)) \coloneqq \mathrm{LCtxt}(\Delta)$ and $\mathcal{L}^{\mathbb{T}}(\Delta)(\Xi, \Xi') \coloneqq \mathrm{LTerm}(\Delta, \Xi, \otimes \Xi')$. Composition is defined through Lin-Tm-Subst and $\otimes$-E. Identities are given by Lin-Var. The monoidal unit is given by $\cdot \in \mathrm{LCtxt}(\Delta)$, while the monoidal product $\otimes$ on objects is given by context concatenation. The monoidal product $\otimes$ on morphisms is given by $\otimes$-I. Note that the associators and unitors follow from the associative and unital laws for the commutative monoid of contexts together with $\otimes$-C and $\otimes$-U. (Note that the rules for $\otimes$ give us an isomorphism between an arbitrary context Ξ and the one-type-context $\otimes \Xi$, while the rules for I do the same for $\cdot$ and I.)

   We define $\mathcal{L}^{\mathbb{T}}(f)$ on objects by parallel substitution in each type in a linear context, via Int-Ty-Subst, and on morphisms by parallel substitution, via Int-Tm-Subst. Note that functoriality is given by implicit properties of the syntax like associativity of substitution. Note that this defines a strong symmetric monoidal functor. We conclude that $\mathcal{L}^{\mathbb{T}}$ is a functor $\mathcal{C}^{\mathbb{T}op} \longrightarrow \mathrm{SMCat}$.

3. We define following comprehension schema on $\mathcal{L}^{\mathbb{T}}$. Suppose $\Delta \in \mathcal{C}^{\mathbb{T}}$ and $A \in \mathcal{L}^{\mathbb{T}}(\Delta)$.

   Define $\Delta.A \xrightarrow{\mathbf{p}^{\mathbb{T}}_{\Delta,A}} \Delta$ as $\mathbf{p}_{\Delta,A}$ from 1. and $I \xrightarrow{\mathbf{v}^{\mathbb{T}}_{\Delta,A}} A\{\mathbf{p}_{\Delta,A}\}$ (through Int-Var) as $\mathrm{der}_A \in \mathrm{LTerm}(\Delta.A, \cdot, A) = \mathcal{L}^{\mathbb{T}}(\Delta.A)(I, A\{\mathbf{p}^{\mathbb{T}}_{\Delta,A}\})$.

Suppose we are given $\Delta' \xrightarrow{f} \Delta$ and $a \in \mathcal{L}^{\mathbb{T}}(\Delta')(I, A\{f\}) = \mathrm{LTerm}(\Delta', \cdot, A[f/c])$. Then, by definition of the morphisms in $\mathcal{C}^{\mathbb{T}}$, there is a unique morphism
$\langle f, a \rangle := f, a \in \mathcal{C}^{\mathbb{T}}(\Delta', \Delta.A) := \Sigma_{f \in \mathcal{C}^{\mathbb{T}}(\Delta', \Delta)} \mathrm{LTerm}(\Delta', \cdot, A[f/x])$ such that $\mathbf{p}^{\mathbb{T}}_{\Delta, A} \circ \langle f, a \rangle = f$ and $\mathbf{v}^{\mathbb{T}}_{\Delta, A}\{\langle f, a \rangle\} = a$. The uniqueness follows from the fact that $\mathbf{p}^{\mathbb{T}}_{\Delta, A} \circ -$ and $\mathbf{v}^{\mathbb{T}}_{\Delta, A}\{-\}$ are the two (dependent) projections of the Σ-type (in Set) that defines this homset.

$\square$

Theorem 8 (Completeness)**.** *The construction described in 'co-Soundness' followed by the one described in 'Soundness' is the identity (up to categorical equivalence[12]): i.e. strict indexed symmetric monoidal categories with comprehension provide a complete semantics for ILDTT with I- and $\otimes$-types[13].*

Proof. This is a trivial exercise. $\square$

Theorem 9 (Failure of co-Completeness)**.** *The construction described in 'Soundness' followed by the one described in 'co-Soundness' is not equivalent to the identity: i.e. co-Completeness fails (as for the categories with families semantics for DTT).*

Proof. Indeed, if we start with a strict indexed symmetric monoidal category with comprehension, construct the corresponding tautological model $\widetilde{\mathbb{T}}$ and then construct its syntactic category, we effectively have thrown away all the non-trivial morphisms into objects that are not of the form $\Delta.A$.

Of course, we can easily obtain a co-complete model theory by putting this extra restriction on our models. Alternatively - this may be nicer from a categorical point of view -, we can take the obvious (see e.g. [19]) conservative extension of our syntax by also talking about context morphisms (corresponding to morphisms in our base category). In that case, we would obtain an actual internal language for strict indexed symmetric monoidal categories with comprehension. This also has the advantage that we can easily obtain an internal language for strict indexed monoidal categories by dropping the axioms Int-C-Ext, Int-C-Ext-Eq, and Int-Var, which correspond to the comprehension schema. We have not chosen this route as it would mean that the syntax does not fit as well with what has been considered so far in the syntactic tradition. $\square$

Corollary 1 (Relation with DTT and ILTT)**.** *A model $(\mathcal{C}, \mathcal{L}, \mathbf{p}, \mathbf{v})$ of ILDTT with I- and $\otimes$-types defines a model $\mathcal{I}$ of DTT, that should be thought of the intuitionistic content of the linear type theory. This will become even more clear through our treatment of $!$-types and in the examples we treat.*

Moreover, it clearly defines a model of ILTT with I- and $\otimes$-types (i.e. a symmetric monoidal category) in every context.

Conversely, it is easily seen that every syntactic model[14] of DTT can be obtained this way, up to equivalence, from a syntactic model of ILDTT (we take the same constants and axioms: effectively the same theory but in a system without contraction and weakening) and that every model of ILTT can be embedded in a model of ILDTT. (As we will see later, we can cofreely add type dependency on Set*.)*

Semantic Type Formers

Theorem 10 (Semantic type formers)**.** *For the other type formers, we have the following. A model of ILDTT with I- and $\otimes$-types (a strict indexed symmetric monoidal category with comprehension)...*

1. *...supports Σ-types iff all the pullback functors $\mathcal{L}(\mathbf{p}_{\Delta, A})$ have left adjoints $\Sigma_{!A}$ that satisfy the*

---

[12]The correct formal statement here would be that co-soundness followed by soundness (both of which define 2-functors between the 2-category of tautological models of ILDTT and the 2-category of strict indexed symmetric monoidal categories with comprehension) is 2-equivalent to the identity.

[13]It is easy to see that, similarly, indexed symmetric multicategories with comprehension form a complete semantics for ILDTT, possibly without I- and $\otimes$-types.

[14]i.e. a model where we do not have any non-trivial morphisms into contexts that are not built from $\cdot$ by appending types.

Beck-Chevalley condition[15][16] for pullback squares in $\mathcal{C}$ of the following form,[17].

$$\begin{array}{ccc}
\Delta'.B\{f\} & \xrightarrow{\ \mathbf{q}_{f,B}\ } & \Delta.B \\
{\scriptstyle \mathbf{p}_{\Delta',B\{f\}}}\downarrow & (*) & \downarrow{\scriptstyle \mathbf{p}_{\Delta,B}} \\
\Delta' & \xrightarrow[\ f\ ]{} & \Delta,
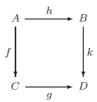\end{array}$$

and that satisfy Frobenius reciprocity[18] in the sense that the canonical morphism

$$\Sigma_{!A}(\Xi'\{\mathbf{p}_{\Delta,A}\}\otimes B)\longrightarrow \Xi'\otimes\Sigma_{!A}B$$

is an isomorphism , for all $\Xi'\in\mathcal{L}(\Delta)$, $B\in\mathcal{L}(\Delta.A)$.

2. ...supports Π-types iff all the pullback functors $\mathcal{L}(\mathbf{p}_{\Delta,A})$ have right adjoints $\Pi_{!A}$ that satisfy the dual Beck-Chevalley condition for pullbacks of the form $(*)$.

3. ...supports $\multimap$-types iff $\mathcal{L}$ factors over SMCCat.

4. ...supports $\top$-types and $\&$-types iff $\mathcal{L}$ factors over the category SMCCat of Cartesian categories with a symmetric monoidal structure and their homomorphisms.

5. ...supports 0-types and $\oplus$-types iff $\mathcal{L}$ factors over the category dSMcCCat of co-Cartesian categories with a distributive[19] symmetric monoidal structure and their homomorphisms.

6. ...that supports $\multimap$-types[20], supports !-types iff all the comprehension functors $\mathcal{L}(\Delta)\xrightarrow{M_\Delta}\mathcal{I}(\Delta)$ have a left adjoint $\mathcal{I}(\Delta)\xrightarrow{L_\Delta}\mathcal{L}(\Delta)$ in the 2-category SMCat of symmetric monoidal categories, lax symmetric monoidal functors, and monoidal natural transformations[21] and (compatibility with

---

[15]Remember that the Beck-Chevalley condition for a pullback square

$$\begin{array}{ccc}
A & \xrightarrow{\ h\ } & B \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle k} \\
C & \xrightarrow[\ g\ ]{} & D
\end{array}$$

corresponds to the statement that the obvious morphism (from commuting of pullback square, unit, and counit) $f_!h^*\longrightarrow f_!h^*k^*k_!\xrightarrow{\cong} f_!f^*g^*k_!\longrightarrow g^*k_!$ is an isomorphism, where we write f^* for $\mathcal{L}(f)$ and $f_!$ for its left adjoint. In case f^* has a right adjoint (as for the Π-type), f_*, we mean by the dual Beck-Chevalley condition that the obvious morphism $g^*k_*\longrightarrow g^*k_*h_*h^*\xrightarrow{\cong} g^*g_*f_*h^*\longrightarrow f_*h^*$ is an iso.

[16]The reader is encouraged to think of this condition as the equivalent for Σ- and Π-types of the condition on the subtitution functors preserving the appropriate categorical structure for other type formers. It says that, in a sense, Σ- and Π-types are preserved under substitution.

[17]Recall that $\mathbf{q}_{f,B}:=\langle f\circ\mathbf{p}_{\Delta',B\{f\}},\mathbf{v}_{\Delta',B\{f\}}\rangle$ and that this square is indeed a pullback.

[18]Frobenius reciprocity expresses compatibility of Σ and $\otimes$, which is reasonable if we want a reading of Σ as a generalisation of $\otimes$. If one wants to drop Frobenius reciprocity in the semantics, it is easy to see that the equivalent in the syntax is setting $\Xi'\equiv\cdot$ in the Σ-E-rule.

[19]Note that in the light of theorem 4, the demand of distributivity here is essentially the same phenomenon as the demand of Frobenius reciprocity for Σ-types.

[20]Actually, we only need this for the 'if'. The 'only if' always holds. To make the 'if' work, as well, in absence of $\multimap$-types, we have to restrict !-E to the case where $\Xi'\equiv\cdot$.

[21]i.e. a symmetric lax monoidal left adjoint functor L_Δ such that an inverse for its lax structure is given by the oplax structure on L_Δ coming from the lax structure on M_Δ. Put differently, L_Δ is a left adjoint functor to M_Δ and is a strong monoidal functor in a way that is compatible with the lax structure on M_Δ.

substitution) for all $\Delta' \xrightarrow{f} \Delta \in \mathcal{C}$ which makes L_- into a morphism of indexed categories.:

$$\begin{array}{ccc}
\mathcal{L}(\Delta) & \xrightarrow{\ \ \mathcal{L}(f)\ \ } & \mathcal{L}(\Delta') \\
\Big\uparrow{\scriptstyle L_\Delta} & & \Big\uparrow{\scriptstyle L_{\Delta'}} \\
\mathcal{I}(\Delta) & \xrightarrow[\mathcal{I}(f) = f^* \ (=\text{pullback along } f)]{} & \mathcal{I}(\Delta').
\end{array}$$

Then the linear exponential comonad $!_\Delta := L_\Delta \circ M_\Delta : \mathcal{L}(\Delta) \longrightarrow \mathcal{L}(\Delta)$ will be our interpretation of the comodality $!$ in the context Δ.

7. *... that supports $\multimap$-types, supports Id-types iff for all $A \in \mathrm{ob}\,\mathcal{L}(\Delta)$, we have left adjoints $\mathrm{Id}_{!A} \dashv -\{\mathrm{diag}_{\Delta,A}\}$ that satisfy a Beck-Chevalley condition: $\mathrm{Id}_{!A\{f\}} \circ \mathcal{L}(\mathbf{q}_{f,A}) \longrightarrow \mathcal{L}(\mathbf{q}_{\mathbf{q}_{f,A},A\{\mathbf{p}_{\Delta,A}\}}) \circ \mathrm{Id}_{!A}$ is an iso.*
   Here, $\Delta.A \xrightarrow{\mathrm{diag}_{\Delta,A} := \langle \mathrm{id}_{\Delta.A}, \mathbf{v}_{\Delta,A}\rangle} \Delta.A.A\{\mathbf{p}_{\Delta,A}\}$.

Proof. 1. Assume our model supports Σ-types. We will show the claimed adjunction. The morphism from left to right is provided by Σ-I. The morphism from right to left is provided by Σ-E. Σ-C and Σ-U say exactly that these are mutually inverse. Naturality corresponds to the compatibility of Σ-I and Σ-E with substitution.

$$c' \longmapsto c'\{\mathbf{p}_{\Delta,A}\} \circ \langle \mathrm{diag}_{\Delta,A,\cdot}, \mathrm{id}_B\rangle$$

$$\mathcal{L}(\Delta)(\Sigma_{!A}B, C) \underset{\cong}{\overset{}{\rightleftarrows}} \mathcal{L}(\Delta.A)(B, C\{\mathbf{p}_{\Delta,A}\})$$

$$\text{let } z \text{ be } !x \otimes y \text{ in } c \longleftarrow\!\!\shortmid\ c$$

We show how the morphism from left to right arises from Σ-I.

$$\cfrac{
\cfrac{\Delta, x : A; \cdot \vdash x : A}{} \text{Int-Var} \quad \cfrac{\cfrac{\Delta, x : A; w : B \vdash w : B}{\Delta, x : A; w : B \vdash !x \otimes w : \Sigma_{!x:!A}B} \text{Lin-Var}}{\Sigma\text{-I}} \quad \cfrac{\cfrac{\Delta; z : \Sigma_{!x:!A}B \vdash c' : C}{\Delta, x : A; z : \Sigma_{!x:!A}B \vdash c' : C} \text{Int-Weak}}{}
}{\Delta, x : A; w : B \vdash c'[!x \otimes w / z] : C} \text{Lin-Tm-Subst}$$

We show how the morphism from right to left is exactly Σ-E (with $\Xi' \equiv \cdot$, $\Xi \equiv z : \Sigma_{!x:!A}B$, $t \equiv z$).

$$\cfrac{\Delta; \cdot \vdash C \text{ type} \quad \cfrac{}{\Delta; z : \Sigma_{!x:!A}B \vdash z : \Sigma_{!x:!A}B} \text{Lin-Var} \quad \Delta, x : A; y : B \vdash c : C}{\Delta; z : \Sigma_{!x:!A}B \vdash \text{let } z \text{ be } !x \otimes y \text{ in } c : C} \Sigma\text{-E}$$

We show how Frobenius reciprocity can be proved in our type system (particularly relying on the form of the Σ-E-rule[22]).

Lemma 1 (Frobenius reciprocity)**.** *The canonical morphism*

$$\Sigma_{!A}(\Xi'\{\mathbf{p}_{\Delta,A}\} \otimes B) \xrightarrow{\ f\ } \Xi' \otimes \Sigma_{!A}B$$

is an isomorphism, for all $\Xi' \in \mathcal{L}(\Delta)$, $B \in \mathcal{L}(\Delta.A)$.

Proof. We first show how to construct the morphism f we mean.

---

[22]To be precise, we will see Frobenius reciprocity is validated because we allow dependency on Ξ' in the Σ-E-rule. Conversely, it is easy to see we can prove Frobenius reciprocity holds in our model if we have (semantic) $\multimap$-types, as this allows us to remove the dependency on Ξ' in Σ-E.

$$\cfrac{\Delta;x':\Sigma_{!x:!A}(\Xi'\otimes B)\vdash x':\Sigma_{!x:!A}(\Xi'\otimes B)\ \text{Lin-Var} \qquad \cfrac{\cfrac{\Delta,x:A;z:\Xi'\vdash z:\Xi'\ \text{Lin-Var} \qquad \cfrac{\Delta,x:A;\cdot\vdash x:A\ \text{Int-Var} \qquad \Delta;y:B\vdash y:B\ \text{Lin-Var}}{\Delta,x:A;y:B\vdash !x\otimes y:\Sigma_{!x:!A}B}\ \Sigma\text{-I}}{\Delta,x:A;z:\Xi',y:B\vdash z\otimes !x\otimes y:\Xi'\otimes\Sigma_{!x:!A}B}\ \otimes\text{-I}}{\Delta,x:A;w:\Xi'\otimes B\vdash \text{let } w \text{ be } z\otimes y \text{ in } z\otimes !x\otimes y:\Xi'\otimes\Sigma_{!x:!A}B}\ \otimes\text{-E}}{\Delta;x':\Sigma_{!x:!A}(\Xi'\otimes B)\vdash f:\Xi'\otimes\Sigma_{!x:!A}B}\ \Sigma\text{-E}$$

We now construct its inverse. Call it g.

$$\cfrac{\cfrac{\Delta;y_2:\Sigma_{!x:!A}B\vdash y_2:\Sigma_{!x:!A}B\ \text{Lin-Var} \qquad \cfrac{\cfrac{\Delta;y_1:\Xi'\vdash y_1:\Xi'\ \text{Lin-Var} \qquad \cfrac{\Delta,x:A;\cdot\vdash x:A\ \text{Int-Var} \qquad \Delta;y:B\vdash y:B\ \text{Lin-Var}}{\Delta,x:A;y_1:\Xi',y:B\vdash y_1\otimes y:\Xi'\otimes B}\ \otimes\text{-I}}{\Delta,x:A;y_1:\Xi',y:B\vdash !x\otimes y_1\otimes y:\Sigma_{!x:!A}(\Xi'\otimes B)}\ \Sigma\text{-I}}{\Delta,x:A;y_1:\Xi',y_2:\Sigma_{!x:!A}B\vdash \text{let } y_2 \text{ be } !x\otimes y \text{ in } !x\otimes y_1\otimes y:\Sigma_{!x:!A}(\Xi'\otimes B)}\ \Sigma\text{-E}}{\Delta;y':\Xi'\otimes\Sigma_{!x:!A}B\vdash g:\Sigma_{!x:!A}(\Xi'\otimes B)}\ \otimes\text{-E}$$

We leave it to the reader to verify that these morphisms are mutually inverse in the sense that

$$\Delta;x':\Sigma_{!x:!A}(\Xi'\otimes B)\vdash g[f/y']\equiv x':\Sigma_{!x:!A}(\Xi'\otimes B) \quad\text{and}\quad \Delta;y':\Xi'\otimes\Sigma_{!x:!A}B\vdash f[g/x']\equiv y':\otimes\Sigma_{!x:!A}B.$$

$\square$

For the converse, we show how to obtain Σ-I from our morphism from left to right:

$$\cfrac{\cfrac{\cfrac{\Delta;z:\Sigma_{!x:!A}B\vdash z:\Sigma_{!x:!A}B\ \text{Lin-Var}}{\Delta,x:A;w:B\vdash !x\otimes w:\Sigma_{!x:!A}B}\ \text{"left to right"} \qquad \Delta;\cdot\vdash a:A}{\Delta;w:B\vdash !a\otimes w:\Sigma_{!x:!A}B}\ \text{Int-Tm-Subst} \qquad \Delta;\Xi\vdash b:B[a/x]}{\Delta;\Xi\vdash !a\otimes b:\Sigma_{!x:!A}B}\ \text{Lin-Tm-Subst}$$

We show how to obtain Σ-E from our morphism from right to left, using Frobenius reciprocity. In particular, note that we don't need Frobius reciprocity.

$$\cfrac{\cfrac{\cfrac{\cfrac{\Delta;\cdot\vdash C\ \text{type} \qquad \cfrac{\Delta,x:A;y:\Xi',B\vdash c:C}{\Delta,x:A;y:\Xi'\otimes B\vdash c:C}\ \otimes\text{-E}}{\Delta;z:\Sigma_{!x:!A}(\Xi'\otimes B)\vdash \text{let } z \text{ be } !x\otimes y \text{ in } c:C}\ \text{"right to left"}}{\Delta;z:(\Xi'\otimes\Sigma_{!x:!A}B)\vdash \text{let } z \text{ be } !x\otimes y \text{ in } c:C}\ \text{Frobenius reciprocity}}{\Delta;z_1:\Xi',z_2:\Sigma_{!x:!A}B\vdash \text{let } z_1\otimes z_2 \text{ be } !x\otimes y \text{ in } c:C}\ \text{Lin-Tm-Subst},\otimes\text{-I},2\times\text{Lin-Var} \qquad \Delta;\Xi\vdash t:\Sigma_{!x:!A}B}{\Delta;z_1:\Xi',\Xi\vdash (\text{let } z_1\otimes z_2 \text{ be } !x\otimes y \text{ in } c)[t/z_2]:C}\ \text{Lin-Tm-Subst}$$

As usual, the Beck-Chevalley condition says precisely that Σ-types commute with substitution, as dictated by the type theory.

2. Assume our model supports Π-types. We will show the claimed adjunction. The morphism from left to right is provided by Π-I - in fact, it is exactly the I-rule - and the one from right to left by Π-E. Π-C and Π-U say exactly that these are mutually inverse. Naturality corresponds to the compatibility of Π-I and Π-E with substitution.

$$b \longmapsto \lambda_{!x:!A}b$$

$$\mathcal{L}(\Delta.A)(\Xi\{\mathbf{p}_{\Delta.A}\},B) \underset{\cong}{\overset{\longrightarrow}{\longleftarrow}} \mathcal{L}(\Delta)(\Xi,\Pi_{!x:!A}B)$$

$$f(!x) \longleftarrow\!\!\mid f.$$

We show how we obtain the definition of $f(!x)$ from Π-E.

$$\cfrac{\Delta,x:A;\cdot\vdash x:A\ \text{Int-Var} \qquad \cfrac{\Delta;\Xi\vdash f:\Pi_{!x:!A}B}{\Delta,x:A;\Xi\vdash f:(\Pi_{!x:!A}B)}\ \text{Int-Weak}}{\Delta,x:A;\Xi\vdash f(!x):B}\ \Pi\text{-E}$$

For the converse, we have to show that we can recover Π-E from the definition of $f(!x)$.

---

[23]At the use of Σ-E, is really where the Frobenius reciprocity comes in, because of the factor Ξ' in the Σ-E-rule.

$$\frac{\Delta;\cdot \vdash a : A \qquad \dfrac{\Delta;\Xi \vdash f : \Pi_{!x:!A}B}{\Delta, x : A;\Xi \vdash f(!x) : B}\ \text{Definition } f(x)}{\dfrac{\Delta;\Xi \vdash f(!x)[a/x] : B[a/x]}{\Delta;\Xi \vdash f(!a) : B[a/x]}\ \text{Int-Tm-Subst}}$$

This shows that individual Π-types correspond to right adjoint functors to substitution along projections. The type theory dictates that Π-types interact well with substitution. This corresponds to the dual Beck-Chevalley condition, as usual.

3. From the categorical semantics of (non-dependent) linear type theory (see e.g. [20] for a very complete account) we know that $\multimap$-types correspond to monoidal closure of the category of contexts. The extra feature in dependent linear type theory is that the syntax dictates that the type formers are compatible with substitution. This means that we also have to restrict the functors $\mathcal{L}(f)$ to preserve the relevant categorical structure.

4. Idem.

5. Idem.

6. Assume that we have !-types. We will define a left adjoint $L_\Delta \dashv M_\Delta$ as $L_\Delta \mathbf{p}_{\Delta,A} := !A$ (this is easily seen to be well-defined up to isomorphism, so we can use AC for a definition on the nose) and, noting that every morphism $\mathbf{p}_{\Delta,A} \longrightarrow \mathbf{p}_{\Delta,B}$ in $\mathcal{C}/\Delta$ is of the form $\langle \mathbf{p}_{\Delta,A}, b\rangle$ for some unique $I \xrightarrow{b} B\{\mathbf{p}_{\Delta,A}\} \in \mathcal{L}(\Delta.A)$, we define L_Δ as acting on b as the map obtained from

$$\dfrac{\dfrac{\Delta, x : A;\cdot \vdash b : B}{\Delta, x : A;\cdot \vdash !b : !B}\ \text{!-I} \qquad \dfrac{}{\Delta; y : !A \vdash y : !a}\ \text{Lin-Var}}{\Delta; y : !A \vdash \text{let } y \text{ be } !x \text{ in } !b : !B}\ \text{!-E}$$

which indeed gives us $L_\Delta(\langle \mathbf{p}_{\Delta,A}, b\rangle) \in \mathcal{L}(\Delta)(!A, !B)$. Note that L_Δ is strong monoidal, as the rules for ! define a natural bijection between terms $\Delta, x : A, y : B;\cdot \vdash t : C$ and $\Delta; x' : !A, y' : !B \vdash t' : C$. In terms of the model, this gives a natural bijection $\mathcal{L}(\Delta)(L_\Delta(M_\Delta A \times M_\Delta B), C) \cong \mathcal{L}(\Delta)(!D, C) \cong \mathcal{L}(\Delta.D)(I, C) \cong \mathcal{L}(\Delta.A.B)(I, C) \cong \mathcal{L}(\Delta)(!A \otimes !B, C)$, where we write D for an object such that $M_\Delta D = M_\Delta A \times M_\Delta B$ (which exists if the product exists), so strong monoidality follows by the Yoneda lemma. (A keen reader can verify that the oplax structure on L_Δ corresponds with the lax structure on M_Δ.)

We exhibit the adjunction by the following isomorphism of hom-sets, where the morphism from left to right comes from !-I and the one from right to left comes from !-E.

$$b \longmapsto b[!x/x']$$

$$\mathcal{L}(\Delta)(L_\Delta \mathbf{p}_{\Delta,A}, B) = \mathcal{L}(\Delta)(!A, B) \underset{\cong}{\overset{\rightarrow}{\leftarrow}} \mathcal{L}(\Delta.A)(I, B\{\mathbf{p}_{\Delta,A}\}) \cong \mathcal{C}/\Delta(\mathbf{p}_{\Delta,A}, \mathbf{p}_{\Delta,B}) = \mathcal{I}(\Delta)(\mathbf{p}_{\Delta,A}, M_\Delta B)$$

$$\text{let } y \text{ be } !x \text{ in } b' \longleftarrow\!\shortmid b'$$

We show how to construct the morphism from left to right, using !-I.

$$\dfrac{\dfrac{\Delta; x' : !A \vdash b : B}{\Delta, x : A; x' : !A \vdash b : B}\ \text{Int-Weak} \qquad \dfrac{\dfrac{\Delta, x : A;\cdot \vdash x : A}{}\ \text{Int-Var}}{\Delta, x : A;\cdot \vdash !x : !A}\ \text{!-I}}{\Delta, x : A;\cdot \vdash b[!x/x'] : B}\ \text{Lin-Tm-Subst}$$

We show to construct the morphism from right to left, using !-E. Suppose we're given $b' \in \mathcal{L}(\Delta.A)(I, B\{\mathbf{p}_{\Delta,A}\})$. From this, we produce a morphism in $\mathcal{L}(\Delta)(!A, B)$ as follows.

$$\dfrac{\dfrac{}{\Delta; y : !A \vdash y : !A}\ \text{Lin-Var} \qquad \Delta, x : A;\cdot \vdash b' : B}{\Delta; y : !A \vdash \text{let } y \text{ be } !x \text{ in } b' : B}\ \text{!E}$$

We leave it up to the reader to verify that these morphisms are mutually inverse, according to !-C and !-U.

Conversely, suppose we have a strong monoidal left adjoint $L_\Delta \dashv M_\Delta$. We define, for $A \in \text{ob}(\mathcal{L}(\Delta))$, $!A := L_\Delta M_\Delta(A)$.

We verify that !-I can be derived from the homset morphism from left to right:

$$\cfrac{\cfrac{\overline{\Delta; x' :!A \vdash x' :!A} \;\; \text{Lin-Var}}{\Delta, x : A; \cdot \vdash !x :!A} \;\; \text{"left to right"} \qquad \Delta; \cdot \vdash a : A}{\Delta; \cdot \vdash !x[a/x] :!A} \;\; \text{Int-Tm-Subst}$$

We verify that, in presence of $\multimap$-types, !-E can be derived from the homset morphism from right to left:

$$\cfrac{\Delta; \Xi \vdash t :!A \qquad \cfrac{\cfrac{\overline{\Delta; w : \Xi' \vdash w : \Xi'} \;\; \text{Lin-Var} \qquad \cfrac{\cfrac{\Delta, x : A; y : \Xi' \vdash b : B}{\Delta, x : A; \cdot \vdash \lambda_{y:\Xi'} b : \Xi' \multimap B} \;\; \multimap\text{-I}}{\Delta; z :!A \vdash \text{let } z \text{ be } !x \text{ in } \lambda_{y:\Xi'} b : \Xi' \multimap B} \;\; \text{"right to left"}}{\Delta; z :!A, \Xi' \vdash \text{let } z \text{ be } !x \text{ in } b[w/y] : B} \;\; \multimap\text{-E}}{\Delta; \Xi, \Xi' \vdash \text{let } t \text{ be } !x \text{ in } b[w/y] : B} \;\; \text{Lin-Tm-Subst}$$

Note that the !-C- and !-U-rules correspond precisely to the fact that our morphisms from left to right and from right to left define a homset isomorphism.

Finally, it is easily verified that the condition that $\mathcal{L}(f) \circ L_\Delta \cong L_{\Delta'} \circ \mathcal{L}(f)$ corresponds exactly to the compatibility of ! with substitution.

7. Suppose we have $\text{Id}_{!A} \dashv -\{\text{diag}_{\Delta,A}\}$, i.e. we have a (natural) homset isomorphism

$$\mathcal{L}(\Delta.A.A\{\mathbf{p}_{\Delta,A}\})(\text{Id}_{!A}(B), C) \underset{\cong}{\overset{\longrightarrow}{\longleftarrow}} \mathcal{L}(\Delta.A)(B, C\{\text{diag}_{\Delta,A}\}).$$

The claim is that $\text{Id}_{!A}(I)$ satisfies the rules for the Id-type of A (or maybe it would be more appropriate to say of $!A$). Indeed, we have Id-I as follows.

$$\cfrac{\cfrac{\cfrac{\overline{\Delta, x : A, x' : A; w : \text{Id}_{!A}(I)(x,x') \vdash w : \text{Id}_{!A}(I)(x,x')} \;\; \text{Lin-Var}}{\Delta, x : A; y : I \vdash \text{refl}_{!x}^y : \text{Id}_{!A}(I)(x,x)} \;\; \text{"left to right"} \qquad \cfrac{}{\Delta, x : A; \cdot \vdash * : I} \;\; \text{I-I}}{\Delta, x : A; \cdot \vdash \text{refl}_{!x} : \text{Id}_{!A}(I)(x,x)} \;\; \text{Lin-Tm-Subst} \qquad \Delta; \cdot \vdash a : A}{\Delta; \cdot \vdash \text{refl}_{!x} : \text{Id}_{!A}(I)(a,a)} \;\; \text{Int-Tm-Subst}$$

We obtain Id-E as follows. Let $\Delta, x : A, x' : A; \cdot \vdash C$ type.

$$\cfrac{\cfrac{\cfrac{\Delta, x : A; B \vdash c : C[x/x']}{\Delta, x : A, x' : A; \text{Id}_{!A}(B) \vdash c' : C} \;\; \text{"right to left"}}{\Delta; \text{Id}_{!A}(B)[a/x, a'/x'] \vdash c'[a/x, a'/x'] : C[a/x, a'/x']} \;\; \text{Lin-Tm-Subst} \quad \Delta; \cdot \vdash a : A \quad \Delta; \cdot \vdash a' : A} \;\; \text{Int-Tm-Subst} \quad (*) \quad \cfrac{\Delta; B' \vdash p : \text{Id}_{!A}(I)[a/x, a'/x']}{\Delta; B[a/x], B' \vdash p' : \text{Id}_{!A}(B)[a/x, a'/x']}}{\Delta; B[a/x], B' \vdash \text{let } (a, a', p) \text{ be } (z, z, \text{refl}_{!z}) \text{ in } c : C[a/x, a'/x]}$$

Here, $(*)$ is a slightly non-trivial step that follows immediately when we note that $\text{Id}_{!A}(B) \cong \text{Id}_{!A}(I) \otimes B\{\mathbf{p}_{\Delta,A}\}$ (by tensoring with id_B). Indeed,

$$\begin{aligned}
\mathcal{L}(\Delta.A.A\{\mathbf{p}_{\Delta,A}\})(\text{Id}_{!A}(B), C) &\cong \mathcal{L}(\Delta.A)(B, C\{\text{diag}_{\Delta,A}\}) \\
&\cong \mathcal{L}(\Delta.A)(I, (B\{\mathbf{p}_{\Delta,A}\} \multimap C)\{\text{diag}_{\Delta,A}\}) \\
&\cong \mathcal{L}(\Delta.A.A\{\mathbf{p}_{\Delta,A}\})(\text{Id}_{!A}(I), B\{\mathbf{p}_{\Delta,A}\} \multimap C) \\
&\cong \mathcal{L}(\Delta.A.A\{\mathbf{p}_{\Delta,A}\})(\text{Id}_{!A}(I) \otimes B\{\mathbf{p}_{\Delta,A}\}, C).
\end{aligned}$$

Since all these isomorphisms are natural in C, the Yoneda lemma says that $\text{Id}_{!A}(B) \cong \text{Id}_{!A}(I) \otimes B\{\mathbf{p}_{\Delta,A}\}$.

Conversely, suppose we have Id-types. Then, define $\text{Id}_{!A}(B) := \text{Id}_{!A} \otimes B\{\mathbf{p}_{\Delta,A}\}$, with the obvious extension on morphisms. Then, we obtain the morphism "left to right" as follows.

$$\cfrac{\cfrac{\Delta, x : A, x' : A; z : \text{Id}_{!A}, y : B \vdash c : C \qquad \cfrac{}{\Delta, x : A; \cdot \vdash x : A} \;\; \text{Int-Var}}{\Delta, x : A; z : \text{Id}_{!A}[x/x'], y : B \vdash c[x/x'] : C[x/x']} \;\; \text{Int-Tm-Subst} \qquad \cfrac{\cfrac{}{\Delta, x : A; \cdot \vdash x : A} \;\; \text{Int-Var}}{\Delta, x : A; \cdot \vdash \text{refl}_{!x} : \text{Id}_{!A}(x,x)} \;\; \text{Id-I}}{\Delta, x : A; y : B \vdash c' : C[x/x']} \;\; \text{Lin-Tm-Subst}$$

The morphism "right to left" is obtained as follows.

$$\cfrac{\Delta, x_0 : A; y : B \vdash c : C[x_0/x_1] \qquad \cfrac{\cfrac{}{\Delta, x_0 : A, x_1 : A; w : \text{Id}_{!A} \vdash w : \text{Id}_{!A}} \;\; \text{Lin-Var} \qquad \cfrac{}{\Delta, x_0 : A, x_1 : A; \cdot \vdash x_i : A} \;\; \text{Int-Var}}{\Delta, x_0 : A, x_1 : A; w : \text{Id}_{!A}, y : B \vdash c' : C} \;\; \text{Id-E}}$$

27

We leave it to the reader to verify that the Id-C- and Id-U-rules translate precisely into the "right to left" and "left to right" morphisms being inverse. $\qquad\square$

The semantics of ! suggests an alternative definition for the notion of a comprehension: if we Σ-types in a strong sense, it is a derived notion!

Theorem 11 (Lawvere Comprehension). *Given a strict indexed monoidal category $(\mathcal{C}, \mathcal{L})$ with left adjoints Σ_{Lf} to $\mathcal{L}(f)$ for arbitrary $\Delta' \xrightarrow{f} \Delta \in \mathcal{C}$, then we can define $\mathcal{C}/\Delta \xrightarrow{L_\Delta} \mathcal{L}(\Delta)$ by*

$$L_\Delta(-) \coloneqq \Sigma_{L-}I.$$

In that case, $(\mathcal{C}, \mathcal{L})$ has a comprehension schema iff L_Δ has a right adjoint M_Δ (which then automatically satisfies $M_{\Delta'} \circ \mathcal{L}(f) = \mathcal{L}(f) \circ M_\Delta$ for all $\Delta' \xrightarrow{f} \Delta \in \mathcal{C}$). That is, our notion of comprehension generalises that of [21].

Finally, if Σ_{Lf} are demanded to satisfy the Beck-Chevalley condition and Frobenius reciprocity, then $(\mathcal{C}, \mathcal{L})$ satisfies the comprehension schema iff it admits !-types.

Proof. Suppose that we have said right adjoints M_Δ. We will construct a comprehension schema.

We define $\mathbf{p}_{\Delta,A} \coloneqq M_\Delta(A)$ and

$$\mathcal{L}(\Delta')(I, A\{f\}) \xrightarrow{\cong} \mathcal{L}(\Delta)(\Sigma_{Lf}I_{\Delta'}, A) = \mathcal{L}(\Delta)(L_\Delta f, A) \xrightarrow{\cong} \mathcal{C}/\Delta(f, M_\Delta A)$$

$$a \longmapsto \qquad\qquad a_f \longmapsto \qquad\qquad \langle f, a \rangle,$$

where the first natural isomorphism comes from the adjunction $\Sigma_{Lf} \dashv -\{f\}$ and the second one comes from the adjunction $L_\Delta \dashv M_\Delta$. Note that, by definition, $\mathbf{p}_{\Delta,A}\langle f, a \rangle = f$.

In particular, we obtain a unique $\mathbf{v}_{\Delta,A} \in \mathcal{L}(\Delta.A)(I, A\{\mathbf{p}_{\Delta,A}\})$ inducing id_{M_Δ} as $\langle \mathbf{p}_{\Delta,A}, \mathbf{v}_{\Delta,A} \rangle$. Finally, the Yoneda lemma (i.e. naturality of these isomorphisms) says that $\mathbf{v}_{\Delta,A}\{\langle f, a \rangle\} = a$.

Conversely, suppose we're given a comprehension schema. Then, we know, by theorem 5, that we can define a comprehension functor M_Δ such that $M_{\Delta'} \circ \mathcal{L}(f) = \mathcal{L}(f) \circ M_\Delta$. Then we have the following:

$$\mathcal{C}/\Delta(f, M_\Delta A) \xrightarrow{\cong} \mathcal{L}(\Delta')(I, A\{f\}) \xrightarrow{\cong} \mathcal{L}(\Delta)(\Sigma_{Lf}I_{\Delta'}, A) = \mathcal{L}(\Delta)(L_\Delta f, A)$$

$$\langle f, a \rangle \longmapsto \qquad\qquad a \longmapsto \qquad\qquad a_f,$$

where the first isomorphism is precisely the representation defined by our comprehension and the second isomorphism comes from the fact that $\Sigma_{Lf} \dashv -\{f\}$.

Finally, the following calculation shows that it follows from Frobenius reciprocity and Beck-Chevalley that L_Δ is strong monoidal:

$$\begin{aligned}
L_\Delta(f) \otimes L_\Delta(g) &= (\Sigma_{Lf}I_{\mathrm{dom}f}) \otimes (\Sigma_{Lg}I_{\mathrm{dom}g}) \\
&= \Sigma_{Lg}((\Sigma_{Lf}I_{\mathrm{dom}f})\{g\} \otimes I_{\mathrm{dom}g}) \quad \text{(Frobenius reciprocity)} \\
&= \Sigma_{Lg}((\Sigma_{Lf}I_{\mathrm{dom}f})\{g\}) \\
&= \Sigma_{Lg}\Sigma_{Lf^*g}I_{\mathrm{dom}f}\{g^*f\} \quad \text{(Beck-Chevalley)} \\
&= \Sigma_{Lg}\Sigma_{Lf^*g}I_{\mathrm{dom}_{f\times g}} \\
&= \Sigma_{L(f\times g)}I_{\mathrm{dom}_{f\times g}} \\
&= L_\Delta(f \times g).
\end{aligned}$$

$\qquad\square$

Theorem 12 (Type Formers in $\mathcal{I}$). *$\mathcal{I}$ supports Σ-types iff $\mathrm{ob}(\mathcal{I})$ is closed under compositions (as morphisms in $\mathcal{C}$). It supports Id-types iff $\mathrm{ob}(\mathcal{I})$ is closed under post-composition with maps $\mathrm{diag}_{\Delta,A}$. If $\mathcal{L}$ supports !- and Π-types, then $\mathcal{I}$ supports Π-types. Moreover, we have that*

$$\Sigma_{!A}!B \cong L(\Sigma_{MA}MB) \qquad\qquad \mathrm{Id}_{!A}(!B) \cong L\mathrm{Id}_{MA}(MB) \qquad\qquad M\Pi_{!B}C \cong \Pi_{MB}MC.$$

Proof. We write out the adjointness condition

$$\mathcal{I}(\Delta)(\Sigma_{\mathbf{p}_{\Delta,B}}f, \mathbf{p}_{\Delta,D}) \overset{!}{\cong} \mathcal{I}(\Delta.B)(f, \mathbf{p}_{\Delta,D}\{\mathbf{p}_{\Delta,B}\})$$
$$\cong \mathcal{I}(\Delta.B)(f, \mathbf{p}_{\Delta,D\{\mathbf{p}_{\Delta,B}\}})$$
$$\cong \mathcal{L}(\Delta.B.C)(I, D\{\mathbf{p}_{\Delta,B}\}\{f\})$$
$$\cong \mathcal{L}(\Delta.B.C)(I, D\{\mathbf{p}_{\Delta,B} \circ f\})$$
$$\cong \mathcal{I}(\Delta)(\mathbf{p}_{\Delta,B} \circ f, \mathbf{p}_{\Delta,D}).$$

Now, the Yoneda lemma gives us that $\Sigma_{\mathbf{p}_{\Delta,B}}f = \mathbf{p}_{\Delta,B} \circ f$.

Similarly,

$$\mathcal{I}(\Delta.A.A)(\mathrm{Id}_{\mathbf{p}_{\Delta,A}}(f), \mathbf{p}_{\Delta.A.A,C}) \overset{!}{\cong} \mathcal{I}(\Delta.A)(f, \mathbf{p}_{\Delta.A.A,C}\{\mathrm{diag}_{\Delta,A}\})$$
$$\cong \mathcal{L}(\Delta.A.B)(I, C\{\mathrm{diag}_{\Delta,A}\}\{f\})$$
$$\cong \mathcal{L}(\Delta.A.B)(I, C\{\mathrm{diag}_{\Delta,A} \circ f\})$$
$$\cong \mathcal{I}(\Delta.A.A)(\mathrm{diag}_{\Delta,A} \circ f, \mathbf{p}_{\Delta.A.A,C}),$$

so $\mathrm{diag}_{\Delta,A} \circ f$ models $\mathrm{Id}_{\mathbf{p}_{\Delta,A}}(f)$.

Finally,

$$\mathcal{I}(\Delta)(M_\Delta D, \Pi_{\mathbf{p}_{\Delta,B}}\mathbf{p}_{\Delta.B,C}) \overset{!}{\cong} \mathcal{I}(\Delta.B)((M_\Delta D)\{\mathbf{p}_{\Delta,B}\}, \mathbf{p}_{\Delta.B,C})$$
$$\cong \mathcal{I}(\Delta.B)((M_\Delta D)\{\mathbf{p}_{\Delta,B}\}, M_{\Delta.B}C)$$
$$\cong \mathcal{L}(\Delta.B)(L_{\Delta.B}((M_\Delta D)\{\mathbf{p}_{\Delta,B}\}), C)$$
$$\cong \mathcal{L}(\Delta.B)((L_\Delta M_\Delta D)\{\mathbf{p}_{\Delta,B}\}, C)$$
$$\cong \mathcal{L}(\Delta)(L_\Delta M_\Delta D, \Pi_{!B}C)$$
$$\cong \mathcal{I}(\Delta)(M_\Delta D, M_\Delta \Pi_{!B}C)$$

Again, using the Yoneda lemma, we conclude that $M_\Delta \Pi_{!B}C$ models $\Pi_{M_{\Delta.B}}M_{\Delta.B}C$.

In all cases, we have not worried about Beck-Chevalley (and Frobenius reciprocity for Σ-types) as they are trivially seen to hold.

Note that if $\mathcal{L}$ has !- and Σ-types, then

$$\mathcal{L}(\Delta)(L_\Delta(\Sigma_{M_{\Delta.A}A}M_{\Delta.A}B), C) \cong \mathcal{I}(\Delta)(\Sigma_{M_{\Delta.A}A}M_{\Delta.A}B, M_\Delta C)$$
$$\cong \mathcal{I}(\Delta.A)(M_{\Delta.A}B, (M_\Delta C)\{\mathbf{p}_{\Delta,A}\})$$
$$\cong \mathcal{I}(\Delta.A)(M_{\Delta.A}B, M_\Delta(C\{\mathbf{p}_{\Delta,A}\}))$$
$$\cong \mathcal{L}(\Delta.A)(!B, C\{\mathbf{p}_{\Delta,A}\})$$
$$\cong \mathcal{L}(\Delta)(\Sigma_{!A}!B, C).$$

By the Yoneda lemma, conclude that $\Sigma_{!A}!B \cong L_\Delta(\Sigma_{M_{\Delta.A}A}M_{\Delta.A}B)$.

Note that, in case $\mathcal{L}$ admits !- and Id-types,

$$\mathcal{L}(\Delta.A.A)(\mathrm{Id}_{!A}(!B), C) \cong \mathcal{L}(\Delta.A)(!B, C\{\mathrm{diag}_{\Delta,A}\})$$
$$\cong \mathcal{I}(\Delta.A)(M_{\Delta.A}B, M_{\Delta.A}(C\{\mathrm{diag}_{\Delta,A}\}))$$
$$\cong \mathcal{I}(\Delta.A)(M_{\Delta.A}B, M_{\Delta.A.A}(C)\{\mathrm{diag}_{\Delta,A}\})$$
$$\cong \mathcal{I}(\Delta.A.A)(\mathrm{diag}_{\Delta,A} \circ M_{\Delta.A}B, M_{\Delta.A}(C))$$
$$\cong \mathcal{I}(\Delta.A.A)(\mathrm{Id}_{M_\Delta A}(M_{\Delta.A}B), M_{\Delta.A}(C))$$
$$\cong \mathcal{L}(\Delta.A.A)(L_{\Delta.A.A}\mathrm{Id}_{M_\Delta A}(M_{\Delta.A}B), C).$$

We conclude that $\mathrm{Id}_{!A}(!B) = L_{\Delta.A.A}\mathrm{Id}_{M_\Delta A}(M_{\Delta.A}B)$ and in particular $\mathrm{Id}_{!A}(I) \cong L_{\Delta.A.A}\mathrm{Id}_{M_\Delta A}(\mathrm{id}_{\Delta.A})$. The last statement is easily seen to also be valid in absence of $\top$-types. $\qquad\square$

Remark 6 (Dependent Seely Isomorphisms?). Note that, in our setup, we have a version of the normal Seely isomorphisms in each fibre. Indeed, suppose $\mathcal{L}$ supports $\top$-, &, and !-types. Then, $M_\Delta(\top) = \mathrm{id}_\Delta$ and $M_\Delta(A\&B) = M_\Delta(A) \times M_\Delta(B)$, as M_Δ has a left adjoint and therefore preserves products. Now, L_Δ is strong monoidal and $!_\Delta = L_\Delta M_\Delta$, so it follows that $!_\Delta\top = I$ and $!_\Delta(A\&B) = !_\Delta A \otimes !_\Delta B$.

Now, theorem 12 suggests the possibility of similar Seely isomorphisms on us for Σ-types and Id-types. Indeed, $\mathcal{I}$ supports Σ-types iff we have additive Σ-types in $\mathcal{L}$ in the sense of objects $\Sigma_A^\& B$ such that

$$M\Sigma_A^\& B \cong \Sigma_{MA}MB \quad \text{and hence} \quad !\Sigma_A^\& B \cong \Sigma_{!A}^\otimes !B,$$

where we suggestively write $\Sigma^\otimes$ for the usual multiplicative Σ-type in $\mathcal{L}$. In an ideal world, one would hope that $\Sigma_A^\& B$ generalise $A\&B$ in the same way that $\Sigma_{!A}B$ is a dependent generalisation of $!A \otimes B$.

Similarly, we get a notion of additive Id-types: $\mathcal{I}$ supports Id-types iff we have objects $\mathrm{Id}_A^\&(B)$ in $\mathcal{L}$ such that

$$M\mathrm{Id}_A^\&(B) \cong \mathrm{Id}_{MA}(MB) \quad \text{and hence} \quad !\mathrm{Id}_A^\&(B) \cong \mathrm{Id}_{!A}^\otimes(!B),$$

writing $\mathrm{Id}^\otimes$ for the usual (multiplicative) Id-type in $\mathcal{L}$. Note that this suggests that, in the same way that $\mathrm{Id}_{!A}^\otimes(B) \cong \mathrm{Id}_{!A}^\otimes(I) \otimes B$ (a sense in which usual Id-types are multiplicative connectives), $\mathrm{Id}_A^\&(B) \cong \mathrm{Id}_A^\&(\top)\&B$. In fact, if we have $\top$- and &-types, we only have to give $\mathrm{Id}_A^\&(\top)$ and can then *define* $\mathrm{Id}_A^\&(B) := \mathrm{Id}_A^\&(\top)\&B$ to obtain additive Id-types in generality.

A fortiori, if some M_Δ is essentially surjective, we obtain such additive Σ- and Id-types in the fibre over Δ. In particular, we are in this situation if $L. \dashv M.$ is the usual co-Kleisli adjunction of !., where $\mathcal{I}(\cdot) = \mathcal{C}$. This shows that if we are hoping to obtain a model of ILDTT indexed over the co-Kleisli category, in the natural way, we need to support these additive connectives.

It still remains to be seen if an understanding of these "additive connectives" can be found from a syntactic point of view. Similarly, it seems like the natural models of ILDTT do not support them. Finally, it is difficult to come up with an intuitive interpretation of the meaning of such connectives, in the sense of a resource interpretation. Clearly, further investigation is necessary, here.

4 Some Discrete Models: Monoidal Families

We discuss a simple class of models in terms of families with values in a symmetric monoidal category. On a logical level, what the construction boils down to is starting with a model $\mathcal{V}$ of a linear propositional logic and taking the cofree linear predicate logic on Set with values in this propositional logic. This important example illustrates how Σ- and Π-types can represent infinitary additive disjunctions and conjunctions. The model is discrete in nature, however, and in that respect not representative for the type theory.

Suppose $\mathcal{V}$ is a symmetric monoidal category. We can then consider a strict Set-indexed category, defined through the following enriched Yoneda embedding $\mathrm{Fam}(\mathcal{V}) := \mathcal{V}^- := \mathrm{SMCat}(-, \mathcal{V})$:

$$\mathrm{Set}^{op} \xrightarrow{\mathrm{Fam}(\mathcal{V})} \mathrm{SMCat} \qquad\qquad S \xrightarrow{f} S' \longmapsto \mathcal{V}^S \xleftarrow{-\circ f} \mathcal{V}^{S'}.$$

Note that this definition naturally extends to a functor Fam.

Theorem 13 (Families Model ILDTT). *The construction* Fam *adds type dependency on* Set *cofreely in the sense that it is right adjoint to the forgetful functor* ev_1 *that evaluates a model of linear dependent type theory at the empty context to obtain a model of linear propositional type theory (where* $\mathrm{SMCat}^{\mathrm{Set}^{op}}_{\mathrm{compr}}$ *is the full subcategory of* $\mathrm{SMCat}^{\mathrm{Set}^{op}}$ *on the objects with comprehension):*

$$\mathrm{SMCat} \underset{\mathrm{Fam}}{\overset{\mathrm{ev}_1}{\underset{\longrightarrow}{\overset{\longleftarrow}{\perp}}}} \mathrm{SMCat}^{\mathrm{Set}^{op}}_{\mathrm{compr}}.$$

Proof. $\mathrm{Fam}(\mathcal{V})$ admits a comprehension, by the following isomorphism

$$\begin{aligned}
\mathrm{Fam}(\mathcal{V})(S)(I, B\{f\}) &= \mathcal{V}^S(I, B \circ f) \\
&= \Pi_{s \in S} \mathcal{V}(I, B(f(s))) \\
&\cong \mathrm{Set}/S(S \xrightarrow{\mathrm{id}_S} S, \Sigma_{s \in S} \mathcal{V}(I, B(f(s))) \xrightarrow{\mathrm{fst}} S) \\
&\cong \mathrm{Set}/S'(S \xrightarrow{f} S', \Sigma_{s' \in S'} \mathcal{V}(I, B(s')) \xrightarrow{\mathrm{fst}} S') \\
&= \mathrm{Set}/S'(f, \mathbf{p}_{S',B}),
\end{aligned}$$

where $\mathbf{p}_{S',B} := \Sigma_{s' \in S'} \mathcal{V}(I, B(s')) \xrightarrow{\mathrm{fst}} S'$. ($\mathbf{v}_{S',B}$ is obtained as the image of $\mathrm{id}_{S'} \in \mathrm{Set}/S'$ under this isomorphism.) To see that $\mathrm{ev}_1 \dashv \mathrm{Fam}$, note that the following naturality diagrams for elements $1 \xrightarrow{s} S$

$$\begin{array}{ccccc}
1 & \mathrm{ev}_1(\mathcal{L}) = \mathcal{L}(1) & \xrightarrow{\phi_1} & \mathcal{V} = \mathrm{Fam}(\mathcal{V})(1) \\
\Big\uparrow{\scriptstyle s} & \Big\uparrow{\scriptstyle -\{s\}} & & \Big\uparrow{\scriptstyle -\circ s} \\
S & \mathcal{L}(S) & \xrightarrow[\phi_S]{} & \mathcal{V}^S = \mathrm{Fam}(\mathcal{V})(S)
\end{array}$$

together with the fact that all $1 \xrightarrow{s} S$ are jointly surjective and hence the fact that $- \circ s$ are jointly injective means that a natural transformation $\phi \in \mathrm{SMCat}^{\mathrm{Set}^{op}}(\mathcal{L}, \mathrm{Fam}(\mathcal{V}))$ is uniquely determined by $\phi_1 \in \mathrm{Cat}(\mathrm{ev}_1(\mathcal{L}), \mathcal{V})$. $\qquad\square$

We have the following results for type formers[24].

Theorem 14 (Type Formers for Families). *$\mathcal{V}$ has small coproducts that distribute over $\otimes$ iff* $\mathrm{Fam}(\mathcal{V})$ *supports Σ-types. In that case,* $\mathrm{Fam}(\mathcal{V})$ *also supports 0- and $\oplus$-types (which correspond precisely to finite distributive coproducts).*

---

[24] We do not examine Id-types here, as they will precisely correspond to intuitionistic identity type in $\mathcal{I}$, which probably is not very interesting seeing that $\mathcal{I}$ is a submodel of the normal set-based model of dependent types (i.e. fibred sets, which is equivalent to indexed sets: Set-valued families).

$\mathcal{V}$ has small products iff $\mathrm{Fam}(\mathcal{V})$ supports Π-types. In that case, $\mathrm{Fam}(\mathcal{V})$ also supports $\top$- and $\&$-types (which correspond precisely to finite products).

$\mathrm{Fam}(\mathcal{V})$ supports $\multimap$-types iff $\mathcal{V}$ is monoidal closed.

$\mathrm{Fam}(\mathcal{V})$ supports $!$-types iff $\mathcal{V}$ has small coproducts of I that are preserved by $\otimes$ in the sense that the canonical morphism $\mathrm{coprod}_S(\Xi' \otimes I) \longrightarrow \Xi' \otimes \mathrm{coprod}_S I$ is an isomorphism for any $\Xi' \in \mathrm{ob}\,\mathcal{V}$ and $S \in \mathrm{ob}\,\mathrm{Set}$. In particular, if $\mathrm{Fam}(\mathcal{V})$ supports Σ-types, then it also supports $!$-types.

$\mathrm{Fam}(\mathcal{V})$ supports Id-types if $\mathcal{V}$ has an initial object. Supposing that $\mathcal{V}$ has a terminal object, the only if also holds.

Proof. The statement about 0-, $\oplus$-, $\top$-, and $\&$-types should be clear from the previous sections, as products and coproducts in $\mathcal{V}^S$ are pointwise (and hence automatically preserved under substitution).

We will denote coproducts in $\mathcal{V}$ with $\oplus$. Then,

$$\Pi_{s' \in S'} \mathcal{V}(\bigoplus_{s \in f^{-1}(s')} A(s), B(s')) \cong \Pi_{s' \in S'}\Pi_{s \in f^{-1}(s')}\mathcal{V}(A(s), B(s'))$$

$$\cong \Pi_{s \in \Sigma_{s' \in S'} f^{-1}(s')}\mathcal{V}(A(s), B(f(s)))$$
$$\cong \Pi_{s \in S}\mathcal{V}(A(s), B(f(s)))$$
$$= \mathcal{V}^S(A, B \circ f).$$

So, we see that we can define $\Sigma_{Lf}(A)(s') := \bigoplus_{s \in f^{-1}(s')} A(s)$ to get a left adjoint $\Sigma_{Lf} \dashv -\{f\}$, if we have coproducts. (With the obvious definition on morphisms coming from the coCartesian monoidal structure on $\mathcal{V}$.) Conversely, we can clearly use Σ_{Lf} to define any coproduct by using, for instance, an identity function for f on the set we want to take a coproduct over and a family A that denotes the objects we want to comultiply. The Beck-Chevalley condition is taken care of by the fact that our substitution morphisms are given by precomposition. Frobenius reciprocity precisely corresponds to distributivity of the coproducts over $\otimes$.

Similarly, if $\mathcal{V}$ has products, we will denote them with $\&$ to suggest the connections with linear type theory. In that case, we can define $\Pi_{Lf}(A)(s') := \&_{s \in f^{-1}(s')} A(s)$ to get a right adjoint $-\{f\} \dashv \Pi_{Lf}$. (With the obvious definition on morphisms coming from the Cartesian monoidal structure on $\mathcal{V}$.) Indeed,

$$\Pi_{s' \in S'} \mathcal{V}(B(s'), \&_{s \in f^{-1}(s')} A(s)) \cong \Pi_{s' \in S'}\Pi_{s \in f^{-1}(s')}\mathcal{V}(B(s'), A(s))$$

$$\cong \Pi_{s \in \Sigma_{s' \in S'} f^{-1}(s')}\mathcal{V}(B(f(s)), A(s))$$
$$\cong \Pi_{s \in S}\mathcal{V}(B(f(s)), A(s))$$
$$= \mathcal{V}^S(B \circ f, A).$$

Again, in the same way as before, we can construct any product using Π_{Lf}. The dual Beck-Chevalley condition comes for free as our substitution morphisms are precomposition.

The claim about $\multimap$-types follows immediately from the previous section: $\mathrm{Fam}(\mathcal{V})$ supports $\multimap$-types iff all its fibres have a monoidal closed structure that is preserved by the substitution functors. Seeing that our monoidal structure is pointwise, the same will hold for any monoidal closed structure. Seeing that substitution is given by precomposition the preservation requirement comes for free.

The characterisation of $!$-types is given by 3, which tells us we can define $!A := \Sigma_{\mathbf{p}_{S',A}} I = s' \mapsto \bigoplus_{\mathcal{V}(I, A(s'))} I$ and conversely.

Finally, for Id-types, note that the adjointness condition $\mathrm{Id}_{!A} \dashv -\{\mathrm{diag}_{\Delta, A}\}$ boils down to the requirement (*)

$$\Pi_{s \in S}\Pi_{a \in A(s)}\mathcal{V}(B(s, a), C(s, a, a)) \cong \mathcal{V}^{\Sigma_{s \in S} A(s)}(B, C\{\mathrm{diag}_{S,A}\})$$

$$\stackrel{!}{\cong} \mathcal{V}^{\Sigma_{s \in S} A(s) \times A(s)}(\mathrm{Id}_{!A}(B), C)$$
$$\cong \Pi_{s \in S}\Pi_{a \in A(s)}\Pi_{a' \in A(s)}\mathcal{V}(\mathrm{Id}_{!A}(B)(s, a, a'), C(s, a, a')).$$

We see that if we have an initial object $0 \in \mathrm{ob}(\mathcal{V})$, we can define

$$\mathrm{Id}_{!A}(B)(s, a, a') := \begin{cases} B(s, a) & \text{if } a = a' \\ 0 & \text{else} \end{cases}$$

For a partial converse, suppose we have a terminal object $\top \in \mathcal{V}$. Let $V \in \mathrm{ob}(\mathcal{V})$. Let $S \coloneqq \{*\}$, $A \coloneqq \{0,1\}$ and C s.t. $C(0,0) = C(1,1) = C(0,1) = \top$ and $C(1,0) = V$. Then, (*) becomes the condition that $\{*\} \cong \mathcal{V}(\mathrm{Id}_{!A}(B)(1,0), V)$. We conclude that $\mathrm{Id}_{!A}(B)(1,0)$ is initial in $\mathcal{V}$. $\qquad \square$

Remark 7. Note that an obvious way to guarantee distributivity of coproducts over $\otimes$ is by demanding that $\mathcal{V}$ is monoidal closed.

Remark 8. It is easily seen that Σ-types in $\mathcal{I}$, or additive Σ-types in $\mathcal{L} = \mathrm{Fam}(\mathcal{V})$, boil down to having an object $\mathrm{or}_{s \in S} C(s) \in \mathrm{ob}(\mathcal{V})$ for a family $(C(s) \in \mathrm{ob}(\mathcal{V}))_{s \in S}$ such that $\Sigma_{s \in S} \mathcal{V}(I, C(s)) \cong \mathcal{V}(I, \mathrm{or}_{s \in S} C(s))$. Similarly, Id-types in $\mathcal{I}$, or additive Id-types in $\mathcal{L}$, boil down to having objects $\mathrm{one}, \mathrm{zero} \in \mathrm{ob}(\mathcal{V})$ such that $\mathcal{V}(I, \mathrm{one}) \cong 1$ and $\mathcal{V}(I, \mathrm{zero}) = 0$.

Two particularly simple concrete examples of $\mathcal{V}$ come to mind that can accomodate all type formers and form a nice illustration: a category $\mathcal{V} = \mathrm{Vect}_F$ of vector spaces over a field F, with the tensor product, and the category $\mathcal{V} = \mathrm{Set}_*$ of pointed sets, with the smash product. All type formers get their obvious interpretation, but let us stop to think about ! for a second as it is a novelty of ILDTT that it gets uniquely determined by the indexing, while in propositional linear type theory we might have several different choices. In the first example, ! boils down to the following: $(!B)(s') = \mathrm{coprod}_{\mathrm{Vect}_F(F, B(s'))} F \cong \bigoplus_{B(s')} F$, i.e. taking the vector space freely spanned by all the vectors. In the second example, $(!B)(s') = \mathrm{coprod}_{\mathrm{Set}_*(2_*, B(s'))} 2_* = \bigvee_{B(s')} 2_* = B(s') + \{*\}$, i.e. ! freely adds a new basepoint. These models show the following.

Theorem 15 (DTT, DILL$\subsetneq$ILDTT)**.** *ILDTT is a proper generalisation of DTT and DILL: we have inclusions of the classes of models DTT,DILL$\subsetneq$ILDTT.*

Proof. By the Grothendieck construction, every split fibration can equivalently be seen as the category of elements of the corresponding strict indexed category defined by the fibres. Under this equivalence split full comprehension categories with finite fibrewise products (i.e. models of DTT with 1- and $\times$-types) correspond precisely to strict indexed Cartesian monoidal categories with comprehension where the comprehension functor is full and faithful. Clearly, these are a special case of our notion of model of ILDTT. Moreover, in such cases clearly $!A \cong A$. From their categorical description, it is also clear that the other connectives of ILDTT reduce to those of DTT. This proves the inclusion DDT$\subset$ILDTT.

The models described above are clearly more general than those of DTT, as we are dealing with a non-Cartesian monoidal structure on the fibre categories. This proves that the inclusion is proper.

We have seen that the Fam-construction realises the category of models of DILL as a reflective subcategory of the category of models of ILDTT. Moreover, from various non-trivial models of DTT indexed over other categories than Set it is clear that this inclusion is proper as well.

Finally, we note that these inclusions still remain valid in the sub-algebraic setting where we do not have I- and $\otimes$-types. A simple variation of the argument using multicategories rather than monoidal categories does the trick. $\qquad \square$

Although this class of families models is important, it is clear that it only represents a very limited part of the generality of ILDTT: not every model of ILDTT is either a model of DTT or of DILL. Hence, we are in need of models that are less discrete in nature but still linear, if we are hoping to observe interesting new phenomena arising from the connectives of linear dependent type theory. Some suggestions and work in progress will be discussed in the next section.

5 Conclusions and Future Work

We hope to have convinced the reader that linear dependent types fit very naturally in the landscape of existing type theories and that they admit a rich theory rather than being limited to the specific examples that had been considered so far. There is a larger story connecting these examples!

On a syntactic level our system is a very natural blend between (intuitionistic) dependent type theory and dual intuitionistic linear logic. On a semantic level, if one starts with the right notion of model for dependent types, the linear generalisation is obtained through the usual philosophy of passing from Cartesian to symmetric monoidal structures. The resulting notion of a model forms a natural blend between comprehension categories, modelling DTT, and linear-non-linear models, modelling DILL.

It is very pleasing to see that all the syntactically natural rules for type formers are equivalent to their semantic counterparts that would be expected based on the traditions of categorical logic of dependent types and linear types. In particular, from the point of view of logic, it is interesting to see that the categorical semantics seems to have a preference for multiplicative quantifiers.

Finally, have shown that, as in the intuitionistic case, we can represent infinitary (additive) disjunctions and conjunctions in linear type theory, through cofree Σ- and Π-types, indexed over Set. In particular, this construction exhibits a family of non-trivial truly linear models of dependent types, providing an essential reality check for our system.

Despite what might be expected from this paper, much of this work has been very semantically motivated, by specific models. In joint work with Samson Abramsky, a model of linear dependent types with comprehension has been constructed in a category of coherence spaces. Apart from the usual type constructors from linear logic, it also supports Σ-, Π-, and Id-types. A detailed account of this model will be made available soon.

In addition to providing (what as far as we are aware is) the first non-trivial, semantically motivated model of such a type system, this work serves as a stepping stone for a model that we are currently developing in a category of games, together with Samson Abramsky and Radha Jagadeesan. This, in particular, should provide a game semantics for dependent type theory.

An indexed category of spectra up to homotopy over topological spaces has been studied in e.g. [15, 3] as a setting for stable homotopy theory. It has been shown to admit I-, $\otimes$-, $\multimap$-, and Σ-types. The natural candidate for a comprehension adjunction, here, is that between the infinite suspension spectrum and the infinite loop space: $L \dashv M = \Sigma^\infty \dashv \Omega^\infty$. A detailed examination of the situation and an explanation of the relation with the Goodwillie calculus would be very desirable here, though. This might fit in with our related objective of giving a linear analysis of homotopy type theory.

Another fascinating possibility is that of models related to quantum mechanics. Non-dependent linear type theory has found very interesting interpretations in quantum computation, e.g. [22]. The question rises if the extension to dependent linear types has a natural counterpart in physics. In [16], Urs Schreiber has recently sketched how linear dependent types can serve as a language to talk about quantum field theory and quantisation in particular. There are plenty of interesting open questions here.

Finally, there are still plenty of theoretical questions within the type theory. Can we expect to have interesting models with type dependency on the co-Kleisli category of ! and can we make sense of additive Σ- and Id-types, e.g. from a syntactic point of view? Is there an equivalent of strong/dependent E-rules for ILDTT? Does the Curry-Howard correspondence extend in its full glory: do we have a propositions-as-types interpretation of linear predicate logic in ILDTT? These questions need to be addressed by a combination of research into the formal system and study of specific models. We hope that the general framework we sketched will play its part in connecting all the different sides of the story: from syntax to semantics; from computer science and logic to geometry and physics.

References

[1] Cervesato, I., Pfenning, F.: A linear logical framework. In: LICS'96. Proceedings., IEEE (1996) 264–275

[2] Shulman, M.: Enriched indexed categories. Theory and Applications of Categories **28**(21) (2013) 616–695

[3] Ponto, K., Shulman, M.: Duality and traces for indexed monoidal categories. Theory and Applications of Categories **26**(23) (2012) 582–659

[4] Barber, A.: Dual intuitionistic linear logic. Technical Report ECS-LFCS-96-347, University of Edinburgh, Edinburgh (1996)

[5] Hofmann, M.: Syntax and semantics of dependent types. In: Extensional Constructs in Intensional Type Theory. Springer (1997) 13–54

[6] Jacobs, B.: Comprehension categories and the semantics of type dependency. Theoretical Computer Science **107**(2) (1993) 169–207

[7] Church, A.: A formulation of the simple theory of types. The journal of symbolic logic **5**(02) (1940) 56–68

[8] Howard, W.A.: The formulae-as-types notion of construction. (1995)

[9] Lambek, J.: Deductive systems and categories iii. cartesian closed categories, intuitionist propositional calculus, and combinatory logic. In: Toposes, algebraic geometry and logic. Springer (1972) 57–82

[10] Martin-Löf, P.: An intuitionistic theory of types. Twenty-five years of constructive type theory **36** (1998) 127–172

[11] Girard, J.Y.: Linear logic. Theoretical computer science **50**(1) (1987) 1–101

[12] Dal Lago, U., Gaboardi, M.: Linear dependent types and relative completeness. In: LiCS 2011. Proceedings., IEEE (2011) 133–142

[13] Petit, B., et al.: Linear dependent types in a call-by-value scenario. In: Proceedings of the 14th symposium on Principles and practice of declarative programming, ACM (2012) 115–126

[14] Gaboardi, M., Haeberlen, A., Hsu, J., Narayan, A., Pierce, B.C.: Linear dependent types for differential privacy. In: ACM SIGPLAN Notices. Volume 48., ACM (2013) 357–370

[15] May, J.P., Sigurdsson, J.: Parametrized homotopy theory. Number 132. American Mathematical Soc. (2006)

[16] Schreiber, U.: Quantization via linear homotopy types. arXiv preprint arXiv:1402.7041 (2014)

[17] Power, A.J.: A general coherence result. Journal of Pure and Applied Algebra **57**(2) (1989) 165–173

[18] Melliès, P.A.: Categorical semantics of linear logic. Panoramas et Syntheses **27** (2009) 15–215

[19] Pitts, A.M.: Categorical logic. In Abramsky, S., Gabbay, D.M., Maibaum, T.S.E., eds.: Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures. Oxford University Press (2000) 39–128

[20] Bierman, G.M.: On intuitionistic linear logic. PhD thesis, Citeseer (1994)

[21] Lawvere, F.W.: Equality in hyperdoctrines and comprehension schema as an adjoint functor. Applications of Categorical Algebra **17** (1970) 1–14

[22] Abramsky, S., Duncan, R.: A categorical quantum logic. Mathematical Structures in Computer Science **16**(3) (2006) 469–489 Preprint available at http://arxiv.org/abs/quant-ph/0512114.