Categorical algebras for linearity and dependency

Norihiro Yamada

yamad041@umn.edu
School of Mathematics
University of Minnesota

May 10, 2023

Abstract

On the one hand, *linear functions* arise in various fields of mathematics and beyond, e.g., linear algebra, functional analysis, representation theory, logic and quantum physics. Also, linear logic defines *linear proofs* and shows that linearity does not restrict but *refines* logical constructions by decomposing them into more primitive ones. On the other hand, predicates, or more generally *dependent types*, are indispensable part of the foundations of mathematics since without them one cannot even talk about properties of individual objects. It is then a natural aim to blend linearity and dependency in terms of the general framework of category theory because, in addition to clarifying how these two fundamental concepts interact, the combination will analyse and polish dependency through the lens of linearity and provide a mathematical universe to reason about linear functions and proofs. However, this blending is notoriously difficult to achieve, and a solution to this problem has not been established for a long time. The present work addresses this well-known problem via *modules* (for linearity) in the setting of *indexed categories* (for dependent types). Specifically, we introduce *module indexed categories* with *comprehension* as a categorical blending of linearity and dependency, and for their reasonability prove that they are sound and complete for a linear refinement of Martin-Löf type theory, a prominent foundation of mathematics. We also show that vector spaces form an example of this blending. This result reveals a module structure underlying dependent types and shows that linearity in algebra coincides with that in logic.

Contents

1 Introduction

1.1 Linearity and dependency

On the one hand, *linear functions* play key roles in many branches of mathematics and beyond, e.g., linear algebra, functional analysis, representation theory, logic and quantum physics. Their standard mathematical formalism is given via vector spaces, or more generally, via *modules*. Also, Girard's linear logic [Gir87] introduces *linear proofs* and refines existing logics by decomposing their logical constructions into more primitive ones. Categorically, the translation of intuitionistic logic into intuitionistic linear logic is characterised in terms of an adjunction [BBDPH93].

On the other hand, *predicates* constitute indispensable part of the foundations of mathematics by expressing properties of individual objects. In the framework of type theories [Chu40, SU06], a class of formal systems as well as functional programming languages, one may consider a natural generalisation of predicates known as *dependent types* [Hof97]. In terms of the standard, model-theoretic interpretation, a predicate is a relation or a boolean-valued function, while a dependent type is a set-indexed family of sets or an arbitrary function. In other words, a dependent type generalises a predicate by allowing a set with more than one element as its component.

Why does one care such a generalisation? First, it is a mathematically natural one because it corresponds to the path from boolean-valued maps to general maps. Second, recall that type theories serve as computational foundations of mathematics [ML75, ML84b, CH88] with applications to programming [CAB$^+$86, Sch09] via the *proofs-as-programs* paradigm [ML84a]. By accommodating *nontrivial* proofs beyond boolean values, dependent types play central roles in this computational paradigm. Lastly, this type-theoretic foundation has been extended significantly by Voevodsky et al. under the name of *homotopy type theory* and *univalent foundations* [Uni13]. This extension has created a very active field of research, forming a beautiful interface between type theory, homotopy theory and higher category theory. This new connection is based on the *higher-dimensional* structure underlying dependent types [HS98, vdBG11, Lum09, War11, AW09, KL21], but this structure becomes trivial if one focuses on predicates.

1.2 Our goal: a categorical blend of linearity and dependency

There are compelling reasons for aiming to combine linearity and dependency. First of all, it is an intriguing mathematical problem in its own right to blend these two fundamental concepts. As explained below, this problem poses a technical challenge. Second, the blend will advance an analysis on dependent types through the lens of linearity by decomposing operations on dependent types into more primitive ones. Also, such an analysis can be a step towards an extension of the type-theoretic foundations *constructively* to classical reasoning as linear logic uncovers a constructive interpretation of classical propositional logic [Gir87]. Such an extension will be an innovation as it is still poorly understood how to combine dependent types constructively with classical logic, e.g., see [Her05]; this problem is one of the main bottlenecks in restoring classical mathematics constructively. Last but not least, a blend of linearity and dependency will lead to a powerful foundation of mathematics that can reason about linear maps and proofs.

While logic and type theory motivate this work to a large extent, linearity and dependency are quite general concepts not specific to logic, type theory or even set theory. Our standpoint is that *category theory* [ML13] is general enough to formulate these ubiquitous concepts. Hence, we aim to blend linearity and dependency in terms of category theory, and show its reasonability concretely in relation to a type-theoretic blend of linear logic and dependent types.

1.3 Past attempts and obstacles

However, it is notoriously difficult to combine linear logic and dependent types, both categorically and type-theoretically, and it has been a long-standing problem in the fields. Indeed, this problem has been unsolved for nearly thirty years since the initial attempt [CP96] though some progresses have been made by various researchers [GL12, SHU13, PS12, Vák15, McB16, Atk18].

Why is the combination hard to accomplish? We shall answer this question in terms of both type theory and category theory. Let us first explain it in terms of type theory. Recall that the components of a type theory are a *context* Γ, a *(dependent) type* A over a context Γ, written $\Gamma \vdash A$ type, and a *term* a from a context Γ to a type A over Γ, written $\Gamma \vdash a : A$. From the logical point of view, a context Γ represents an *assumption*, a type A over Γ a *(generalised) predicate* over Γ, and a term a from Γ to A a *proof* of A under Γ. The model- or set-theoretic semantics $[\![\_]\!]$ interprets a context Γ by a set $[\![\Gamma]\!]$, a type $\Gamma \vdash A$ type by a family $[\![A]\!] = \{[\![A]\!]_\gamma\}_{\gamma \in [\![\Gamma]\!]}$ of sets $[\![A]\!]_\gamma$, and a term $\Gamma \vdash a : A$ by a map $[\![a]\!] : [\![\Gamma]\!] \to \bigcup_{\gamma \in [\![\Gamma]\!]} [\![A]\!]_\gamma$ such that $[\![a]\!](\gamma) \in [\![A]\!]_\gamma$ for all $\gamma \in [\![\Gamma]\!]$. A type is said to be *simple* if it does not contain a variable in the context. In the set-theoretic semantics, simple types are constant predicates or *propositions*. Every type S over the *empty context* $(\_)$ is simple as $(\_)$ does not contain any variable; we abbreviate $(\_) \vdash S$ type as S. For instance, there are a simple type N of natural numbers and a type $N \vdash \mathrm{List}_N$ type of finite lists of natural numbers. The set-theoretic semantics defines $[\![N]\!] := \mathbb{N}$ and $[\![\mathrm{List}_N]\!] := \{\mathbb{N}^k\}_{k \in \mathbb{N}}$.

Now, recall that linearity of a proof in the sense of linear logic is reflected in a type theory by the property of a term that it contains precisely one copy of each variable in the context. For instance, the term $x : N, y : N \vdash x + y : N$ is linear, but the one $x : N \vdash x + x : N$ is not. Then, the problem in blending linearity in this sense and dependent types is that a variable may occur not only in a term *but also in a dependent type*. Note that this problem is irrelevant to simple types, and simple type-theoretic formulations of linear proofs have been given (§1.3.1). For example, consider the following term (which is taken from [Atk18, §1]):

$$x : N, y : \mathrm{List}_N(x) \vdash y : \mathrm{List}_N(x). \tag{1}$$

This term y is not linear in the above sense as the variable x in the context does not occur in the term y, and also because x occurs twice in $\mathrm{List}_N(x)$. Intuitively, however, the computation of this term y is essentially the identity map, so one should be able to obtain its linear variant by removing x in the context. Nevertheless, this is impossible because x plays an indispensable role here as a parameter for $\mathrm{List}_N(x)$ though it is not a computational input for the term y.

Next, to depict the same problem from a categorical angle, recall that a standard categorical characterisation of dependent types is Jacobs' *comprehension categories* [Jac93] or any equivalent categories [Hof97, §3.2]. The set-theoretic semantics forms their instance. A comprehension category interprets dependent sum and product types, or generalised existential and universal quantifiers, respectively, by adjoints to the indexing functors induced by the first projections on (generalised) cartesian products. In type theory, the cartesian products correspond to contexts of the form $\Gamma, x : A$, and the projections to terms of the form $\Gamma, x : A \vdash x : A$. Therefore, the base category of a comprehension category has finite (generalised) products. However, projections *discard* inputs, and pairings *contract* them. Thus, the basic structure of a comprehension category is already incompatible with linearity in logic. Besides, as non-cartesian categories include those whose morphisms are a class of linear maps, e.g., the category of Hilbert spaces, the cartesian structure makes it hopeless for comprehension categories to embrace linearity in algebra too.

In the following, we list some attempts to address this problem in the literature of mathematics and computer science, and explain why they do not solve the problem completely.

1.3.1 Dual contexts

A major type-theoretic method adopted by Cervesato and Pfenning [CP96], Krishnaswami et al. [KPB15] and Vákàr [Vák15] for circumventing the conflict between linearity and dependency is to split a context into two regions, *cartesian* and *linear* ones, by a semicolon, and allow a type to contain only variables in the cartesian region. This *dual context* method is originally invented by Barber and Plotkin [BP96] on a simply-typed calculus for intuitionistic propositional linear logic. For instance, this approach transforms the term (1) into the form

$$x : N; y : \mathrm{List}_N(x) \vdash y : \mathrm{List}_N(x), \tag{2}$$

in which the left- (respectively, right-) hand side of the semicolon is the cartesian (respectively, linear) region. One can then apply the concept of linearity of a term to this dual context method by focusing on variables in the linear region. In this sense, the term (2) is linear.

In contrast, variables in the cartesian region are discardable and contractible as the reasoning about terms by a dependent type must be cartesian. For instance, the dependent type $f, g : N \multimap N, x : N \vdash f(f(x)) < g(x)$ type, where $\multimap$ represents the construction of linear maps, and $<$ the standard order between natural numbers, should a basic vocabulary in a linear dependent type theory, and it *duplicates* the variables f and x, respectively. I.e., dependent types must be cartesian for a type theory to be expressive even if it requires terms to be non-cartesian.

However, this approach does not realise a *true interaction* between linearity and dependency since the two regions are completely separated; i.e., it is only a *disjoint union* of linear logic and a dependent type theory. For instance, this method can never define a linear variant of dependent sum or product types because a type cannot vary over variables in the linear region. For the same reason, it cannot reason properly about linear maps either. Accordingly, the dual context system is hopeless with our aim to refine dependency by linearity or to extend the type-theoretic foundations of mathematics by reasoning about linear maps and proofs.

Unsurprisingly, the categorical counterpart of the dual context linear dependent type theory given by Vákàr [Vák15] is essentially the disjoint union of the existing categorical semantics of intuitionistic linear logic [BBDPH93] and dependent type theories [Jac93]. Needless to say, this approach does not depict how linearity interact with dependency in a genuine sense.

1.3.2 Quantitative type theories

Dual contexts were the best possible approaches to linear dependent type theories until McBride [McB16] made a breakthrough by adapting the *usage annotation* by semirings [BGMZ14, POM14, GS14] to a dependent type theory. The annotation is to assign an element n of a semiring to each variable in a context, which means that the variable is to be consumed precisely n times. The use of a semiring is suited here as type-theoretic constructions require addition and multiplication on annotated elements. McBride's innovative idea is then to regard variables used by a dependent type as *consumed 0-times*. For instance, his method transforms the term (1) into

$$x^0 : N, y^1 : \mathrm{List}_N(x^0) \vdash y^1 : \mathrm{List}_N(x^0), \tag{3}$$

where the superscripts on variables are elements of a fixed semiring. Note that the total number of consumed variables matches that of produced variables: $0 = 0 + 0$ for x, and $1 = 1$ for y. In this way, McBride accomplished a linear dependent type theory in which a type may depend on variables classified as those in the linear region in the dual context system (§1.3.1).

However, it has turned out that McBride's linear dependent type theory has a fundamental flaw due to its linear pi-types: not closed under substitution. This problem is fixed by Atkey [Atk18] by restricting the annotation on a term to 0 or 1. Unfortunately, this restriction prohibits terms from possessing quantitative information. This restriction also makes it hopeless to regard terms as linear maps because it bans addition or scalar multiplication on terms. Consequently, the linearity of a term in the sense of Atkey diverges from the standard one in linear algebra.

Finally, no categorical semantics of McBride's or Atkey's type theory has been established (though Atkey built equational semantics [Atk18, §3]). Thus, the problem of combining linearity and dependency in the general framework of category theory was still open.

1.4 Main results and our contributions

The present work realises a true blend of linearity and dependency in terms of type theory and category theory as follows. We first define the syntax of a linear variant of *Martin-Löf type theory (MLTT)* [ML82, ML84b, ML98] or one of the best-known dependent type theories, called *linear MLTT (LMLTT)*, and prove its basic properties. In particular, we show that LMLTT is closed under substitution, overcoming the deficiency of McBride, and moreover it gives rise to the categorical structure sketched below. LMLTT is also free from Atkey's undesirable restriction on the annotation of terms. Our key idea that makes this innovation is to remedy the previous versions of linear pi-types [McB16, Atk18] in a principled way compatible with the structure of modules (in the sense of linear algebra). Besides, by a generalisation of Girard's translation from intuitionistic linear logic to intuitionistic logic [Gir87], LMLTT recovers MLTT; i.e., LMLTT is a linear refinement of MLTT, where constructions in the former refines those in the latter.

Next, we introduce a categorical reformulation of modules over a semiring and equip them with a linear refinement of comprehension [Jac93]. We further define categorical constructions in this framework that correspond to type constructions in LMLTT. The resulting structure is called *module comprehension categories*. As a main result, we show that module comprehension categories yield sound and complete semantics of LMLTT; i.e., they achieve a categorical blend of linearity and dependency that matches the type-theoretic one. The soundness and the completeness also imply that our formulation of linearity in the sense of linear algebra (or modules) coincides with that in linear logic (or type theory).

Finally, we lift the linear/non-linear adjunction [BBDPH93], the categorical counterpart of Girard's translation, to an adjunction between module comprehension categories and comprehension categories. This adjunction verifies that our categorical framework refines the standard categorical semantics of dependent type theories.

To the best of our knowledge, the present work presents the first complete solution to the long-standing problem of combining linear logic and dependent types. One of the main innovations behind this result is that, while categorical semantics of MLTT relies on (generalised) cartesian projections in a crucial way, our categorical approach dispenses with them entirely. This novel method is partly inspired by the ideas given by McBride and Atkey.

2 Linear Martin-Löf type theory

This section presents a linear variant of Martin-Löf type theory (MLTT), called ***linear MLTT (LMTLL)***. We assume that the reader is familiar with the basics of MLTT; see the original articles [ML82, ML84b, ML98] by Martin-Löf or the tutorial [Hof97] by Hofmann for the details.

This section proceeds as follows. We first fix the general format or *judgements* of LMLTT in §2.1, and, following this format, present *contexts* in §2.2, *structural rules* in §2.3, and *context morphisms* in §2.4. These structures constitute the core of LMLTT in the sense that they are applicable regardless of postulated types. We proceed to specific type constructions in LMLTT in §2.5 and finally prove basic properties of LMLTT, e.g., LMLTT recovers MLTT, in §2.6.

For our aim, it is convenient to employ the following categorical formulation of *semirings*:

Definition 2.1 (semirings)**.** A ***semiring*** is a strict symmetric monoidal category $\mathcal{R} = (\mathcal{R}, +, 0)$ with precisely one object $\star_{\mathcal{R}}$ (abbreviated as $\star$) that satisfies the equations

$$(x + y) \times z = (x \times z) + (y \times z) \qquad x \times (y + z) = (x \times y) + (x \times z) \qquad x \times 0 = 0 = 0 \times x,$$

where $\times$ denotes the composition of morphisms, for all morphisms x, y and z in $\mathcal{R}$, and it is said to be ***commutative*** if it additionally satisfies the equation

$$x \times y = y \times x$$

for all morphisms x and y in $\mathcal{R}$. A ***(totally) ordered semiring*** is a semiring $\mathcal{R}$ enriched over the category of (totally) ordered sets and monotone functions.

Notation. Given a semiring $\mathcal{R} = (\mathcal{R}, +, 0)$, we write $\mathcal{R}_{+,0}$ for the strict symmetric monoidal category $(\mathcal{R}, +, 0)$ or ***additive structure***, and $\mathcal{R}_{\times,1}$ for the strict monoidal category $(\mathcal{R}, \times, 1)$ or ***multiplicative structure***, where $1 := \mathrm{id}_\star$. Abusing notation, we write $r \in \mathcal{R}$ if r is a morphism in $\mathcal{R}$, and pq for $p \times q$. If $\mathcal{R}$ is ordered, then $\leqslant_{\mathcal{R}}$ or $\leqslant$ denotes the order on the hom-set $\mathcal{R}(\star, \star)$.

Example 2.2. If one takes an arbitrary element $\star$ as the unique object, and natural numbers as morphisms, then the set ω of all natural numbers forms an ordered semiring under the standard ordering on ω, where addition $+$ and zero 0 constitute the additive structure, and multiplication $\times$ and one 1 the multiplicative structure.

Throughout the present article, we fix an arbitrary ordered commutative semiring $\mathcal{R}$.

2.1 Judgements

Recall that MLTT is a formal system similar to *natural deduction* [Gen35, TS00] except that vertices of a derivation (tree) are ***judgements***, not formulae. Specifically, MLTT consists of the following six judgements (followed by their intended meanings):

1. $\vdash \Gamma$ ctx (Γ is a *context*);

2. $\Gamma \vdash \mathsf{A}$ type (A is a *type* in the context Γ);

3. $\Gamma \vdash \mathsf{a} : \mathsf{A}$ (a is a *term* or *proof* of the type A in the context Γ);

4. $\vdash \Gamma = \Delta$ ctx (Γ and Δ are *(judgmentally) equal* contexts);

5. $\Gamma \vdash \mathsf{A} = \mathsf{A}'$ type (A and A' are *(judgmentally) equal* types in the context Γ);

6. $\Gamma \vdash \mathsf{a} = \mathsf{a}' : \mathsf{A}$ (a and a' are *(judgmentally) equal* terms of the type A in the context Γ),

where judgemental equality is distinguished from *propositional* equality. A type and a term are primitive concepts, while a context is a derived one: A context is a finite sequence $\mathsf{x}_1 : \mathsf{A}_1, \dots, \mathsf{x}_\mathsf{n} : \mathsf{A}_\mathsf{n}$ of pairs of a variable x_i and a type A_i, written $\mathsf{x}_\mathsf{i} : \mathsf{A}_\mathsf{i}$ ($1 \leqslant i \leqslant n$), such that the variables are pairwise distinct.

Similarly, LMLTT consists of these six judgements except that its contexts, types and terms are decorated with elements of the ordered commutative semiring $\mathcal{R}$ for reasoning about *resources*:

1. $\vdash \Gamma$ ctx ($\Gamma = \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}$ is a context);

2. $\Gamma^0 \vdash \mathsf{A}^p$ type (A^p is a type in the context Γ^0);

3. $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$ (a^s is a term of the type A^p in the context Γ);

4. $\vdash \Gamma = \Delta$ ctx (Γ and Δ are equal contexts);

5. $\Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q$ type (A^p and B^q are equal types in the context Γ^0);

6. $\Gamma \vdash \mathsf{a}_1^{s_1} = \mathsf{a}_2^{s_2} : \mathsf{A}^p$ ($\mathsf{a}_1^{s_1}$ and $\mathsf{a}_2^{s_2}$ are equal terms of the type A^p in the context Γ),

where $p, p_1, \dots, p_n, s, s_1, s_2, q \in \mathcal{R}$, and $\Gamma^q := \mathsf{x}_1^{p_1 q} : \mathsf{A}_1^{p_1 q}, \dots, \mathsf{x}_\mathsf{n}^{p_n q} : \mathsf{A}_\mathsf{n}^{p_n q}$. The context Γ^0 of each type A^p is decorated by 0 as in [McB16, Atk18] because it does not consume any resources. As we shall verify shortly, if $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$, then there is some $\tilde{\Gamma} \vdash \tilde{\mathsf{a}}^q : \tilde{\mathsf{A}}^q$ such that $\tilde{\Gamma} = \Gamma$, $\tilde{\mathsf{A}}^q = \mathsf{A}^p$ and $\tilde{\Gamma} \vdash \mathsf{a}^s = \tilde{\mathsf{a}}^q : \tilde{\mathsf{A}}^q$. We write $\Gamma^0 \vdash \mathsf{A}$ type for $\Gamma^0 \vdash \mathsf{A}^1$ type, and $\Gamma \vdash \mathsf{a} : \mathsf{A}^p$ for $\Gamma \vdash \mathsf{a}^1 : \mathsf{A}^p$.

Remark. Strictly speaking, judgements in LMLTT are identified up to the renaming of bound variables to avoid a *variable capture* [B$^+$84] as in the case of MLTT [Hof97, §2]. Accordingly, we henceforth rename bound variables suitably and identify judgements modulo the renaming.

Unlike Atkey [Atk18] $p \in \mathcal{R}$ in a term $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$ ranges over *any* element of $\mathcal{R}$, not only 0 or 1. Also, we shall show that unlike McBride [McB16] LMLTT is closed under substitution.

We have not introduced any axioms or rules; we have just taken an overview of the general *format* of LMLTT. In the rest of this section, we present the axioms and the rules in LMLTT.

2.2 Contexts

A ***context*** in LMLTT is a finite sequence $\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}$ of pairs of a variable x_i ($i \in \overline{n}$) decorated with an element p_i of $\mathcal{R}$ and a type $\mathsf{A}_\mathsf{i}^{p_i}$, written $\mathsf{x}_\mathsf{i}^{p_i} : \mathsf{A}_\mathsf{i}^{p_i}$, such that the variables are pairwise distinct. We write $(\_)$ for the ***empty context***, i.e., the empty sequence, and we usually omit it if it occurs on the left-hand side of the ***turnstile*** $\vdash$ in a judgement. In this case, we even omit the turnstile $\vdash$ in a judgement as well.

Formally, Figure 1 displays the axioms and the rules on contexts in LMLTT. They are almost the same as the axioms and the rules on contexts in MLTT; the only difference is the attachment

$$(\textsc{Ctx-Emp})\ \frac{}{\vdash (\_)\ \mathsf{ctx}} \qquad (\textsc{Ctx-Ext})\ \frac{\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}}{\vdash \Gamma, \mathsf{x}^p : \mathsf{A}^p\ \mathsf{ctx}}\ (\mathsf{x}\text{ does not occur in }\Gamma)$$

$$(\textsc{Ctx-ExtEq})\ \frac{\vdash \Gamma = \Delta\ \mathsf{ctx} \qquad \Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q\ \mathsf{type}}{\vdash \Gamma, \mathsf{x}^p : \mathsf{A}^p = \Delta, \mathsf{y}^q : \mathsf{B}^q\ \mathsf{ctx}}\ (\mathsf{x}\text{ does not occur in }\Gamma\text{, and }\mathsf{y}\text{ does not in }\Delta)$$

Figure 1: The axioms and the rules for contexts in LMLTT

of elements of $\mathcal{R}$. The axiom Ctx-Emp and the rule Ctx-Ext together define that contexts are the finite sequences of variable-type pairs as sketched above. The rule Ctx-ExtEq is a *congruence rule* since it states that the judgmental equality on contexts is preserved under the context extension by Ctx-Ext. Note that we have $\vdash (\_) = (\_)\ \mathsf{ctx}$ by Ctx-Emp and the rule Ctx-EqRefl in §2.3.

Convention. As Hofmann [Hof97] does, we omit the congruence rules for other constructions.

2.3 Structural rules

Next, we collect some standard rules applicable to all types, called ***structural rules***, in Figure 2. The rule Var formulates the reasonable postulate that one may copy an element $\mathsf{x}^p : \mathsf{A}^p$ in the

$$(\textsc{Var})\ \frac{\vdash \Gamma, \mathsf{x}^p : \mathsf{A}^p, \Phi\ \mathsf{ctx}}{\Gamma^0, \mathsf{x}^p : \mathsf{A}^p, \Phi^0 \vdash \mathsf{x}^p : \mathsf{A}^p} \qquad (\textsc{Ctx-EqRefl})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\vdash \Gamma = \Gamma\ \mathsf{ctx}} \qquad (\textsc{Ctx-EqSym})\ \frac{\vdash \Gamma = \Delta\ \mathsf{ctx}}{\vdash \Delta = \Gamma\ \mathsf{ctx}}$$

$$(\textsc{Ctx-EqTrans})\ \frac{\vdash \Gamma = \Delta\ \mathsf{ctx} \qquad \vdash \Delta = \Omega\ \mathsf{ctx}}{\vdash \Gamma = \Omega\ \mathsf{ctx}} \qquad (\textsc{Ty-EqRefl})\ \frac{\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}}{\Gamma^0 \vdash \mathsf{A}^p = \mathsf{A}^p\ \mathsf{type}}$$

$$(\textsc{Ty-EqSym})\ \frac{\Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q\ \mathsf{type}}{\Gamma^0 \vdash \mathsf{B}^q = \mathsf{A}^p\ \mathsf{type}} \qquad (\textsc{Ty-EqTrans})\ \frac{\Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q\ \mathsf{type} \qquad \Gamma^0 \vdash \mathsf{B}^q = \mathsf{C}^r\ \mathsf{type}}{\Gamma^0 \vdash \mathsf{A}^p = \mathsf{C}^r\ \mathsf{type}}$$

$$(\textsc{Tm-EqRefl})\ \frac{\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p}{\Gamma \vdash \mathsf{a}^s = \mathsf{a}^s : \mathsf{A}^p} \qquad (\textsc{Tm-EqSym})\ \frac{\Gamma \vdash \mathsf{a}_1^{s_1} = \mathsf{a}_2^{s_2} : \mathsf{A}^p}{\Gamma \vdash \mathsf{a}_2^{s_2} = \mathsf{a}_1^{s_1} : \mathsf{A}^p}$$

$$(\textsc{Tm-EqTrans})\ \frac{\Gamma \vdash \mathsf{a}_1^{s_1} = \mathsf{a}_2^{s_2} : \mathsf{A}^p \qquad \Gamma \vdash \mathsf{a}_2^{s_2} = \mathsf{a}_3^{s_3} : \mathsf{A}^p}{\Gamma \vdash \mathsf{a}_1^{s_1} = \mathsf{a}_3^{s_3} : \mathsf{A}^p}$$

$$(\textsc{Ty-Conv})\ \frac{\vdash \Gamma = \Delta\ \mathsf{ctx} \qquad \Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}}{\Delta^0 \vdash \mathsf{A}^p\ \mathsf{type}}$$

$$(\textsc{Tm-Conv})\ \frac{\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p \qquad \vdash \Gamma = \Delta\ \mathsf{ctx} \qquad \Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q\ \mathsf{type}}{\Delta \vdash \mathsf{a}^s : \mathsf{B}^q}$$

Figure 2: The structural rules in LMLTT

context and paste it to the right-hand side. The remaining part of the context is decorated with 0 as the term only consumes the one $\mathsf{x}^p : \mathsf{A}^p$. In contrast, the corresponding rule in MLTT permits the discard of resources. The other structural rules in LMLTT are the same as the corresponding ones in MLTT except the attachment of elements of $\mathcal{R}$: The next nine rules stipulate that each judgmental equality is an equivalence relation, and the last two rules formalise the assumption that judgements are to be preserved under the exchange of equal contexts and/or types.

2.4 Context morphisms

One of the main differences between MLTT and LMLTT lies in *context morphisms* [Hof97, §2.4]: Those in MLTT are cartesian, while those in LMLTT are monoidal and *non-cartesian* by taking into account the *quantitative* information in *resource-aware* computation.

Let $\mathsf{\Gamma} = \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}$ and $\Delta = \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m}$ be contexts. To represent the quantitative information, a context morphism $\phi : \Delta \to \mathsf{\Gamma}$ in LMLTT is of the form

$$\mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m} \vdash \mathsf{a}_1^{s_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{a}_\mathsf{n}^{s_n} : \mathsf{A}_\mathsf{n}^{p_n}$$

such that

$$\mathsf{y}_1^0 : \mathsf{B}_1^0, \dots, \mathsf{y}_\mathsf{m}^0 : \mathsf{B}_\mathsf{m}^0, \mathsf{x}_1^0 : \mathsf{A}_1^0, \dots, \mathsf{x}_{\mathsf{i}-1}^0 : \mathsf{A}_{\mathsf{i}-1}^0 \vdash \mathsf{A}_\mathsf{i}^{p_i} \ \mathsf{type}$$

$$\mathsf{y}_1^{q_{1,i}} : \mathsf{B}_1^{q_{1,i}}, \dots, \mathsf{y}_\mathsf{m}^{q_{m,i}} : \mathsf{B}_\mathsf{m}^{q_{m,i}} \vdash \mathsf{a}_\mathsf{i}^{s_i} : \mathsf{A}_\mathsf{i}\{\mathsf{a}_1/\mathsf{x}_1, \dots, \mathsf{a}_{\mathsf{i}-1}/\mathsf{x}_{\mathsf{i}-1}\}^{p_i}$$

$$q_{j,i}, q_j, p_i \in \mathcal{R} \qquad 1 \leqslant i \leqslant n \qquad 1 \leqslant j \leqslant m \qquad \sum_{i=1}^{n} q_{j,i} = q_j.$$

We identify context morphisms up to the componentwise equality between the component terms; this convention corresponds to a congruence rule, so we do not present it formally.

Further, given $r \in \mathcal{R}$, we define another context morphism $\phi^r : \Delta^r \to \mathsf{\Gamma}^r$ by

$$\phi^r := \mathsf{y}_1^{q_1 r} : \mathsf{B}_1^{q_1 r}, \dots, \mathsf{y}_\mathsf{m}^{q_m r} : \mathsf{B}_\mathsf{m}^{q_m r} \vdash \mathsf{a}_1^{s_1 r} : \mathsf{A}_1^{p_1 r}, \dots, \mathsf{a}_\mathsf{n}^{s_n r} : \mathsf{A}_\mathsf{n}^{p_n r},$$

which is well-defined as we shall show later.

Example 2.3. Each context $\mathsf{\Gamma} = \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}$ has the *identity context morphism*

$$(\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n} \vdash \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}) : \mathsf{\Gamma} \to \mathsf{\Gamma},$$

and together with a term $\mathsf{\Gamma} \vdash \mathsf{b}^t : \mathsf{B}^q$ it induces the context morphism

$$(\mathsf{x}_1^{2p_1} : \mathsf{A}_1^{2p_1}, \dots, \mathsf{x}_\mathsf{n}^{2p_n} : \mathsf{A}_\mathsf{n}^{2p_n} \vdash \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}, \mathsf{b}^t : \mathsf{B}^q) : \mathsf{\Gamma}^2 \to \mathsf{\Gamma}, \mathsf{y}^q : \mathsf{B}^q,$$

where $2 := 1 + 1$.

Given another context $\Delta = \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m}$, there is the *twisting context morphism*

$$(\mathsf{\Gamma}, \Delta \vdash \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m}, \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}) : \mathsf{\Gamma}, \Delta \to \Delta, \mathsf{\Gamma}.$$

Notation. If $\phi : \Delta \to \mathsf{\Gamma}$ is a context morphism $\mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m} \vdash \mathsf{a}_1^{s_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{a}_\mathsf{n}^{s_n} : \mathsf{A}_\mathsf{n}^{p_n}$, and $\mathsf{\Gamma}, \Omega \vdash \mathcal{J}$ is a judgement, then the expression

$$\Delta, \Omega\{\phi\} \vdash \mathcal{J}\{\phi\} \tag{4}$$

denotes the judgement obtained from $\mathsf{\Gamma}, \Omega \vdash \mathcal{J}$ by (simultaneously) substituting [Hof97, §2.4] a_i for x_i in Ω and in $\mathcal{J}$ for $i = 1, \dots, n$, which is well-defined as we shall prove shortly.

Another, much more superficial difference between MLTT and LMLTT is that, whilst context morphisms are an auxiliary, derived concept in MLTT, those in LMLTT are *formal objects* with formal rules (similarly to contexts, types and terms) for making the quantitative reasoning of LMLTT explicit. Specifically, LMLTT has the rules displayed in Figure 3 for context morphisms. The axiom CtxMor-Emp yields the empty context morphism. This axiom requires the domain of the morphism to be of the form $\mathsf{\Gamma}^0$ because the morphism should not consume any resources. In contrast, MLTT allows any context to be the domain of the empty context morphism, which admits the *discard* of resources. The rule Mor-Ext extends a context morphism by a term in a resource sensitive fashion. By the corresponding rule, MLTT permits the *copying* of resources.

Our context morphisms play a key role for the categorical structure of LMLTT and constitute one of the main differences between LMLTT and the type theory due to McBride or Atkey.

$$(\textsc{Mor-Emp})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma^0 \vdash (\_) : (\_)} \qquad (\textsc{Mor-Ext})\ \frac{\Delta^d \vdash \phi : \Gamma \qquad \Gamma^0 \vdash \mathsf{B}^q\ \mathsf{type} \qquad \Delta^e \vdash \mathsf{b}^t : \mathsf{B}\{\phi^0\}^q}{\Delta^{d+e} \vdash \phi : \Gamma, \mathsf{b}^t : \mathsf{B}^q}$$

Figure 3: The rules for context morphisms in LMLTT

2.5 Type constructions

In the following subsections, we present the axioms and the rules for specific type constructions in LMLTT. As in the case of MLTT, each type construction in LMLTT is defined in terms of ***formation***, ***introduction***, ***elimination*** and ***computation*** rules. The formation rule defines how to construct the type, and the introduction rule stipulates how to generate terms[1] of the type. The elimination and the computation rules describe how to consume the terms and the result of the consumption (in the form of an equation), respectively, both of which are justified by the introduction rule. One may further postulate an optional ***uniqueness*** rule, which imposes some *canonical form* on terms of the type.

Convention. Following a standard convention, henceforth we often omit *evident* judgements (in the sense that they are easily detected from other hypotheses) in the hypotheses of a rule. For instance, instead of presenting both $\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}$ and $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$, we do only the latter.

Notation. We write $\top$ and 1 for the units of tensor $\otimes$ and with &, respectively, i.e., we swap the traditional notations [Gir87] (similarly to [Tro91, §2.7]), because we find it more systematic.

2.5.1 Top-type

Let us first present the rules on the ***top-type*** $\top$ in Figure 4, As the name indicates, the top-type is a simple type essentially the same as the *multiplicative unit* in linear logic [Gir87]. Alternatively, it is a linear refinement of the *unit-type* in MLTT [Uni13, A.2.8], and therefore intuitively it is a trivially true formula with no information or computation involved.

$$(\top\textsc{-Form})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma^0 \vdash \top^p\ \mathsf{type}}\ (p \in \mathcal{R}) \qquad (\top\textsc{-Intro})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma^0 \vdash \star^p : \top^p}\ (p \in \mathcal{R})$$

$$(\top\textsc{-Elim})\ \frac{\Gamma^g \vdash \mathsf{e}^s : \top^p \qquad \Gamma^h \vdash \mathsf{a}^t : \mathsf{A}^q}{\Gamma^{(g+h)r} \vdash \mathsf{a}^t[\star^p \leftarrow \mathsf{e}^s]^r : \mathsf{A}^{qr}}\ (r \in \mathcal{R})$$

$$(\top\textsc{-Comp})\ \frac{\Gamma^0 \vdash \star^p : \top^p \qquad \Gamma^h \vdash \mathsf{a}^t : \mathsf{A}^q}{\Gamma^{hr} \vdash \mathsf{a}^t[\star^p \leftarrow \star^p]^r = \mathsf{a}^{tr} : \mathsf{A}^{qr}}\ (r \in \mathcal{R}) \qquad (\top\textsc{-Uniq})\ \frac{\Gamma^0 \vdash \mathsf{e}^s : \top^p}{\Gamma^0 \vdash \mathsf{e}^s = \star^p : \top^p}$$

Figure 4: The axioms and the rules for the top-type in LMLTT

The formation rule $\top$-Form constructs the top-type over any context with the zero quantity 0. Note that the top-type is *simple* since it does not contain any variables. The introduction rule $\top$-Intro generates the unique term $\star$ of the top-type, where the context has the zero quantity because $\star$ consumes no resources. The remaining rules are almost the same as the corresponding ones on the unit-type in MLTT except that they take care of the quantitative information.

[1] Strictly speaking, the introduction rule defines *canonical* terms of the type, which in turn defines terms of the type; see [ML84b, NPS90] for the details.

2.5.2 One-type

Let us next introduce the rules on the ***one-type*** 1 in Figure 5. Similarly to the top-type, the one-type is intended to be a trivially true formula. The difference between the two types is that the one-type is essentially the same as the *additive unit* in linear logic, so the introduction rule 1-Intro permits an arbitrary context Γ.

$$(1\text{-Form})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma^0 \vdash 1^p\ \mathsf{type}}\ (p \in \mathcal{R}) \qquad (1\text{-Intro})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma \vdash \bullet^p : 1^p}\ (p \in \mathcal{R}) \qquad (1\text{-Uniq})\ \frac{\Gamma \vdash \mathsf{o}^s : 1^p}{\Gamma \vdash \mathsf{o}^s = \bullet^p : 1^p}$$

Figure 5: The rules for the one-type in LMLTT

2.5.3 Bottom-type

We next present the rules on the ***bottom-type*** $\bot$ in Figure 6. The intuition is that the bottom-type is a false formula with no information or computation involved. The formation rule $\bot$-Form is just like that of the top- or the one-type, and the elimination rule $\bot$-Elim corresponds to *ex falso*, i.e., anything follows from a contradiction.

2.5.4 Theta-types

Now, let us introduce a generalisation Θ of *tensor* [Gir87] to dependent types, called ***dependent tensor*** or ***theta-types***. LMLTT has the rules on theta-types displayed in Figure 7.

The intuition is that theta-types represent a dependent conjunction similarly to sigma-types in MLTT. However, being a generalisation of tensor, theta-types are *multiplicative*, while sigma-types are additive. This intuition explains the rules of theta-types.

Notation. We write $\mathsf{A}^p \otimes \mathsf{B}^q$ for a theta-type $\Theta_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q$ if B does not contain the variable x. This convention makes sense because in this case the rules on the theta-type coincides with those on a tensor-type; we leave the details to the reader.

2.5.5 Sigma-types

LMLTT inherits the ***dependent sum*** or ***sigma-types*** Σ form MLTT, which is a dependent type generalisation of *additive conjunction* in linear logic. The rules for sigma-types are displayed in Figure 8. Because contexts in LMLTT are multiplicative and resource sensitive, the elimination rule Σ-Elim and the computation rule Σ-Comp are different from those in MLTT.

2.5.6 Lambda-types

We next introduce ***dependent linear implication*** or ***lambda-types*** Λ, whose rules are displayed in Figure 9. They are a linear refinement of *dependent products* or *pi-types* Π in MLTT.

$$(\bot\text{-Form})\ \frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma^0 \vdash \bot^p\ \mathsf{type}}\ (p \in \mathcal{R}) \qquad (\bot\text{-Elim})\ \frac{\Gamma \vdash \mathsf{b}^s : \bot^p \qquad \Delta^0, \mathsf{x}^0 : \bot^0 \vdash \mathsf{A}^q\ \mathsf{type}}{\Delta^0, \Gamma^0, \vdash \mathsf{R}^{\bot}_{\mathsf{A}}(\mathsf{b}^s)^r : \mathsf{A}\{\mathsf{b}/\mathsf{x}\}^{qr}}\ (r \in \mathcal{R})$$

Figure 6: The rules for the bottom-type in LMLTT

$$(\Theta\text{-Form})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type}}{\Gamma^0 \vdash (\Theta_{x^p : A^p} B^q)^r \text{ type}}\ (p, r \in \mathcal{R})$$

$$(\Theta\text{-Intro})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type} \qquad \Gamma^g \vdash a^s : A^p \qquad \Gamma^h \vdash b^t : B\{a/x\}^q}{\Gamma^{(g+h)r} \vdash (a^s, b^t)^r : (\Theta_{x^p : A^p} B^q)^r}\ (r \in \mathcal{R})$$

$$(\Theta\text{-Elim})\ \frac{\Gamma^0, z^0 : (\Theta_{x^p : A^p} B^q)^0 \vdash C \text{ type} \qquad \Gamma^g, x^{pr} : A^{pr}, y^{qr} : B^{qr} \vdash c^u : C\{(x^p, y^q)/z\}^l \qquad \Gamma^h \vdash t^v : (\Theta_{x^p : A^p} B^q)^r}{\Gamma^{(g+h)k} \vdash c^u[(x^p, y^q)^r \leftarrow t^v]^k : C\{t/z\}^{lk}}\ (k \in \mathcal{R})$$

$$(\Theta\text{-Comp})\ \frac{\Gamma^0, z^0 : (\Theta_{x^p : A^p} B^q)^0 \vdash C \text{ type} \qquad \Gamma^g, x^{pr} : A^{pr}, y^{qr} : B^{qr} \vdash c^u : C\{(x^p, y^q)/z\}^l \qquad \Gamma^h \vdash (a^s, b^t)^r : (\Theta_{x^p : A^p} B^q)^r}{\Gamma^{(g+h)k} \vdash c^u[(x^p, y^q)^r \leftarrow (a^s, b^t)^r]^k = c\{a/x\}\{b/y\}^{uk} : C\{(a^s, b^t)/z\}^{lk}}\ (k \in \mathcal{R})$$

Figure 7: The rules for theta-type in LMLTT

$$(\Sigma\text{-Form})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type}}{\Gamma^0 \vdash (\Sigma_{x^p : A^p} B^q)^r \text{ type}}\ (p, r \in \mathcal{R})$$

$$(\Sigma\text{-Intro})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type} \qquad \Gamma \vdash a^s : A^p \qquad \Gamma \vdash b^t : B\{a/x\}^q}{\Gamma^r \vdash \langle a^s, b^t \rangle^r : (\Sigma_{x^p : A^p} B^q)^r}\ (r \in \mathcal{R})$$

$$(\Sigma\text{-Elim})\ \frac{\Gamma \vdash c^u : (\Sigma_{x^p : A^p} B^q)^r}{\Gamma^k \vdash \pi_1(c^u)^k : A^{prk}}\ (k \in \mathcal{R}) \qquad (\Sigma\text{-Elim})\ \frac{\Gamma \vdash c^u : (\Sigma_{x^p : A^p} B^q)^r}{\Gamma^k \vdash \pi_2(c^u)^k : B\{\pi_1(c^u)/x\}^{qrk}}\ (k \in \mathcal{R})$$

$$(\Sigma\text{-Comp})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type} \qquad \Gamma \vdash a^s : A^p \qquad \Gamma \vdash b^t : B\{a/x\}^q}{\Gamma^r \vdash \pi_1(\langle a^s, b^t \rangle)^r = a^{sr} : A^{pr}}\ (r \in \mathcal{R})$$

$$(\Sigma\text{-Comp})\ \frac{\Gamma^0, x^0 : A^0 \vdash B^q \text{ type} \qquad \Gamma \vdash a^s : A^p \qquad \Gamma \vdash b^t : B\{a/x\}^q}{\Gamma^r \vdash \pi_2(\langle a^s, b^t \rangle)^r = b^{tr} : B\{a/x\}^{qr}}\ (r \in \mathcal{R})$$

$$(\Sigma\text{-Uniq})\ \frac{\Gamma \vdash c^u : (\Sigma_{x^p : A^p} B^q)^r}{\Gamma^k \vdash \langle \pi_1(c^u), \pi_2(c^u) \rangle^k = c^{uk} : (\Sigma_{x^p : A^p} B^q)^{rk}}\ (k \in \mathcal{R})$$

Figure 8: The rules for sigma-type in LMLTT

2.5.7 Exponential-types

As the last type construction in LMLTT, we introduce ***exponential-types*** !. They are essentially the same as *exponential* or *of-course* ! in linear logic, i.e., a countable iteration of tensor $\otimes$. We display the rules on exponential-types in Figure 10.

2.6 Meta-theoretic properties

This section collects basic properties of LMLTT. We utilise some of them in later sections.

Proposition 2.4 (simultaneous substitution)**.** *For a context morphism $\phi : \Gamma \to \Delta$ and a judgement $\Delta, \Omega \vdash \mathcal{J}$ derivable in LMLTT, the judgement $\Gamma, \Omega\{\phi\} \vdash \mathcal{J}\{\phi\}$ is derivable in LMLTT.*

Proof. By induction on the length of Δ (as in the case of MLTT [Hof97, Proposition 2.12]). $\square$

$$(\Lambda\text{-Form})\ \frac{\Gamma^0, \mathsf{x}^0 : \mathsf{A}^0 \vdash \mathsf{B}^q\ \mathsf{type}}{\Gamma^0 \vdash (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r\ \mathsf{type}}\ (p, r \in \mathcal{R}) \qquad (\Lambda\text{-Intro})\ \frac{\Gamma, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q}{\Gamma^r \vdash (\lambda \mathsf{x}^p.\, \mathsf{b}^t)^r : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r}\ (r \in \mathcal{R})$$

$$(\Lambda\text{-Elim})\ \frac{\Gamma^g \vdash \mathsf{f}^t : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r \qquad \Gamma^h \vdash \mathsf{a}^s : \mathsf{A}^p}{\Gamma^{(g+rh)k} \vdash \mathsf{App}(\mathsf{f}^t, \mathsf{a}^{rs})^k : \mathsf{B}\{\mathsf{a}/\mathsf{x}\}^{qrk}}\ (k \in \mathcal{R})$$

$$(\Lambda\text{-Comp})\ \frac{\Gamma^g, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q \qquad \Gamma^h \vdash \mathsf{a}^s : \mathsf{A}^p}{\Gamma^{(rg+rh)k} \vdash \mathsf{App}((\lambda \mathsf{x}^p.\, \mathsf{b}^t)^r, \mathsf{a}^{rs})^k = \mathsf{b}\{\mathsf{a}/\mathsf{x}\}^{trk} : \mathsf{B}\{\mathsf{a}/\mathsf{x}\}^{qrk}}\ (r, k \in \mathcal{R})$$

$$(\Lambda\text{-Uniq})\ \frac{\Gamma \vdash \mathsf{f}^t : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r}{\Gamma^k \vdash (\lambda \mathsf{x}^{pr}.\, \mathsf{App}(\mathsf{f}^t, \mathsf{x}^{pr}))^k = \mathsf{f}^{tk} : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^{rk}}\ (k \in \mathcal{R})$$

Figure 9: The rules for lambda-type in LMLTT

$$(!\text{-Form})\ \frac{\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}}{\Gamma^0 \vdash (!\mathsf{A}^p)^r\ \mathsf{type}}\ (r \in \mathcal{R})$$

$$(!\text{-Intro})\ \frac{\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p}{\Gamma^r \vdash (!\mathsf{a}^s)^r : (!\mathsf{A}^p)^r}\ (r \in \mathcal{R}\text{; all variables in } \mathsf{a} \text{ are of exponential types})$$

$$(!\text{-Elim})\ \frac{\Gamma \vdash \mathsf{e}^s : (!\mathsf{A}^p)^r \qquad \Delta, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q}{\Delta^{rk}, \Gamma^k \vdash \mathsf{b}^{rt}[(!\mathsf{x}^p)^r \leftarrow \mathsf{e}^s]^k : \mathsf{B}\{\mathsf{e}/\mathsf{x}\}^{rqk}}\ (k \in \mathcal{R})$$

$$(!\text{-Comp})\ \frac{\Gamma \vdash (!\mathsf{a}^s)^r : !\mathsf{A}^{pr} \qquad \Delta, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q}{\Delta^{rk}, \Gamma^k \vdash \mathsf{b}^{rt}[(!\mathsf{x}^p)^r \leftarrow (!\mathsf{a}^s)^r]^k = \mathsf{b}\{\mathsf{a}/\mathsf{x}\}^{rtk} : \mathsf{B}\{\mathsf{a}/\mathsf{x}\}^{rqk}}\ (k \in \mathcal{R})$$

Figure 10: The rules for exponential-type in LMLTT

Corollary 2.5 (weakening and substitution)**.** *The following rules are* admissible *in LMLTT:*

$$(\text{Weak})\ \frac{\Gamma, \Delta \vdash \mathcal{J} \qquad \Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}}{\Gamma, \mathsf{x}^0 : \mathsf{A}^0, \Delta \vdash \mathcal{J}}\ (\mathsf{x} \notin \Gamma, \Delta)$$

$$(\text{Subst})\ \frac{\Gamma, \mathsf{x}^p : \mathsf{A}^p, \Delta \vdash \mathcal{J} \qquad \Omega \vdash \mathsf{a}^p : \mathsf{A}^p}{\Gamma, \Omega, \Delta\{\mathsf{a}/\mathsf{x}\} \vdash \mathcal{J}\{\mathsf{a}/\mathsf{x}\}}\ (\mathsf{x} \notin \Gamma, \Omega)$$

where $\mathcal{J}$ is the right-hand side of an arbitrary judgement, and $\mathcal{J}\{\mathsf{a}/\mathsf{x}\}$ (respectively, $\Delta\{\mathsf{a}/\mathsf{x}\}$) denotes the (capture-free) substitution *of a for x in $\mathcal{J}$ (respectively, in Δ) as in [Hof97, §2].*

Proof. By Proposition 2.4. □

Proposition 2.6 (multiplication)**.** *Let r be an arbitrary element of the semiring $\mathcal{R}$.*

1. *If $\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{type}$ is derivable in LMLTT, then so is $\Gamma^0 \vdash \mathsf{A}^{pr}\ \mathsf{type}$;*
2. *If $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$ is derivable in LMLTT, then so is $\Gamma^r \vdash \mathsf{a}^{sr} : \mathsf{B}^{pr}$.*

Proof. By induction on derivations in LMLTT. □

Lemma 2.7 (canonical grading)**.** *If* $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p$ *is a term, then there is another one* $\tilde{\Gamma} \vdash \tilde{\mathsf{a}}^q : \tilde{\mathsf{A}}^q$ *such that* $\tilde{\Gamma} = \Gamma$*,* $\tilde{\mathsf{A}}^q = \mathsf{A}^p$ *and* $\tilde{\Gamma} \vdash \mathsf{a}^s = \tilde{\mathsf{a}}^q : \tilde{\mathsf{A}}^q$*. Therefore, if* $\phi = (\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \ldots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n} \vdash \mathsf{b}_1^{t_1} : \mathsf{B}_1^{q_1}, \ldots, \mathsf{b}_\mathsf{m}^{t_m} : \mathsf{B}_\mathsf{m}^{q_m})$ *is a context morphism* $\Gamma \to \Delta$*, then there is another context morphism* $\tilde{\phi} = (\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \ldots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n} \vdash \tilde{\mathsf{b}}_1^{r_1} : \tilde{\mathsf{B}}_1^{r_1}, \ldots, \tilde{\mathsf{b}}_\mathsf{m}^{r_m} : \tilde{\mathsf{B}}_\mathsf{m}^{r_m})$ *such that* $\phi = \tilde{\phi} : \Gamma \to \Delta$*.*

Proof. By induction on derivations of terms in LMLTT. □

Lastly, we define the *composition* of context morphisms as in [Hof97, S 2.4]:

Definition 2.8 (composition of context morphisms)**.** The ***composition*** of context morphisms $\phi : \Gamma \to \Delta$ and $\psi : \Delta \to \Xi$ is the context morphism $\psi \circ \phi : \Gamma \to \Xi$ defined by

$$\psi \circ \phi := \Gamma \vdash \mathsf{c}_1\{\phi\}^{u_1} : \mathsf{C}_1\{\phi\}^{r_1}, \ldots, \mathsf{c}_\mathsf{n}\{\phi\}^{u_n} : \mathsf{C}_\mathsf{n}\{\phi\}^{r_n},$$

where $\psi = \Delta \vdash \mathsf{c}_1^{u_1} : \mathsf{C}_1^{r_1}, \ldots, \mathsf{c}_\mathsf{n}^{u_n} : \mathsf{C}_\mathsf{n}^{r_n}$.

Proposition 2.9 (basic properties of composition of context morphisms)**.** *Given context morphisms* $\phi, \phi' : \Delta \to \Gamma$*,* $\psi, \psi' : \Gamma \to \Xi$ *and* $\varphi : \Xi \to \Omega$ *and a judgement* $\Xi \vdash \mathcal{J}$*, we have*

1. $\psi \circ \phi : \Delta \to \Xi$*;*
2. $\mathcal{J}\{\psi \circ \phi\} = \mathcal{J}\{\psi\}\{\phi\}$*;*
3. $\psi \circ \phi = \psi' \circ \phi'$ *if* $\phi = \phi'$ *and* $\psi = \psi'$*;*
4. $(\varphi \circ \psi) \circ \phi = \varphi \circ (\psi \circ \phi)$*;*
5. $\mathrm{id}_\Gamma \circ \phi = \phi = \phi \circ \mathrm{id}_\Delta$.

Proof. Straightforward and left to the reader. □

3 Categorical semantics of linear dependency

Having established the syntax of LMLTT, let us proceed to its categorical semantics.

3.1 Module comprehension categories

Our semantics of LMLTT is based on a categorical generalisation of a *module* over a semiring:

Definition 3.1 (module categories)**.** A ***module category*** over the semiring $\mathcal{R}$ consists of

- A symmetric monoidal category $\mathcal{M} = (\mathcal{M}, \otimes, I)$;
- A colax monoidal functor $(\_)^{-} = ((\_)^{-}, \omega, \delta) : \mathcal{R}(\star, \star) \to [\mathcal{M}, \mathcal{M}]$,[2] called the ***grading***, where $[\mathcal{M}, \mathcal{M}]$ is the strict monoidal category of strong symmetric monoidal endofunctors on $\mathcal{M}$ such that the diagrams in Figure 11 commute for all $\Gamma \in \mathcal{M}$ and $p, q, r \in \mathcal{R}$.

It is said to be ***closed*** if so is the underlying symmetric monoidal category.

Notation. Let $\mathcal{M}$ be a module category over the semiring $\mathcal{R}$, and $p \in \mathcal{R}$. We write $\mathcal{M}_p$ for the subcategory of $\mathcal{M}$ whose objects are of the form Γ^p such that $\Gamma \in \mathcal{M}$.

A useful intuition is that a module category $\mathcal{M}$ over the semiring $\mathcal{R}$ is a generalisation of the category Vec_K of vector spaces over a field K as well as the category Set of sets:

[2] Note that the underlying category of $\mathcal{R}(\star, \star)$ is discrete.

$$
\begin{array}{ccc}
\Gamma^0 & \xrightarrow{\omega} & I \\
\Big\| & & \Big\downarrow \cong \\
(\Gamma^0)^r & \xrightarrow[\omega^r]{} & I^r
\end{array}
\qquad\qquad
\begin{array}{ccc}
\Gamma^0 & \xrightarrow{\omega} & I \\
\Big\| & & \Big\| \\
(\Gamma^r)^0 & \xrightarrow[\omega]{} & I
\end{array}
$$

$$
\begin{array}{ccc}
(\Gamma^{p+q})^r & \xrightarrow{\delta^r_{p,q}} & (\Gamma^p \otimes \Gamma^q)^r \\
\Big\| & & \Big\downarrow \cong \\
\Gamma^{pr+qr} & \xrightarrow[\delta_{pr,qr}]{} & \Gamma^{pr} \otimes \Gamma^{qr}
\end{array}
\qquad\qquad
\begin{array}{ccc}
\Gamma^{rp+rq} & \xrightarrow{\delta_{rp,rq}} & \Gamma^{rp} \otimes \Gamma^{rq} \\
\Big\| & & \Big\| \\
(\Gamma^r)^{p+q} & \xrightarrow[\delta_{p,q}]{} & (\Gamma^r)^p \otimes (\Gamma^r)^q
\end{array}
$$

Figure 11: Commutative diagrams for a module category

Example 3.2. The category Vec_K gives rise to a module category over the semiring ω:

- The tensor $\otimes$ is the tensor product of vector spaces;
- The unit I is the trivial vector space;
- Given vector spaces $\Gamma, \Delta \in \mathrm{Vec}_K$, a linear function $\phi : \Delta \to \Gamma$ and a natural number $n \in \omega$, the vector space $\Gamma^n \in \mathrm{Vec}_K$ is defined by $\Gamma^n := \underbrace{\Gamma \otimes \Gamma \otimes \cdots \otimes \Gamma}_{n}$ $(n > 0)$ and $\Gamma^0 := I$, and the linear function $\phi^n : \Delta^n \to \Gamma^n$ by $\phi^n := \underbrace{\phi \otimes \phi \otimes \cdots \otimes \phi}_{n}$ $(n > 0)$ and $\phi^0 := \mathrm{id}_I$;
- The linear function $\Gamma^0 = I \to I$ for each $\Gamma \in \mathrm{Vec}_K$ is the identity linear map;
- The linear function $\Gamma^{p+q} \to \Gamma^p \otimes \Gamma^q$ for each $\Gamma \in \mathrm{Vec}_K$ maps

  $$v_1 \otimes v_2 \otimes \cdots \otimes v_{p+q} \mapsto (v_1 \otimes v_2 \otimes \cdots \otimes v_p) \otimes (v_{p+1} \otimes v_{p+2} \otimes \cdots \otimes v_{p+q})$$

  for all $v_i \in \Gamma$ $(i = 1, 2, \ldots, p+q)$.

Example 3.3. The category Set forms a module category over ω:

- The tensor $\times$ is the cartesian product of sets;
- The unit T is an arbitrarily fixed singleton set $\{\star\}$;
- Given sets $X, Y \in \mathrm{Set}$, a function $f : Y \to X$ and a natural number $n \in \omega$, the set $X^n \in \mathrm{Set}$ is given by $X^n := \underbrace{X \times X \times \cdots \times X}_{n}$ $(n > 0)$ and $X^0 := T$, and the function $f^n : Y^n \to X^n$ by $f^n := \underbrace{f \times f \times \cdots \times f}_{n}$ $(n > 0)$ and $f^0 := \mathrm{id}_T$;
- The function $X^0 = T \to T$ for each $X \in \mathrm{Set}$ is the identity map;
- The function $X^{p+q} \to X^p \times X^q$ for each $X \in \mathrm{Set}$ maps

  $$(x_1, x_2, \ldots, x_{p+q}) \mapsto ((x_1, x_2, \ldots, x_p), (x_{p+1}, x_{p+2}, \ldots, x_{p+q}))$$

  for all $x_i \in X$ $(i = 1, 2, \ldots, p+q)$.

Next, the following two examples describe the links between module categories and LMLTT, and they also explain why the monoidal functor of a module category is *colax*:

Example 3.4. LMLTT gives rise to a module category $\mathcal{T}$ over the semiring ω as follows. First, the underlying category $\mathcal{T}$ consists of

- Contexts modulo judgemental equality as objects, where we write $[\Gamma]$ for the equivalence class of each context Γ;
- Context morphisms between contexts Δ and Γ modulo judgemental equality as morphisms $[\Delta] \to [\Gamma]$, where we write $[\phi]$ for the equivalence class of each context morphism ϕ;
- The composition $[\Delta] \stackrel{[\phi]}{\to} [\Gamma] \stackrel{[\psi]}{\to} [\Xi]$ of morphisms given by $[\psi] \circ [\phi] := [\psi \circ \phi]$;
- The identity $\mathrm{id}_{[\Gamma]}$ on each object $[\Gamma] = [\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}]$ is the equivalence class of the context morphism

$$\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n} \vdash \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}.$$

By Proposition 2.9, these structures constitute a category $\mathcal{T}$. Abusing notation, we omit the bracket $[\_]$ on objects and morphisms. Similarly, by abuse of language, we call an equivalence class of a context (respectively, a context morphism) a context (respectively, a context morphism).

Next, we equip the category $\mathcal{T}$ with the structure of a module category over ω:

- The category $\mathcal{T}$ is strict symmetric monoidal, in which the tensor of contexts $\Gamma = \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}$ and $\Delta = \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m}$ is the concatenation Γ, Δ, the unit is the empty context $(\_)$, and the symmetry $\Gamma, \Delta \stackrel{\sim}{\to} \Delta, \Gamma$ is the context morphism

$$\mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n}, \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m} \vdash \mathsf{y}_1^{q_1} : \mathsf{B}_1^{q_1}, \dots, \mathsf{y}_\mathsf{m}^{q_m} : \mathsf{B}_\mathsf{m}^{q_m}, \mathsf{x}_1^{p_1} : \mathsf{A}_1^{p_1}, \dots, \mathsf{x}_\mathsf{n}^{p_n} : \mathsf{A}_\mathsf{n}^{p_n};$$

- The grading $(\_)^{(\text{-})} : \omega(\star, \star) \to [\mathcal{T}, \mathcal{T}]$ consists of the maps

$$r \mapsto (\Gamma \mapsto \Gamma^r) \qquad \mathrm{id}_r \mapsto (\Delta \stackrel{\phi}{\to} \Gamma \mapsto \Delta^r \stackrel{\phi^r}{\to} \Gamma^r),$$

  where $r \in \mathcal{R}$, and the (colax) context morphisms

$$(\Gamma^0 \vdash (\_) : (\_)) : \Gamma^0 \to (\_)$$

$$(\mathsf{x}_1^{p+q} : \mathsf{A}_1^{p+q}, \dots, \mathsf{x}_\mathsf{n}^{p+q} : \mathsf{A}_\mathsf{n}^{p+q} \vdash \mathsf{x}_1^{p} : \mathsf{A}_1^{p}, \dots, \mathsf{x}_\mathsf{n}^{p} : \mathsf{A}_\mathsf{n}^{p}, \mathsf{x}_1^{q} : \mathsf{A}_1^{q}, \dots, \mathsf{x}_\mathsf{n}^{q} : \mathsf{A}_\mathsf{n}^{p}) : \Gamma^{p+q} \to \Gamma^p, \Gamma^q$$

  given by the rules Mor-Emp and Mor-Ext, respectively. The *opposite* ones, $(\_) \to \Gamma^0$ and $\Gamma^p, \Gamma^q \to \Gamma^{p+q}$, in general do not exist. It is easy to see that the four diagrams in Figure 11 commute, and the induced functor $(\_)^r : \mathcal{T} \to \mathcal{T}$ is *strict* symmetric monoidal.

Example 3.5. For each $\Gamma \in \mathcal{T}$, there is the category $\mathrm{Ty}(\Gamma)$ that consists of

- Types $\Gamma^0 \vdash \mathsf{A}^p\ \mathsf{ctx}$ modulo judgemental equality as objects, where we abbreviate the types as A^p, and write $[\mathsf{A}^p]$ for their equivalence classes;
- Terms $\Gamma^u, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q$ for some $u \in \mathcal{R}$ modulo judgemental equality as morphisms $[\mathsf{A}^p] \to [\mathsf{B}^q]$, where we abbreviate them as b^t, and write $[\mathsf{b}^t]$ for their equivalence classes;

- The composition $[\mathsf{A}^p] \xrightarrow{[\mathsf{b}^t]} [\mathsf{B}^q] \xrightarrow{[\mathsf{c}^s]} [\mathsf{C}^r]$ of morphisms $[\Gamma^u, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q]$ and $[\Gamma^v, \mathsf{y}^q : \mathsf{B}^q \vdash \mathsf{c}^s : \mathsf{C}^r]$ defined by
$$[\mathsf{c}^s] \circ [\mathsf{b}^t] := [\Gamma^{u+v}, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{c}\{\mathsf{b}/\mathsf{y}\}^s : \mathsf{C}\{\mathsf{b}/\mathsf{y}\}^r];$$
- The identity $\mathrm{id}_{[\mathsf{A}^p]}$ on each object $[\mathsf{A}^p]$ defined by
$$\mathrm{id}_{[\mathsf{A}^p]} := [\Gamma^0, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{x}^p : \mathsf{A}^p].$$

As in the case of the category $\mathcal{T}$, these structures constitute a well-defined category $\mathrm{Ty}(\Gamma)$. Again, we omit the bracket $[\_]$ and confuse equivalence classes with their representatives.

This category is symmetric monoidal, where the tensor $\otimes : \mathrm{Ty}(\Gamma) \times \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma)$ maps

$$\big(\mathsf{A}^p, \mathsf{B}^q\big) \mapsto \mathsf{A}^p \otimes \mathsf{B}^q$$
$$\big((\Gamma^u, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{c}^s : \mathsf{C}^l), (\Gamma^v, \mathsf{y}^q : \mathsf{B}^q \vdash \mathsf{d}^t : \mathsf{D}^r)\big) \mapsto \Gamma^{u+v}, \mathsf{z} : \mathsf{A}^p \otimes \mathsf{B}^q \vdash (\mathsf{c}^s, \mathsf{d}^t)[(\mathsf{x}^p, \mathsf{y}^q) \leftarrow \mathsf{z}] : \mathsf{C}^l \otimes \mathsf{D}^r,$$

and the unit is (the equivalence class of) the top-type. We leave it as an exercise to construct the coherence natural isomorphisms for the associativity, the unit law and the symmetry.

Moreover, this symmetric monoidal category extends to a module category over the semiring ω, where the grading functor $(\_)^{-} : \omega(\star, \star) \to [\mathrm{Ty}(\Gamma), \mathrm{Ty}(\Gamma)]$ is given by the maps

$$(r, \mathsf{A}^p) \mapsto \mathsf{A}^{pr}$$
$$(\mathrm{id}_r, \Gamma^u, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^q : \mathsf{B}^q) \mapsto \Gamma^{ur}, \mathsf{x}^{pr} : \mathsf{A}^{pr} \vdash \mathsf{b}^{qr} : \mathsf{B}^{qr}$$

together with the evident (colax) morphisms

$$(\mathsf{A}^r)^0 = \mathsf{A}^0 \to \top \qquad (\mathsf{A}^r)^{p+q} = \mathsf{A}^{pr+qr} \to \mathsf{A}^{pr} \otimes \mathsf{A}^{qr} = (\mathsf{A}^r)^p \otimes (\mathsf{A}^r)^q.$$

given by weakening and the respective rules Θ-Intro and $\top$-Intro. These morphisms make the diagrams in Figure 11 commute; we focus on the third one since the other three are trivial: The composition of $\Gamma^0, \mathsf{z}^r : (\mathsf{A}^p \otimes \mathsf{A}^q)^r \vdash (\mathsf{x}^{pr}, \mathsf{y}^{qr})[(\mathsf{x}^{pr}, \mathsf{y}^{qr}) \leftarrow \mathsf{z}^r] : \mathsf{A}^{pr} \otimes \mathsf{A}^{qr}$ and $\Gamma^0, \mathsf{x}^{(p+q)r} : \mathsf{A}^{(p+q)r} \vdash (\mathsf{x}^p, \mathsf{x}^q)^r : (\mathsf{A}^p \otimes \mathsf{A}^q)^r$ is $\Gamma^0, \mathsf{x}^{(p+q)r} : \mathsf{A}^{(p+q)r} \vdash (\mathsf{x}^{pr}, \mathsf{y}^{qr})[(\mathsf{x}^{pr}, \mathsf{y}^{qr}) \leftarrow (\mathsf{x}^{pr}, \mathsf{y}^{qr})^r] : \mathsf{A}^{pr} \otimes \mathsf{A}^{qr}$, which is equal to $\Gamma^0, \mathsf{x}^{(p+q)r} : \mathsf{A}^{(p+q)r} \vdash (\mathsf{x}^{pr}, \mathsf{y}^{qr}) : \mathsf{A}^{pr} \otimes \mathsf{A}^{qr}$.

We have shown that $\mathrm{Ty}(\Gamma)$ forms a module category over ω. Note that, by construction, we have the equation

$$\mathrm{Ty}(\Gamma) = \mathrm{Ty}(\Gamma^0). \tag{5}$$

We next generalise module categories along the path from simple to dependent type theories (or from propositional to predicate logic). To this end, we need a 2-category of module categories:

Definition 3.6 (module functors)**.** A ***module functor*** over a sermiring $\mathcal{R}$ is a strong symmetric monoidal functor $F : \mathcal{M} \to \mathcal{M}'$ between module categories over $\mathcal{R}$ together with a family of mediating isomorphisms $[\Gamma]^r_F : F(\Gamma^r) \xrightarrow{\sim} F(\Gamma)^r$ for all $r \in \mathcal{R}$ natural in $\Gamma \in \mathcal{M}$.

Definition 3.7 (module natural transformations)**.** A ***module natural transformation*** over the semiring $\mathcal{R}$ is a symmetric monoidal natural transformation $\alpha : F \Rightarrow G : \mathcal{M} \to \mathcal{M}'$ between module functors over $\mathcal{R}$ such that the diagram

$$\begin{array}{ccc} F(\Gamma^r) & \xrightarrow{[\Gamma]^r_F} & F(\Gamma)^r \\ {\scriptstyle \alpha_{\Gamma^r}}\downarrow & & \downarrow{\scriptstyle \alpha^r_\Gamma} \\ G(\Gamma^r) & \xrightarrow[{[\Gamma]^r_G}]{} & G(\Gamma)^r \end{array}$$

commutes for all $\Gamma \in \mathcal{M}$ and $r \in \mathcal{R}$.

Proposition 3.8 (2-category of module categories)**.** *(Small) module categories, module functors and module natural transformations over the semiring $\mathcal{R}$ constitute a (large) 2-category* $\mathrm{ModCat}_{\mathcal{R}}$*.*

Proof. First, given small module categories $\mathcal{M} = (\mathcal{M}, \otimes, I)$ and $\mathcal{M}' = (\mathcal{M}', \otimes', I')$ over $\mathcal{R}$, let $\mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M}')$ be the category such that

- Objects are module functors $\mathcal{M} \to \mathcal{M}'$;
- Morphisms between objects $F, G : \mathcal{M} \to \mathcal{M}'$ are module natural transformations $F \Rightarrow G$;
- The composition of morphisms $\alpha : F \Rightarrow G$ and $\beta : G \Rightarrow H$ is the vertical composition of natural transformations
$$F \overset{\alpha}{\Rightarrow} G \overset{\beta}{\Rightarrow} H : \mathcal{M} \to \mathcal{M}'$$
- The identity on each object $F : \mathcal{M} \to \mathcal{M}'$ is the identity natural transformation $F \Rightarrow F$.

Next, given another small module category $\mathcal{M}'' = (\mathcal{M}'', \otimes'', I'')$ over $\mathcal{R}$, we define the bifunctor $\mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M}') \times \mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}', \mathcal{M}'') \to \mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M}'')$ that maps

- Each pair of objects $F \in \mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M}')$ and $F' \in \mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}', \mathcal{M}'')$ to the composition $F' \circ F$ of strong symmetric monoidal functors
$$\mathcal{M} \overset{F}{\to} \mathcal{M}' \overset{F'}{\to} \mathcal{M}'' \qquad\qquad (F' \circ F)_I := \left(I'' \overset{F'_{I'}}{\longrightarrow} F'I' \overset{F'F_I}{\longrightarrow} F'FI\right)$$
$$(F' \circ F)_{A,B} := \left(F'FA \otimes'' F'FB \overset{F'_{FA,FB}}{\longrightarrow} F'(FA \otimes' FB) \overset{F'F_{A,B}}{\longrightarrow} F'F(A \otimes B)\right)$$
together with the composition of mediating isomorphisms
$$(F'F\Gamma)^r \overset{[F\Gamma]^r_{F'}}{\longrightarrow} F'(F\Gamma)^r \overset{F'[\Gamma]^r_F}{\longrightarrow} F'F(\Gamma^r)$$
for each $\Gamma \in \mathcal{M}$ and $r \in \mathcal{R}$;
- Each pair of morphisms $\alpha : F \to G$ in $\mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M}')$ and $\alpha' : F' \to G'$ in $\mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}', \mathcal{M}'')$ to the horizontal composition of natural transformations
$$\alpha' * \alpha : F' \circ F \to G' \circ G.$$

Next, let the functor $\mathrm{id}_{\mathcal{M}} : 1 \to \mathrm{ModCat}_{\mathcal{R}}(\mathcal{M}, \mathcal{M})$ map the unique object of the terminal category 1 to the identity functor $\mathrm{id}_{\mathcal{M}} : \mathcal{M} \to \mathcal{M}$ together with the family of trivial mediating isomorphisms, and the unique morphism in 1 to the identity natural transformation on $\mathrm{id}_{\mathcal{M}}$.

Finally, it is just a routine to verify that these structures constitute a well-defined 2-category as in the case of the 2-category of symmetric monoidal categories. □

We are now ready to generalise module categories and define our basic categorical structure to interpret LMLTT:

Definition 3.9 (module comprehension categories)**.** A ***module pre-comprehension category*** over the semiring $\mathcal{R}$ is a triple $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ of

- A module category $\mathcal{B} = (\mathcal{B}, \otimes, I, (\_)^{\_})$ over $\mathcal{R}$, called the ***base***;

- A functor $\mathscr{D} : \mathcal{B}^{\mathrm{op}} \to \mathrm{ModCat}_{\mathcal{R}}$, called the ***dependency***, such that the diagram

$$\begin{array}{ccccc} \mathcal{B}^{\mathrm{op}} & & \xrightarrow{\quad (\_)^0 \quad} & & \mathcal{B}_0^{\mathrm{op}} \\ {\scriptstyle \mathscr{D}}\downarrow & & & & \downarrow{\scriptstyle \mathscr{D}} \\ \mathrm{ModCat}_{\mathcal{R}} & \xrightarrow[\mathrm{Ob}]{} & \mathrm{Set} & \xleftarrow[\mathrm{Ob}]{} & \mathrm{ModCat}_{\mathcal{R}} \end{array}$$

  in the category Cat commutes, where the functor $\mathrm{Ob} : \mathrm{ModCat}_{\mathcal{R}} \to \mathrm{Set}$ maps each module category $A \in \mathrm{ModCat}_{\mathcal{R}}$ to the set $\mathrm{Ob}(A)$ of all its objects, and each module functor $F : A \to B$ to its object-map $\mathrm{Ob}(f) : \mathrm{Ob}(A) \to \mathrm{Ob}(B)$;

  Notation. We define a functor $\mathscr{D}^+_\Gamma : \mathscr{D}(\Gamma)^{\mathrm{Op}} \to \mathrm{ModCat}_{\mathcal{R}}$ for each $\Gamma \in \mathcal{B}$ by $\mathscr{D}^+_\Gamma := \mathscr{D} \circ (\Gamma.\_)$.

- A functor $\_.\_ : \int_{\mathcal{B}} \mathscr{D}_0 \to \mathcal{B}$, called the ***module pre-comprehension***, where $\int_{\mathcal{B}} \mathscr{D}_0$ denotes the Grothendieck construction for $\mathscr{D}_0 := \mathscr{D} \circ (\_)^0 : \mathcal{B}^{\mathrm{Op}} \to \mathrm{ModCat}_{\mathcal{R}}$,

that satisfies the following axioms:

- (Left unit law) The functor $A \in \mathscr{D}(I) \mapsto I.A \in \mathcal{B}$ defines an equivalence $\mathscr{D}(I) \simeq \mathcal{B}$ in $\mathrm{ModCat}_{\mathcal{R}}$, for which the mapping $\Gamma \mapsto \underline{\Gamma}$ denotes the inverse, and an isomorphism $\Delta \otimes \Gamma \cong \Delta.\underline{\Gamma}\{\omega\}$ natural in $\Gamma, \Delta \in \mathcal{B}$;

- (Right unit law) The unit $\top_\Gamma$ of the module category $\mathscr{D}(\Gamma) = (\mathscr{D}(\Gamma), \otimes_\Gamma, \top_\Gamma, (\_)^{\cdot}_{\Gamma})$ for each $\Gamma \in \mathcal{B}$ admits an isomorphism $\Gamma.\top_\Gamma \cong \Gamma$;

- (Associativity) There is a natural transformation $\Theta : \int \mathscr{D}^+ \Rightarrow \mathscr{D} : \mathcal{B}^{\mathrm{Op}} \to \mathrm{Cat}$ that admits an isomorphism $\Gamma.\Theta_\Gamma(A, B) \cong \Gamma.A.B$ natural in $\Gamma \in \mathcal{B}$, $A \in \mathscr{D}(\Gamma)$ and $B \in \mathscr{D}(\Gamma.A)$, where the functor $\int \mathscr{D}^+ : \mathcal{B}^{\mathrm{Op}} \to \mathrm{Cat}$ maps each object $\Gamma \in \mathcal{B}$ to the Grothendieck construction $\int \mathscr{D}^+_\Gamma$, and each morphism $\phi : \Delta \to \Gamma$ in $\mathcal{B}$ to the functor $\int \mathscr{D}^+_\phi := (\_\{\phi\}, \_\{\phi^+\}) : \int \mathscr{D}^+_\Gamma \to \int \mathscr{D}^+_\Delta$, and $\phi^+ := \phi.\mathrm{id} : \Delta.A\{\phi\} \to \Gamma.A$.

The module pre-comprehension is called a ***module comprehension*** if there are bijections

$$\mathscr{D}(I)(\Theta_I(\underline{\Gamma}, A\{\cong\}), A'\{\cong\}) \cong \mathscr{D}(\Gamma)(A, A') \cong \mathscr{D}(\Gamma.A)(\top, A'\{\pi_1^A \circ \cong\}) \tag{6}$$

natural in $\Gamma \in \mathcal{B}$ and $A, A' \in \mathscr{D}(\Gamma)$, where the morphism $\pi_1^A : \Gamma.A^0 \to \Gamma$ in $\mathcal{B}$ is the composition

$$\Gamma.A^0 \overset{\mathrm{id}.\omega}{\to} \Gamma.\top_\Gamma \overset{\sim}{\to} \Gamma.$$

A ***module comprehension category*** over $\mathcal{R}$ is a module pre-comprehension category over $\mathcal{R}$ whose module pre-comprehension is a module comprehension.

Remark. $\mathrm{ModCat}_{\mathcal{R}}$ is not a module category, and the dependency or the module comprehension of a module comprehension category is in general not a module functor. These structures seem impossible to obtain. For instance, there appears no evident choice for the grading on $\mathrm{ModCat}_{\mathcal{R}}$. Also, an obvious choice for the tensor on the result of applying a module pre-comprehension is the componentwise one, but clearly it does not work.

The intuition behind this definition is best provided by the following canonical example:

Example 3.10. If we postulate top- and theta-types, then the module category $\mathcal{T}$ over ω extends to a module comprehension category, which satisfies the left unit law on the nose, as follows:

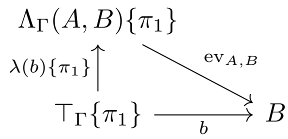
- The dependency $\mathrm{Ty} : \mathcal{T}^{\mathrm{Op}} \to \mathrm{ModCat}_\omega$ maps each context Γ to the module category $\mathrm{Ty}(\Gamma)$ defined in Example 3.5, and each context morphism $\phi : \Delta \to \Gamma$ to the module functor $\mathrm{Ty}(\phi) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Delta)$ that maps each type $\Gamma^0 \vdash \mathsf{A}^p$ type to the type $\Delta^0 \vdash \mathsf{A}\{\phi\}^p$ type, and each term $\Gamma^u, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q$ to the term $\Delta^u, \mathsf{x}^p : \mathsf{A}\{\phi\}^p \vdash \mathsf{b}\{\phi^u\}^t : \mathsf{B}\{\phi\}^q$;
- The module comprehension maps each pair (Γ, A^p) of a context Γ and a type $\Gamma^0 \vdash \mathsf{A}^p$ type to the context $\Gamma, \mathsf{x}^p : \mathsf{A}^p$, and a pair (ϕ, a^p) of a context morphism $\phi : \Delta \to \Gamma$ and a term $\Delta^v, \mathsf{y}^q : \mathsf{B}^q \vdash \mathsf{a}^s : \mathsf{A}\{\phi\}^p$ to the context morphism $\phi.\mathsf{a}^s : \Delta^{v+1}, \mathsf{y}^q : \mathsf{B}^q \to \Gamma, \mathsf{x}^p : \mathsf{A}^p$;
- The right-unit law is satisfied by the top-type;
- The family of all sigma-types $\Gamma^0 \vdash \Theta(\mathsf{A}^p, \mathsf{B}^q)^r$ type forms theta.

3.2 Semantic type constructions

We have seen that the right unit law and the associativity of a module comprehension category correspond in LMLTT to the top-type and theta-types, respectively. In this section, we define categorical semantics of the remaining type constructions.

Convention. We henceforth skip writing evident isomorphisms such as those in (6).

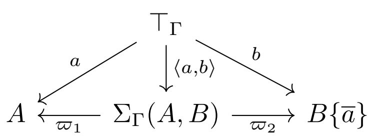
Definition 3.11 (lambda)**.** A module comprehension category $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over a semiring $\mathcal{R}$ is said to be ***closed*** if it possesses a family $\Lambda_\Gamma = \{\Lambda_\Gamma(A, B)\}_{\Gamma \in \mathcal{B}, A \in \mathscr{D}(\Gamma), B \in \mathscr{D}(\Gamma.A)}$, called ***lambda***, of objects $\Lambda_\Gamma(A, B) \in \mathscr{D}(\Gamma)$ stable under indexing together with morphisms $\mathrm{ev}_{A,B} \in \mathscr{D}(\Gamma.A)(\Lambda_\Gamma(A, B)\{\pi_1\}, B)$ that has a unique morphism $\lambda(b) \in \mathscr{D}(\Gamma)(\top, \Lambda_\Gamma(A, B))$ for a given one $b \in \mathscr{D}(\Gamma.A)(\top, B)$ such that the diagram



in $\mathscr{D}(\Gamma.A)$ commutes.

Example 3.12. For each context $\Gamma \in \mathcal{T}$, the family of all lambda-types $\Gamma^0 \vdash \Lambda(\mathsf{A}^p, \mathsf{B}^q)^r$ type over Γ forms lambda for the module comprehension category $\mathcal{T}$.

Definition 3.13 (sigma)**.** ***Sigma*** for a module comprehension category $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over a semiring $\mathcal{R}$ is a family $\Sigma_\Gamma = \{\Sigma_\Gamma(A, B)\}_{\Gamma \in \mathcal{B}, A \in \mathscr{D}(\Gamma), B \in \mathscr{D}(\Gamma.A)}$ of objects $\Sigma_\Gamma(A, B) \in \mathscr{D}(\Gamma)$ stable under indexing together with a pair of morphisms $\varpi_1 \in \mathscr{D}(\Gamma)(\Sigma_\Gamma(A, B), A)$ and $\varpi_2 \in \mathscr{D}(\Gamma)(\Sigma_\Gamma(A, B), B\{\overline{\varpi_1}\})$, where $\overline{\varpi_1} := \mathrm{id}.\varpi_1 : \Gamma^0.\Sigma_\Gamma(A, B) \to \Gamma^0.A$, that has a unique morphism $\langle a, b \rangle \in \mathscr{D}(\Gamma)(\top_\Gamma, \Sigma_\Gamma(A, B))$ for a given pair of morphisms $a \in \mathscr{D}(\Gamma)(\top_\Gamma, A)$ and $b \in \mathscr{D}(\Gamma)(\top_\Gamma, B\{\overline{a}\})$ such that the diagram



in $\mathscr{D}(\Gamma)$ commutes.

Example 3.14. For each context $\Gamma \in \mathcal{T}$, the family of all sigma-types $\Gamma^0 \vdash \Sigma(\mathsf{A}^p, \mathsf{B}^q)^r$ type over Γ forms sigma for the module comprehension category $\mathcal{T}$.

Definition 3.15 (one)**.** ***One*** for a module comprehension category $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over a semiring $\mathcal{R}$ is a family $1 = \{1_\Gamma\}_{\Gamma \in \mathcal{B}}$ of objects $1_\Gamma \in \mathscr{D}(\Gamma)$ together with a unique morphism $o_\Gamma \in \mathscr{D}(\Gamma)(\top, 1_\Gamma)$ for each $\Gamma \in \mathcal{B}$ such that $\Sigma_\Gamma(1_\Gamma, A\{\bar{o}\}) \cong A \cong \Sigma_\Gamma(A, 1_{\Gamma.A})$ for all $A \in \mathscr{D}(\Gamma)$.

Example 3.16. For each context $\Gamma \in \mathcal{T}$, the family of all one-types $\Gamma^0 \vdash 1 \ \mathsf{type}$ over Γ forms one for the module comprehension category $\mathcal{T}$.

Definition 3.17 (exponential)**.** ***Exponential*** for a module comprehension $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over an ordered semiring $\mathcal{R}$ is a limit $! : \mathcal{B} \to \mathcal{B}$ of the grading $(\_)^{\text{-}} : \mathcal{R}(\star, \star) \to [\mathcal{B}, \mathcal{B}]$.

Example 3.18. The function that maps types by $\mathsf{A}^p \mapsto !\mathsf{A}^p$ and terms by $\Gamma \vdash \mathsf{a}^s : \mathsf{A}^p \mapsto !\Gamma \vdash !\mathsf{a}^s : !\mathsf{A}^p$ forms exponential for the module comprehension category $\mathcal{T}$.

Definition 3.19 (bottom)**.** ***Bottom*** for a module comprehension $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over an ordered semiring $\mathcal{R}$ is a family $\bot = \{\bot_\Gamma\}_{\Gamma \in \mathcal{B}}$ of objects $\bot_\Gamma \in \mathscr{D}(\bot)$ together with morphisms $\xi_{\Gamma,A} \in \mathscr{D}(\Gamma^0)(\bot_\Gamma, A)$ for all $A \in \mathscr{D}(\Gamma.\bot_\Gamma)$.

Theorem 3.20 (completeness of module comprehension categories)**.** *LMLTT forms a module comprehension category $\mathcal{T}$ over the semiring ω, and top-types, theta-types, pi-types, sigma-types, one-types and exponential-types form top, theta, pi, sigma, one and exponential, respectively.*

Theorem 3.21 (dependent linear/non-linear adjunction)**.** *Under construction.*

4 Categorical semantics of linear Martin-Löf type theory

In this last section, we establish categorical semantics of LMLTT in an arbitrary module comprehension category (§4.1) and prove its soundness (§4.2).

4.1 Interpretation

Throughout this section, fix an ordered semiring $\mathcal{R}$ underlying LMLTT and a module comprehension category $\mathcal{B} = (\mathcal{B}, \mathscr{D}, \_.\_)$ over $\mathcal{R}$. Our aim is to define a *semantic map* $[\![\_]\!]_\mathcal{B}$ that interprets LMLTT in $\mathcal{B}$. Roughly, the semantic map $[\![\_]\!]_\mathcal{B}$ interprets judgements in LMLTT by

$$\begin{aligned} \vdash \Gamma \ \mathsf{ctx} &\mapsto [\![\Gamma]\!]_\mathcal{B} \in \mathcal{B} \\ \Gamma^0 \vdash \mathsf{A}^p \ \mathsf{type} &\mapsto [\![\mathsf{A}]\!]^p_\mathcal{B} \in \mathscr{D}([\![\Gamma]\!]_\mathcal{B}) \\ \Gamma \vdash \mathsf{a}^s : \mathsf{A}^p &\mapsto [\![\mathsf{a}]\!]^s_\mathcal{B} \in \mathscr{D}([\![\Gamma]\!]_\mathcal{B})(\top, [\![\mathsf{A}]\!]^p_\mathcal{B}) \\ \vdash \Gamma = \Delta \ \mathsf{ctx} &\Rightarrow [\![\Gamma]\!]_\mathcal{B} = [\![\Delta]\!]_\mathcal{B} \in \mathcal{B} \\ \Gamma^0 \vdash \mathsf{A}^p = \mathsf{B}^q \ \mathsf{type} &\Rightarrow [\![\mathsf{A}]\!]^p_\mathcal{B} = [\![\mathsf{B}]\!]^q_\mathcal{B} \in \mathscr{D}([\![\Gamma]\!]_\mathcal{B}) \\ \Gamma \vdash \mathsf{a}_1^{s_1} = \mathsf{a}_2^{s_2} : \mathsf{A}^p &\Rightarrow [\![\mathsf{a}_1]\!]^{s_1}_\mathcal{B} = [\![\mathsf{a}_2]\!]^{s_2}_\mathcal{B} \in \mathscr{D}([\![\Gamma]\!]_\mathcal{B})(\top, [\![\mathsf{A}]\!]^p_\mathcal{B}). \end{aligned}$$

As in the case of the interpretation of MLTT in a CwF [Hof97, §3.5], a priori we cannot define the semantic map $[\![\_]\!]_\mathcal{B}$ for judgements in LMLTT by induction on derivations as a derivation of a judgement may not be unique in the presence of the rules Ty-Con and Tm-Con. Nevertheless, let us first define the semantic map $[\![\_]\!]_\mathcal{B}$ by induction on derivations and later in Section 4.2 show that it is well-defined and in particular invariant over derivations of a judgement.

Definition 4.1 (semantics of contexts)**.** The semantic map $[\![\_]\!]_\mathcal{B}$ assigns an object $[\![\Gamma]\!]_\mathcal{B} \in \mathcal{B}$ to each context $\vdash \Gamma \ \mathsf{ctx}$ by induction on $|\Gamma|$,

$$[\![(\_)]\!]_\mathcal{B} := I \qquad\qquad [\![\Gamma, \mathsf{x}^p : \mathsf{A}^p]\!]_\mathcal{B} := [\![\Gamma]\!]_\mathcal{B}.[\![\mathsf{A}]\!]^p_\mathcal{B}.$$

Definition 4.2 (semantics of context morphisms)**.** The semantic map $[\![\_]\!]_{\mathcal{B}}$ assigns a morphism $[\![\phi]\!]_{\mathcal{B}} : [\![\Delta]\!]_{\mathcal{B}} \to [\![\Gamma]\!]_{\mathcal{B}}$ to each context morphism $\Delta \vdash \phi : \Gamma$ by induction on $|\phi|$:

$$[\![\Delta^0 \vdash (\_) : (\_)]\!]_{\mathcal{B}} := \omega : [\![\Delta]\!]^0_{\mathcal{B}} \to I$$

$$[\![\Delta^{d+e} \vdash \phi : \Gamma, \mathsf{a}^s : \mathsf{A}^p]\!]_{\mathcal{B}} := \big([\![\Delta]\!]^{d+e}_{\mathcal{B}} \xrightarrow{\delta} [\![\Delta]\!]^d_{\mathcal{B}} \otimes [\![\Delta]\!]^e_{\mathcal{B}} \cong [\![\Delta]\!]^d_{\mathcal{B}}.\underline{[\![\Delta]\!]^e_{\mathcal{B}}}\{\omega\} \xrightarrow{\phi.\iota[\![\mathsf{a}]\!]^s_{\mathcal{B}}\{\omega\}} [\![\Gamma]\!]_{\mathcal{B}}.[\![\mathsf{A}]\!]^p_{\mathcal{B}}\big),$$

where $\iota[\![\mathsf{a}]\!]^s_{\mathcal{B}}\{\omega\} \in \mathscr{D}([\![\Delta]\!]^0_{\mathcal{B}})(\underline{[\![\Delta]\!]^e_{\mathcal{B}}}\{\omega\}, [\![\mathsf{A}]\!]^p_{\mathcal{B}}\{[\![\phi]\!]_{\mathcal{B}} \circ \omega\})$ is obtained from $[\![\mathsf{a}]\!]^s_{\mathcal{B}} \in \mathscr{D}([\![\Delta]\!]^e_{\mathcal{B}})(\top, [\![\mathsf{A}]\!]^p_{\mathcal{A}}\{[\![\phi]\!]_{\mathcal{B}}\})$ along the natural bijection $\iota : \mathscr{D}([\![\Delta]\!]^e_{\mathcal{B}})(\top, [\![\mathsf{A}]\!]^p_{\mathcal{B}}\{[\![\phi]\!]_{\mathcal{B}}\}) \cong \mathscr{D}(I)(\underline{[\![\Delta]\!]^e_{\mathcal{B}}}, [\![\mathsf{A}]\!]^p_{\mathcal{B}}\{[\![\phi]\!]_{\mathcal{B}}\})$.

Definition 4.3 (semantics of variables)**.** The semantic map $[\![\_]\!]_{\mathcal{B}}$ interprets the rule Var by

$$[\![\Gamma^0, \mathsf{x}^p : \mathsf{A}^p, \Phi^0 \vdash \mathsf{x}^p : \mathsf{A}^p]\!]_{\mathcal{B}} := \pi_2\{\pi_1\} \in \mathscr{D}([\![\Gamma]\!]^0_{\mathcal{B}}.[\![\mathsf{A}]\!]^p_{\mathcal{B}}.[\![\Phi]\!]^0_{\mathcal{B}})(\top, [\![\mathsf{A}]\!]^p_{\mathcal{B}}\{\pi_1\}).$$

Definition 4.4 (semantics of lambda-types)**.** The semantic map $[\![\_]\!]_{\mathcal{B}}$ interprets the rules on lambda-types by

- (Λ-Form) $[\![\Gamma^0 \vdash (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r \ \mathsf{type}]\!]_{\mathcal{B}} := \Lambda([\![\Gamma^0 \vdash \mathsf{A}\ \mathsf{type}]\!]^p_{\mathcal{B}}, [\![\Gamma^0, \mathsf{x}^0 : \mathsf{A}^0 \vdash \mathsf{B}\ \mathsf{type}]\!]^q_{\mathcal{B}})^r$
- (Λ-Intro) $[\![\Gamma^r \vdash (\lambda \mathsf{x}^p.\, \mathsf{b}^t)^r : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r]\!] := \lambda([\![\Gamma, \mathsf{x}^p : \mathsf{A}^p \vdash \mathsf{b}^t : \mathsf{B}^q]\!]_{\mathcal{B}})^r$
- (Λ-Elim) $[\![\Gamma^{(g+rh)k} \vdash \mathsf{App}(\mathsf{f}^t, \mathsf{a}^{rs})^k : \mathsf{B}\{\mathsf{a}/\mathsf{x}\}^{qrk}]\!]_{\mathcal{B}} := \mathrm{ev} \circ ([\![\Gamma^g \vdash \mathsf{f}^t : (\Lambda_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r]\!]_{\mathcal{B}} \otimes [\![\Gamma^h \vdash \mathsf{a}^s : \mathsf{A}^p]\!]^s_{\mathcal{B}} \circ \delta)^k$.

Definition 4.5 (semantics of sigma-types)**.** The semantic map $[\![\_]\!]$ interprets sigma-types by

- (Σ-Form) $[\![\Gamma^0 \vdash (\Sigma_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r \ \mathsf{type}]\!]_{\mathcal{B}} := \Sigma([\![\Gamma^0 \vdash \mathsf{A}\ \mathsf{type}]\!]^p_{\mathcal{B}}, [\![\Gamma^0, \mathsf{x}^0 : \mathsf{A}^0 \vdash \mathsf{B}\ \mathsf{type}]\!]^q_{\mathcal{B}})^r$;
- (Σ-Intro) $[\![\Gamma^r \vdash \langle \mathsf{a}^s, \mathsf{b}^t\rangle^r : (\Sigma_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r]\!]_{\mathcal{B}} := \langle [\![\Gamma \vdash \mathsf{a}^s : \mathsf{A}]\!]_{\mathcal{B}}, [\![\Gamma \vdash \mathsf{b}^t : \mathsf{B}\{\mathsf{a}/\mathsf{x}\}]\!]^q_{\mathcal{B}}\rangle^r$;
- (Σ-Elim) $[\![\Gamma^k \vdash \pi_1(\mathsf{c}^u)^k : \mathsf{A}^{prk}]\!] := \pi_1 \circ [\![\Gamma \vdash \mathsf{c}^u : (\Sigma_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r]\!]^k_{\mathcal{B}}$ and $[\![\Gamma^k \vdash \pi_2(\mathsf{c}^u)^k : \mathsf{B}\{\pi_1(\mathsf{c}^u)/\mathsf{x}\}^{qrk}]\!] := \pi_2\{[\![\Gamma \vdash \mathsf{c}^u : (\Sigma_{\mathsf{x}^p:\mathsf{A}^p}\mathsf{B}^q)^r]\!]^k_{\mathcal{B}}\}$.

Definition 4.6 (semantics of one-types)**.** The semantic map $[\![\_]\!]_{\mathcal{B}}$ interprets the one-type by

- (1-Form) $[\![\Gamma^0 \vdash 1^p \ \mathsf{type}]\!] := 1^p$
- (1-Intro) $[\![\Gamma \vdash \star^p : 1^p]\!] := \star^p$.

Definition 4.7 (semantics of bottom-types)**.** The semantic map $[\![\_]\!]_{\mathcal{B}}$ interprets the bottom-type by

- ($\bot$-Form) $[\![\Gamma^0 \vdash \bot^p \ \mathsf{type}]\!]_{\mathcal{B}} := \bot^p$;
- ($\bot$-Elim) $[\![\Gamma^r, \Delta^r \vdash \mathsf{R}^{\bot}_{\mathsf{A}}(\mathsf{b}^s)^r : \mathsf{A}\{\mathsf{b}/\mathsf{x}\}^{qr}]\!]_{\mathcal{B}} := \xi \circ [\![\Gamma \vdash \mathsf{b}^s : \bot^p]\!]_{\mathcal{B}}$.

4.2 Soundness

Soundness is proven by the standard method as in the case of MLTT [Hof97].

References

[Atk18] Robert Atkey, *Syntax and semantics of quantitative type theory*, Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, 2018, pp. 56–65.

[AW09] Steve Awodey and Michael A Warren, *Homotopy theoretic models of identity types*, Mathematical proceedings of the cambridge philosophical society, vol. 146, Cambridge University Press, 2009, pp. 45–55.

[B$^+$84] Hendrik Pieter Barendregt et al., *The lambda calculus*, vol. 3, North-Holland, Amsterdam, 1984.

[BBDPH93] Nick Benton, Gavin Bierman, Valeria De Paiva, and Martin Hyland, *A term calculus for intuitionistic linear logic*, International Conference on Typed Lambda Calculi and Applications, Springer, 1993, pp. 75–90.

[BGMZ14] Aloïs Brunel, Marco Gaboardi, Damiano Mazza, and Steve Zdancewic, *A core quantitative coeffect calculus.*, ESOP, vol. 8410, Springer, 2014, pp. 351–370.

[BP96] Andrew Barber and Gordon Plotkin, *Dual intuitionistic linear logic*, University of Edinburgh, Department of Computer Science, Laboratory for . . . , 1996.

[CAB$^+$86] RL Constable, SF Allen, HM Bromley, WR Cleaveland, JF Cremer, RW Harper, DJ Howe, TB Knoblock, NP Mendler, P Panangaden, et al., *Implementing mathematics with the nuprl proof development system.*

[CH88] Thierry Coquand and Gerard Huet, *The calculus of constructions*, Information and computation **76** (1988), no. 2, 95–120.

[Chu40] Alonzo Church, *A formulation of the simple theory of types*, The journal of symbolic logic **5** (1940), no. 2, 56–68.

[CP96] I Cervesato and F Pfenning, *A linear logical framework*, Proceedings 11th Annual IEEE Symposium on Logic in Computer Science, IEEE, 1996, pp. 264–275.

[Gen35] Gerhard Gentzen, *Untersuchungen über das logische schließen. i*, Mathematische zeitschrift **39** (1935), no. 1, 176–210.

[Gir87] Jean-Yves Girard, *Linear logic*, Theoretical computer science **50** (1987), no. 1, 1–101.

[GL12] Marco Gaboardi and Ugo Dal Lago, *Linear dependent types and relative completeness*, Logical Methods in Computer Science **8** (2012).

[GS14] Dan R Ghica and Alex I Smith, *Bounded linear types in a resource semiring*, Programming Languages and Systems: 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings 23, Springer, 2014, pp. 331–350.

[Her05] Hugo Herbelin, *On the degeneracy of σ-types in presence of computational classical logic*, Typed Lambda Calculi and Applications: 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005. Proceedings 7, Springer, 2005, pp. 209–220.

[Hof97] Martin Hofmann, *Syntax and Semantics of Dependent Types*, Extensional Constructs in Intensional Type Theory, Springer, 1997, pp. 13–54.

[HS98] Martin Hofmann and Thomas Streicher, *The Groupoid Interpretation of Type Theory*, Twenty-five years of constructive type theory (Venice, 1995) **36** (1998), 83–111.

[Jac93] Bart Jacobs, *Comprehension categories and the semantics of type dependency*, Theoretical Computer Science **107** (1993), no. 2, 169–207.

[KL21] Krzysztof Kapulkin and Peter LeFanu Lumsdaine, *The simplicial model of univalent foundations (after voevodsky)*, Journal of the European Mathematical Society **23** (2021), no. 6, 2071–2126.

[KPB15] Neelakantan R Krishnaswami, Pierre Pradic, and Nick Benton, *Integrating linear and dependent types*, ACM SIGPLAN Notices **50** (2015), no. 1, 17–30.

[Lum09] Peter LeFanu Lumsdaine, *Weak ω-categories from intensional type theory*, Typed lambda calculi and applications, Springer, 2009, pp. 172–187.

[McB16] Conor McBride, *I got plenty o'nuttin'*, A List of Successes That Can Change the World: Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday (2016), 207–233.

[ML75] Per Martin-Löf, *An Intuitionistic Theory of Types: Predicative Part*, Studies in Logic and the Foundations of Mathematics **80** (1975), 73–118.

[ML82] ———, *Constructive Mathematics and Computer Programming*, Studies in Logic and the Foundations of Mathematics **104** (1982), 153–175.

[ML84a] Per Martin-Lof, *Constructive mathematics and computer programming*, Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences **312** (1984), no. 1522, 501–518.

[ML84b] Per Martin-Löf, *Intuitionistic Type Theory: Notes by Giovanni Sambin of a series of lectures given in Padova, June 1980*, 1984.

[ML98] ———, *An Intuitionistic Theory of Types*, Twenty-five years of constructive type theory **36** (1998), 127–172.

[ML13] Saunders Mac Lane, *Categories for the working mathematician*, vol. 5, Springer Science & Business Media, 2013.

[NPS90] Bengt Nordström, Kent Petersson, and Jan M Smith, *Programming in martin-löf's type theory, volume 7 of international series of monographs on computer science*, 1990.

[POM14] Tomas Petricek, Dominic Orchard, and Alan Mycroft, *Coeffects: a calculus of context-dependent computation*, ACM SIGPLAN Notices **49** (2014), no. 9, 123–135.

[PS12] KATE PONTO and MICHAEL SHULMAN, *Duality and traces for indexed monoidal categories*, Theory and Applications of Categories **26** (2012), no. 23, 582–659.

[Sch09] Helmut Schwichtenberg, *Program extraction in constructive analysis*, Springer, 2009.

[SHU13] MICHAEL SHULMAN, *Enriched indexed categories*, Theory and Applications of Categories **28** (2013), no. 21, 616–695.

[SU06] Morten Heine Sørensen and Pawel Urzyczyn, *Lectures on the curry-howard isomorphism*, Elsevier, 2006.

[Tro91] Anne Sjerp Troelstra, *Lectures on linear logic.*

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg, *Basic proof theory*, no. 43, Cambridge University Press, 2000.

[Uni13] The Univalent Foundations Program, *Homotopy type theory: Univalent foundations of mathematics*, `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[Vák15] Matthijs Vákár, *A categorical semantics for linear logical frameworks*, Foundations of Software Science and Computation Structures: 18th International Conference, FOSSACS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 18, Springer, 2015, pp. 102–116.

[vdBG11] Benno van den Berg and Richard Garner, *Types are weak ω-groupoids*, Proceedings of the London Mathematical Society **102** (2011), no. 2, 370–394.

[War11] Michael A Warren, *The strict ω-groupoid interpretation of type theory*, Models, logics, and higher-dimensional categories **53** (2011), 291–340.