# A Formal Logic for Formal Category Theory (Extended Version)

Max S. New[1,2][0000−0001−8141−195X] and Daniel R. Licata[2]

[1] University of Michigan
[2] Wesleyan University

**Abstract.** We present a domain-specific type theory for constructions and proofs in category theory. The type theory axiomatizes notions of category, functor, profunctor and a generalized form of natural transformations. The type theory imposes an ordered linear restriction on standard predicate logic, which guarantees that all functions between categories are functorial, all relations are profunctorial, and all transformations are natural by construction, with no separate proofs necessary. Important category-theoretic proofs such as the Yoneda lemma and Co-yoneda lemma become simple type-theoretic proofs about the relationship between unit, tensor and (ordered) function types, and can be seen to be ordered refinements of theorems in predicate logic. The type theory is sound and complete for a categorical model in *virtual equipments*, which model both internal and enriched category theory. While the proofs in our type theory look like standard set-based arguments, the syntactic discipline ensure that all proofs and constructions carry over to enriched and internal settings as well.

## 1 Introduction

Category theory is a branch of mathematics that studies higher-dimensional typed algebraic structures. Originally developed for applications to homological algebra, it was quickly discovered that categorical structures were common in logic and computer science. Formal systems like logics, type theories and programming languages typically have sound and complete models given by notions of structured categories [32, 31, 35]. This Curry-Howard-Lambek correspondence applies to simply typed lambda calculus [31], computational lambda calculus [35], linear logic [25] dependent type theory [15, 46], and many other type theories designed based on category-theoretic semantics. The syntax of a type theory should present an initial object in its category of models, a category-theoretic reformulation of logical soundness and completeness.

While this research program has been quite successful, category-theoretic notions can be overwhelming for beginners. In a traditional set-theoretic formulation, notions such as adjoint functors and limits produce a proliferation of "naturality" and "functoriality" side-conditions that must be discharged. For example, when constructing an adjoint pair of functors between two categories,

a naïve approach would define all of the data of the action on objects, action on arrows, prove the functoriality of such actions, as well as construct two families of transformations, prove they are natural and then finally proving a pair of equalities relating compositions of natural transformations. Carrying out these proofs explicitly is quite tedious and many newcomers are left with the impression that category theory is full of long, but ultimately trivial constructions. This complexity is compounded when moving from ordinary category theory to enriched and internal category theory, where constructions must be additionally proven continuous, monotone, etc, in addition to natural or functorial. However, these generalizations are often exactly what is needed for programming language applications; for example, domain-, metric- and step-index-enriched categories have been used to model recursive programming languages and internal categories have been used to model parametricity and gradual typing [54, 10, 45, 37].

Fortunately, the tools of category theory itself can be employed to simplify this complexity, specifically the tools of *higher* category theory. As an analogy in differential calculus, when an adept analyst writes down a function, they do not expand out the $\epsilon-\delta$ definition of continuity for a function and proceed from first principles, but rather use certain *syntactic principles* for defining functions that are continuous by construction — e.g. that composition of continuous functions is continuous. Similar principles apply to category theory itself: functors and natural transformations are closed under composition and whiskering operations, and experienced category theorists rely on these syntactic principles to eliminate the tedium of explicit proofs. In the case of category theory, these principles can be formalized using algebraic structures such as 2-categories, bicategories, Yoneda structures, (virtual) double categories, pro-arrow equipments [7, 57, 50, 33, 18], an approach known as *formal category theory*. In these structures, rather than defining notions of category, functor and natural transformation from first principles, they are axiomatized in a manner similar to how a category axiomatizes a notion of space and homomorphism. Proofs in formal category theory apply to enriched and internal settings, which are instances of the formal axioms. A downside is that these algebraic structures are quite complicated, and practitioners typically employ either an algebraic combinator syntax (formalized in [19]) or a 2-dimensional diagrammatic language that can be quite beautiful and elegant, but is also somewhat removed from the traditional formulation of category theory in terms of sets and functions.

In this work, we apply the techniques of categorical logic to define a more familiar logical syntax for carrying out constructions and proofs in formal category theory. We call the resulting theory *virtual equipment type theory* (VETT) as (hyperdoctrines of) *virtual equipments* [33, 18], a particular semantic model of formal category theory, provide a sound and complete notion of model for the theory. VETT provides syntax for categories, functors, profunctors, and natural transformations, which are defined using familiar term syntax and $\beta\eta$ reasoning principles for $\lambda$-functions, bound variables, tuples, etc. By adhering to a *syntactic discipline*, the logic guarantees that all functor terms are automatically

functorial, and all natural transformation terms are natural. More specifically, the syntax for transformations is a kind of *indexed, ordered linear lambda* calculus, where the indexing ensures that transformations are correctly natural and the ordering and linearity ensure that the proofs are valid in a large class of enriched and internal categories, such as enrichment in a non-symmetric monoidal category. VETT provides an alternative to algebraic and string-diagram syntaxes for working with virtual equipments, similar to how the lambda calculus provides an alternative to categorical combinators and string diagram calculi for cartesian closed categories.

The syntax of VETT is an indexed, ordered linear, proof-relevant variant of predicate logic over a unary type theory. Just as a predicate logic has a notion of type, term, relation and implication, VETT is based on four analogous category-theoretic concepts: categories, functors, profunctors and natural transformations of profunctors. Categories are treated like types, and the unary functors we consider in this paper are each represented by a term whose type is a category and whose one free variable ranges over a category. The analog of a relation is a *profunctor* (defined below), which is written like a set with free category variables. Like the restriction to unary functors, we restrict to profunctors with two free variables. The logic is proof-relevant in that the implications of relations are generalized to natural transformations of profunctors, and we use a $\lambda$-calculus notation to describe these "proof terms". This analogy to predicate logic can be made formal: any construction in VETT can be erased to a corresponding construction or proof in predicate logic, as sets, functions, relations, and implication of relations define a (somewhat degenerate) virtual equipment.

While the restricted syntax developed in this paper does not express some important concepts such as functor categories or opposite categories, the restriction is natural in that it corresponds exactly to virtual equipments, a well-understood notion of model that can express a great deal of fundamental results and constructions in category theory [44, 48]. Moreover, we can work around these unary/binary restrictions to some extent by viewing the type theory as a domain-specific language embedded in a metalanguage. For example, while we cannot talk about functor categories, we can state a theorem that quantifies over functors using the meta-language's "external" universal quantifier (which does not have automatic functoriality/naturality properties). To support this, VETT includes a third layer, an extensional dependent type theory in the style of Martin-Löf type theory. All of our ordered predicate logic judgments are also indexed by a context from this dependent type theory, and the type theory includes universe types for categories, functors, profunctors and natural transformations. This allow us to formalize theorems the object logic is too restrictive to encode, analogous to 2-level [52, 2, 40] or indexed type theories [28, 16, 53, 30].

While we emphasize the applications to enriched and internal category theory in this work, there is potential for more direct application to programming language semantics. Ordinary predicate logic is the foundation for proof-theoretic presentations of logical relations, such as Abadi-Plotkin logic for parametricity and LSLR and Iris for step-indexed logical relations proofs [41, 21, 29]. We con-

jecture that VETT might similarly serve as the foundation for a logic of *ordered* structures, which abound in applications: rewriting and approximation relations can both be modeled as orderings and logical relations involving these structures are proven to respect orderings; operational logical relations must be downward-closed and approximation relations should satisfy transitivity. Just as LSLR and Iris release the user from the syntactic burden of explicit step-indexing, VETT may be used to release the user from the syntactic burden of proving downward-closure or transitivity side-conditions. Additionally, VETT may serve as the basis of a future domain specific proof assistant for category-theoretic proofs. To pilot-test this, we have formalized the syntax of VETT in Agda 2.6.2.2, using the rewrite mechanism to make VETT's substitution and $\beta$-reduction rules definitional equalities.[3] We have used this lightweight implementation to check a number of examples.

**Basics of Profunctors.** While we assume the reader has some background knowledge of category theory, we briefly define profunctors, which are not included in many introductory texts. Recall that a category $\mathbb{C}$ has a collection of objects and morphisms with identity and composition, and a functor $F : \mathbb{C} \to \mathbb{D}$ is a function on objects and a function on morphisms that preserves identity and composition. A category can be thought of as a generalization of a preordered set, which has a set of elements and a binary *relation* on its objects satisfying reflexivity and transitivity. A category is then a *proof-relevant preorder*, where morphisms are the proofs of ordering, and the reflexivity and transitivity proofs must satisfy identity and unit equations. A functor is then a *proof-relevant monotone function*. Given categories $\mathcal{C}$ and $\mathcal{D}$, a profunctor $R$ from $\mathcal{C}$ to $\mathcal{D}$, written $R : \mathbb{C} \nrightarrow \mathbb{D}$ is a functor $R : \mathbb{C}^o \times \mathbb{D} \to \mathrm{Set}$[4]. Because a profunctor outputs a Set rather than a proposition, it is itself a *proof-relevant relation*. Thinking of categories as proof-relevant preorders, functoriality says that the profunctor is downward-closed in $\mathbb{C}$ and upward-closed in $\mathbb{D}$. Given profunctors $R, S : \mathbb{C} \nrightarrow \mathbb{D}$, a homomorphism from $R$ to $S$ is a natural transformation, which in the preordered setting is simply an implication of relations.

Profunctors are very useful for formalizing category theory, but an additional reason we make them a basic concept of VETT is that they allow us to give a *universal property* for the type of "morphisms in a category $\mathbb{C}$". This is analogous to how the $J$ elimination rule for the identity type in Martin-Löf type theory gives a universal property for morphisms in a groupoid (the special case of a category where all morphisms are invertible) [27, 6, 51]. The reason profunctors are useful for this purpose is that, for any category $\mathbb{C}$, $\mathrm{Hom}_{\mathbb{C}} : \mathbb{C} \nrightarrow \mathbb{C}$ is a profunctor. On preorders this is just the preorder's ordering relation itself. Moreover, the hom profunctor is the unit for a composition of profunctors $R \odot S$ which is defined as a *co-end*. The composition of profunctors is a generalization of the composition of relations, and just as the equality relation is the identity for the composition of relations, the hom profunctor is the identity for this composition. The unit law for the hom profunctor can be seen as a "morphism

[3] https://github.com/maxsnew/virtual-equipments/blob/master/agda/STC.agda
[4] $\mathbb{C}^o$ is the notation we use for the opposite category of $\mathbb{C}$

induction" principle, analogous to the "path induction" used in homotopy type theory (though in this paper we consider only ordinary 1-dimensional categories, not higher generalizations).

**Outline.** In Section 2 we introduce the syntax of VETT. In Section 3 we demonstrate how to use our syntax for formal category theory. In Section 4, we develop some model theory for VETT, including a sound and complete notion of categorical model and sound interpretation in virtual equipments modeling ordinary, enriched and internal category theory. In Section 5, we discuss related type theories and potential extensions.

## 2 Syntax of VETT

In Figure 1 we give a table summarizing the relationship between the judgments and connectives of higher-order predicate logic with our ordered variant. Due to the incorporation of variance, some unordered concepts generalize to multiple different ordered notions. For instance, covariant and contravariant presheaf categories generalize the power set. Further, because we only have binary relations rather than relations of arbitrary arity, we have only restricted forms of universal and existential quantification which come combined with implications and conjunctions.

| Higher-Order Logic | Virtual Equipment Type Theory |
|---|---|
| Set $X$ | Category $\mathbb{C}$ |
| $X \times Y$ | $\mathbb{C} \times \mathbb{D}$ |
| $1$ | $1$ |
| $\mathcal{P}X$ | $\mathcal{P}^+X$ and $\mathcal{P}^-X$ |
| $\{(x,y) \in X \times Y \mid R(x,y)\}$ | $\sum_{\alpha:C;\beta:D} R$ |
| Function $f(x:X):Y$ | Functor/Object $\alpha : \mathbb{C} \vdash A : \mathbb{D}$ |
| Relation $R(x,y)$ | Profunctor/Set $\alpha : \mathbb{C}; \beta : \mathbb{D} \vdash R$ |
| $R \wedge Q$ | $R \times Q$ |
| $\top$ | $1$ |
| $\forall x.P \Rightarrow Q$ | $P \triangleright^{\forall \alpha:\mathbb{C}} Q$ and $Q \,{}^{\forall \alpha:\mathbb{C}}\triangleleft P$ |
| $\exists x.P \wedge Q$ | $P \overset{\exists \alpha:\mathbb{C}}{\odot} Q$ |
| $x =_X y$ | $\alpha \to_{\mathbb{C}} \beta$ |
| Proof $\forall \overrightarrow{\alpha}.R_1 \wedge \cdots \Rightarrow Q$ | Nat. Trans./Element $\alpha_1, x_1 : R_1(\alpha_1, \alpha_2), \ldots \vdash t : Q$ |

**Fig. 1.** Analogy between Higher-Order Logic and VETT Judgments and Connectives

The syntactic forms of VETT are given in Figure 2. First, we have categories, which are analogous to sorts in a first-order theory. We have $M$ a base sort, product and unit sorts, as well as the graph of a profunctor and the negative and positive presheaf categories. Next, objects $a, b, c$ are the syntax for the *functors* between categories. We call them objects rather than functors, because

in type-theoretic style, a functor is viewed as a "generalized object" parameterized by an input variable $\alpha : \mathbb{C}$. Next, sets $P, Q, R$ are the syntax for *sets*. These sets denote *profunctors*, i.e., a categorification of relations. Similar to functors, rather than writing profunctors as functions $\mathbb{C}^o \times \mathbb{D} \to \mathrm{Set}$, we write them as sets with a contravariant variable $\alpha : \mathbb{C}$ and a covariant variable $\beta : \mathbb{D}$. The sets we can define are the Hom-set, the tensor and internal hom, as well as products of sets, profunctors applied to two objects and elements of positive and negative presheaves. Finally we have elements of sets, which correspond to natural transformations of multiple inputs, where again we view natural transformations valued in a profunctor as generalized elements of profunctors.

After these forms we have types and terms, which represent the meta-language that we use to talk about categories/profunctors/natural transformations. In addition to standard dependent type theory with $\Pi$ and $\Sigma$ and identity types, we have universes of categories, functors, profunctors and natural transformations.

Finally we have several forms of context which are used in the theory. The contexts $\Gamma$ of term variables with their types are as usual; we write "$\Gamma$ type context" to indicate that a context is well-formed. We name the remaining contexts after the judgements that they are used by. The set contexts $\Xi$, which will be used to type-check sets, contain object variables with their categories. The two forms of set context are $\alpha : \mathbb{C}$, containing one variable that can be used both contravariantly and covariantly, and $\alpha : \mathbb{C}; \beta : \mathbb{D}$, containing a contravariant variable $\alpha$ and covariant variable $\beta$. Finally, the transformation contexts $\Phi$ contain element variables with their sets, alternating with those sets' object variables with their categories. A typical $\Phi$ has the shape

$$\alpha_1 : \mathbb{C}_1, x_1 : R_1(\alpha_1, \alpha_2), \alpha_2 : \mathbb{C}_2, x_2 : R_2(\alpha_2, \alpha_3), \ldots, R_n(\alpha_n, \alpha_{n+1}), \alpha_{n+1} : \mathbb{C}_{n+1}$$

and represents the composition of the "relations" $R_1, R_2, R_3, \ldots, R_n$. We write $d^-(\Phi)$ for the first category variable in $\Phi$ (which we regard as the negative or contravariant position), $d^+(\Phi)$ for the last category variable in $\Phi$ (which we regard as the positive or covariant position) and use the notation $d^\pm \Xi$ with the same meaning. We write $\Phi_1 \between \Phi_2$ for the append of two transformation contexts, which is only well-formed when the last variable in $\Phi_1$ is equal to the first variable in $\Phi_2$. Formal inductive definitions are in the appendix, but intuitively:

$$\begin{aligned} d^-(\alpha_1 : \mathbb{C}_1, x_1 : R_1(\alpha_1, \alpha_2), \ldots, x_n : R_n(\alpha_n, \alpha_n), \alpha_{n+1} : \mathbb{C}_{n+1}) &= \alpha_1 : \mathbb{C}_1 \\ d^+(\alpha_1 : \mathbb{C}_1, x_1 : R_1(\alpha_1, \alpha_2), \ldots, x_n : R_n(\alpha_n, \alpha_n), \alpha_{n+1} : \mathbb{C}_{n+1}) &= \alpha_{n+1} : \mathbb{C}_{n+1} \\ (\Phi_1, \beta : \mathbb{D}) \between (\beta : \mathbb{D}, \Phi_2) &= \Phi_1, \beta : \mathbb{D}, \Phi_2 \end{aligned}$$

Next, we overview our basic judgement forms. We have

- Categories: $\Gamma \vdash \mathbb{C}$ Cat, where $\Gamma$ type context.
- Objects/functors: $\Gamma \mid \alpha : \mathbb{C} \vdash a : \mathbb{D}$, where $\Gamma \vdash \mathbb{C}$ Cat and $\Gamma \vdash \mathbb{D}$ Cat. Objects are typed with an input object variable $\alpha : \mathbb{C}$ and an output category $\mathbb{D}$; in the semantics, objects are modeled as functors $\mathbb{C} \to \mathbb{D}$.
- Sets/profunctors: $\Gamma \mid \Xi \vdash S$ Set, where $\Gamma \vdash \Xi$ set context. A set $S$ is typed with respect to a set context $\Xi$ to describe its covariant/contravariant dependence on some input objects. Sets are semantically modeled as profunctors.

$$
\begin{array}{rrcl}
\text{Categories} & \mathbb{C}, \mathbb{D}, \mathbb{E} & ::= & \lfloor M \rfloor \mid \mathbb{C} \times \mathbb{D} \mid \mathbb{1} \mid \sum_{\alpha;\beta} P \mid \mathcal{P}^- \mathbb{C} \mid \mathcal{P}^+ \mathbb{C} \\
\text{Objects} & a, b, c & ::= & \alpha \mid Ma \mid (a, b) \mid () \mid \pi_i a \mid (a_-, a_+, s) \mid \pi_- a \mid \pi_+ a \mid \lambda \alpha : \mathbb{C}.R \\
\text{Sets} & P, Q, R & ::= & a \to_{\mathbb{C}} b \mid P \overset{\exists \beta}{\odot} Q \mid P \triangleright^{\forall \beta} Q \mid S \,{}^{\forall \alpha}\!\triangleleft R \mid 1 \mid P \times Q \\
& & & \mid M(a; b) \mid b \in a \mid a \ni b \\
\text{Elements} & s, t, u & ::= & x \mid \mathrm{ind}_{\to}(\alpha.t, b_1, s, b_2) \mid \mathrm{id}_b \mid \mathrm{ind}_{\odot}(x, \beta, y.r; s) \mid (s, b, t) \mid s \triangleright^a t \\
& & & \mid \lambda^{\triangleright}(x, \alpha).s \mid s \,{}^a\!\triangleleft t \mid \lambda^{\triangleleft}(\alpha, x).s \mid \pi_i s \mid (s_1, s_2) \mid () \mid \pi_e a \mid M^b \\
\text{Type} & A, B, C & ::= & \ldots \mid \mathrm{SmallCat} \mid \mathrm{Cat} \mid \mathrm{Fun}\, \mathbb{C}\, \mathbb{D} \mid \mathrm{Prof}\, \mathbb{C}\, \mathbb{D} \mid \forall \alpha : \mathbb{C}.R \\
\text{Term} & L, M, N & ::= & \ldots \mid \lceil \mathbb{C} \rceil \mid \lambda \alpha : \mathbb{C}.a \mid \lambda(\alpha : \mathbb{C}; \beta : \mathbb{D}).R \mid \lambda \alpha.t \\
\text{Type Context} & \Gamma, \Delta & ::= & \cdot \mid \Gamma, X : A \\
\text{Set Context} & \Xi, Z & ::= & \alpha : \mathbb{C} \mid \alpha : \mathbb{C}; \beta : \mathbb{D} \\
\text{Trans. Context} & \Phi, \Psi & ::= & \alpha : \mathbb{C} \mid \Phi, x : P, \beta : \mathbb{D}
\end{array}
$$

**Fig. 2.** VETT Syntactic Forms

- Elements/natural transformations: $\Gamma \mid \Phi \vdash s : R$, where $\Gamma \vdash \Phi$ trans. context and $\Gamma \mid \underline{\Phi} \vdash R$ Set. A transformation $s$ has a context $\Phi$ of transformation variables and a single output set $R$. To be well-formed, the context and set must be parameterized by the same contravariant and covariant object variables. To ensure this, we use a coercion operation $\underline{\Phi}$ from transformation contexts to set contexts that erases everything in the context but the left-most and right-most object variables ($\underline{\alpha : \mathbb{C}} = \alpha : \mathbb{C}$ and $\underline{\Phi} = d^-(\Phi); d^+(\Phi)$).
- Meta-language types and terms: $\Gamma \vdash A$ Type and $\Gamma \vdash M : A$ as in standard dependent type theory.

The variable rules for objects and elements are

$$
\frac{}{\Gamma \mid \alpha : \mathbb{C} \vdash \alpha : \mathbb{C}} \qquad \frac{}{\Gamma \mid \alpha : \mathbb{C}, x : R, \beta : \mathbb{D} \vdash x : R}
$$

As when using variables in linear logic, the latter rule applies only when the context contains a single set $R$. All syntactic forms typed in context admit an action of substitution. For types and terms, this is as usual. Objects $\alpha : \mathbb{C} \vdash a : \mathbb{D}$ can be substituted for object variables $\beta : \mathbb{D}$ in other objects. We can also substitute objects into *sets*, that is, if we have a set $P$ parameterized by a contravariant variable $\alpha : \mathbb{C}$ and a covariant variable $\beta : \mathbb{D}$, then we can substitute objects $a : \mathbb{C}$ and $b : \mathbb{D}$ for these variables $P[a/\alpha; b/\beta]$. This generalizes the ordinary precomposition of a relation by a function. Semantically this is the "restriction" of a profunctor along two functors, which is just composition of functors if a profunctor is viewed as a functor to Set. Modeling this operation as a substitution considerably simplifies reasoning using profunctors. Finally we have the action of substitution on elements/natural transformations. First, we can substitute elements/natural transformations for the set variables in elements, denoting the composition of natural transformations. Second, an element is also parameterized by a contravariant and a covariant category variable $\alpha; \beta$. We can think of natural transformations as *polymorphic* in the categories involved, and so

when we make a transformation substitution, we also *instantiate* the polymorphic category variables with objects. The full syntactic details of substitution are included in the appendix.

### 2.1 Category Connectives

In this section we discuss some connectives for constructing categories, which are specified by introduction and elimination rules in Figure 3 (the $\beta\eta$ equality and substitution rules are included in the appendix). The introduction and elimination rules make use of functors, profunctors, and natural transformations. First we introduce the additive connectives: the unit category 1 and product category $\mathbb{C} \times \mathbb{D}$ have the usual introduction and elimination rules defining functors to/from them. Next, we introduce the *graph of a profunctor* $\sum_{\alpha;\beta} P$. Just as a relation $R : A \times B \to \text{Set}$ can be viewed as a subset $\{(a, b) \in A \times B | R(a, b)\}$, any profunctor $P : \mathbb{C}_-^o \times \mathbb{D}_+ \to \text{Set}$ can be viewed as a category with a functor to $\mathbb{C}_- \times \mathbb{D}_+$ (no op), specifically a two-sided discrete fibration. In set-based category theory, the objects of $\sum_{\alpha;\beta} P$ are triples $(a_-, a_+, s : P(a_-, a_+))$ and morphisms from $(a_-, a_+, s)$ to $(a'_-, a'_+, s')$ are pairs of morphisms $f_- : a_- \to a'_-$ and $f_+ : a_+ \to a'_+$ such that $P(\text{id}, f_+)(s) = P(f_-, \text{id})(s')$. With various choices of $P$, this connective can be used to define the arrow category, slice category, comma category and category of elements. In our syntax we define it as the universal category $\mathbb{C}$ equipped with functors to $\mathbb{C}_-$ and $\mathbb{C}_+$ and a natural transformation to $P$.

Lastly, we define the *negative* and *positive* presheaf categories $\mathcal{P}^-\mathbb{C}$ and $\mathcal{P}^+\mathbb{D}$. These are given a syntax suggestive of the fact that they generalize the notion of a powerset, and so can be thought of as "power categories". Note that we include a restriction that the input category is *small*, which is an inductively defined by saying all base categories are small, the unit is small, product of small categories is small and the graph of a profunctor over small categories is small. Notably, the presheaf categories themselves are not small. The negative presheaf category is defined by its universal property that a functor into it $\mathbb{D} \to \mathcal{P}^-\mathbb{C}$ is equivalent to a profunctor $\mathbb{C}^o \times \mathbb{D} \to \text{Set}$. The introduction rule constructs an object of the negative presheaf category from such a profunctor and the elimination rule inverts it. We use the notation $p \in a$ for the elements of the induced profunctor. Since $a$ occurs in a negative position, it must depend only on the contravariant variable $d^-\Xi$ and vice-versa for $p$. The positive presheaf category is then the dual. In ordinary set-theoretic category theory the negative presheaf category is the usual presheaf category $\text{Set}^{\mathbb{C}^o}$, and the positive presheaf category is the opposite of the dual presheaf category $(\text{Set}^{\mathbb{D}})^o$.

### 2.2 Set Connectives

Next, in Figure 4, we cover the connectives for the sets/profunctors, which classify elements/natural transformations (the $\beta/\eta$-rules are in the appendix). First, the unit set $a \to_{\mathbb{C}} b$ is our syntax for the profunctor of morphisms in $\mathbb{C}$ instantiated at generalized objects $a$ and $b$. Its introduction and elimination rules are

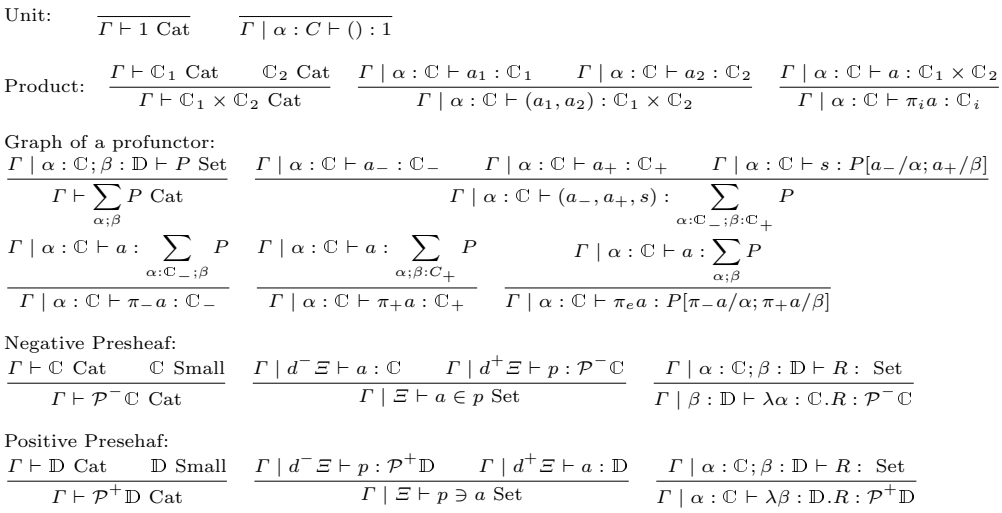



**Fig. 3.** Category Conectives

analogous to the usual rules for equality in intensional Martin-Löf type theory. The introduction rule is the identity morphism (reflexivity) and the elimination rule is an induction principle: we can use a term of $s : a \to_{\mathbb{C}} b$ by specifying the behavior when $s$ is of the form $\mathrm{id}_\alpha$ in the form of a continuation $\alpha.t$. Like the $J$ elimination rule for equality in Martin-Löf type theory, $P$ must be "fully general", i.e. well-typed for variables $\alpha$ and $\beta$. This is because for distinct variables $\alpha$ and $\beta$, $\alpha \to_{\mathbb{C}} \beta$ denotes the unit in a virtual double category, which has a universal property, but $a \to_{\mathbb{C}} b$ denotes a restriction of the unit, which in general does not. Those familiar with linear logic as in e.g. [42] might expect a more general rule, where the continuation $t$ is allowed to use variables that are not used in $s$, i.e., have a context $\Phi_l \between \Phi_r$ and the conclusion of the rule to have a context $\Phi_l \between \Phi \between \Phi_r$. Because of dependency, this is not necessarily well-formed in cases where the endpoints $a$ and $b$ of $a \to_{\mathbb{C}} b$ are not distinct variables. However, the instances of this more general rule that do type check are derivable from our more restricted rule using right/left-hom types.

The tensor product of sets is a kind of combined existential quantifier and monoidal product, which we combine into a single notation $P \overset{\exists\beta}{\odot} Q$, where $\beta$ is the covariant variable of $P$ and the contravariant variable of $Q$. Then the covariant variable of the tensor product is the covariant variable of $Q$ and the contravariant variable similarly comes from $P$. In ordinary category theory, this is the *composition* of profunctors, and is defined by a coend of a product. We require that the variable $\beta$ quantifies over a small category $\mathbb{D}$, as in general this composite doesn't exist for large categories. The introduction and elimination are

like those for a combined tensor product and existential type: the introduction rule is a pair of terms, with an appropriate instantiation of $\beta$, and the elimination rule says to use a term of a tensor product, it is sufficient to specify the behavior on two elements typed with an arbitrary middle object $\beta$.

Next, we introduce the contravariant ($P \,{}^{\forall\alpha}\!\triangleleft\, R$) and covariant ($R \triangleright^{\forall\alpha}\, P$) homs of sets, which are different from each other because we are in an ordered logic. These are a kind of universally quantified function type, where the universally quantified variable must occur with the same variance in domain and codomain. In the contravariant case, it occurs as the contravariant variable in both, and vice-versa for the covariant case. To highlight this, the notation for the contravariant dependence puts the quantified variable on the *left* of the triangle, as contravariant variables occur to the left of the covariant variable, and similarly the covariant hom has the quantified variable on the right. Similar to ordered lambda calculus, the covariant hom is right-associative while the contravariant hom is left-associative. Then the covariant variable of the contravariant hom set is the covariant variable of the codomain and, and the contravariant variable of the hom set is the *covariant* variable of the domain, as the two contravariances cancel. The covariant hom is dual. Semantically, in ordinary category theory these are known as the *hom* of profunctors and are adjoint to the composition of profunctors [8]. The two connectives have similar introduction and elimination rules in the form of $\lambda$ terms abstracting over both the object of the category and the element of the set, and appropriate application forms. To keep with our invariant that the variable occurrences occur left to right in the term syntax in a manner matching the context, we write the covariant application in the usual order $s \triangleright^a t$ where the function is on the left and the argument is on the right, and the contravariant application in the flipped order. We also write the instantiating object as a superscript to de-emphasize it, as in practice it can often be inferred.

Finally, we have the cartesian unit and product sets, which are analogous to the normal unit and product of types. The most notable point to emphasize is that in the formation rule for the product, the two subformulae should have the same covariant and contravariant dependence (as with linear logic, some constructions can syntactically use a variable more than once and still be "linear").

### 2.3 Type Connectives

Finally, we briefly describe the connectives for the "meta-logic", which extends Martin-Löf type theory with $\Pi/\Sigma$ and extensional identity types (with their standard rules). We use extensional identity types so that the description of models is simpler, but intensional identity types could be used instead. The types we include are *universes* for the object categorical logic: types of small categories and locally small categories, functors, profunctors and natural transformations. The rule for the types of small categories and (large) categories are very similar: any definable category defines an element of type Cat, and any element of that type can be reflected back into a category. The only difference for SmallCat is that the categories involved additionally satisfy $\mathbb{C}$ Small. Again we elide the $\beta\eta$

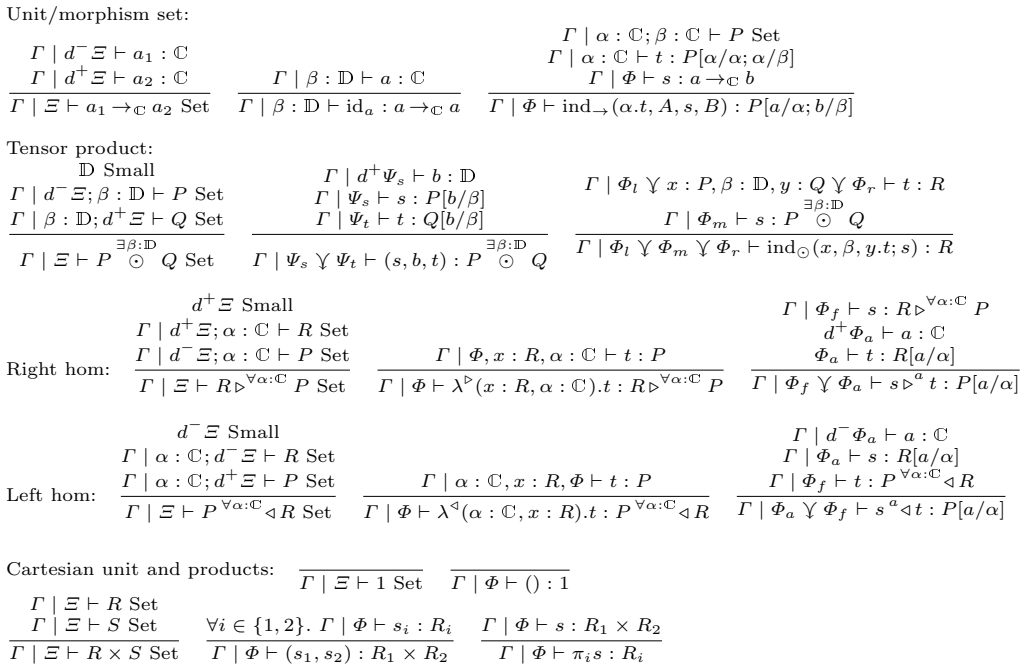



**Fig. 4.** Set Connectives

principles, which state that $\lceil - \rceil$ and $\lfloor - \rfloor$ are mutually inverse. Since every small category $\mathbb{C}$ Small is a category $\mathbb{C}$ Cat, there is a definable inclusion function from SmallCat to Cat and the $\beta\eta$ properties ensure that this is a monomorphism.

Next, we have the types of all functors and profunctors between any two fixed categories. The introduction and elimination forms are those for unary and binary function types respectively, where metalanguage terms of type $\mathrm{Fun}\,\mathbb{C}\,\mathbb{D}$ can be used to construct an object/functor, while metalanguage terms of type $\mathrm{Prof}\,\mathbb{C}\,\mathbb{D}$ can be used to construct a set/profunctor.

Finally we include a type $\forall\alpha : \mathbb{C}.P$ which we call the set of "natural elements" of $P$. The name comes from the case that $P$ is of the form $F(\alpha) \to G(\alpha)$ in which case the type $\forall\alpha : \mathbb{C}.F(\alpha) \to G(\alpha)$ can be interpreted as the set of all natural transformations from $F$ to $G$. More generally this is modeled as an end, and we notate it with a universal quantifier (just as we do for the quantifiers in left/right hom types). Syntactically, $\forall\alpha.P$ is a meta-language type that represents elements/natural transformations with exactly one free variable.

$$\frac{}{\Gamma \vdash \mathrm{SmallCat}} \quad \frac{\Gamma \vdash \mathbb{C}\ \mathrm{Small}}{\Gamma \vdash \lceil \mathbb{C} \rceil : \mathrm{SmallCat}} \quad \frac{\Gamma \vdash M : \mathrm{SmallCat}}{\Gamma \vdash \lfloor M \rfloor\ \mathrm{Small}} \quad \frac{}{\Gamma \vdash \mathrm{Cat}} \quad \frac{\Gamma \vdash \mathbb{C}\ \mathrm{Cat}}{\Gamma \vdash \lceil \mathbb{C} \rceil : \mathrm{Cat}} \quad \frac{\Gamma \vdash M : \mathrm{Cat}}{\Gamma \vdash \lfloor M \rfloor\ \mathrm{Cat}}$$

$$\frac{\Gamma \vdash \mathbb{C}\ \mathrm{Cat} \qquad \Gamma \vdash \mathbb{D}\ \mathrm{Cat}}{\Gamma \vdash \mathrm{Fun}\,\mathbb{C}\,\mathbb{D}\ \mathrm{Type}} \quad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash A : \mathbb{D}}{\Gamma \vdash \lambda\alpha : \mathbb{C}.A : \mathrm{Fun}\,\mathbb{C}\,\mathbb{D}} \quad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash A : \mathbb{D} \qquad \Gamma \vdash M : \mathrm{Fun}\,\mathbb{D}\,\mathbb{E}}{\Gamma \mid \alpha : \mathbb{C} \vdash MA : \mathbb{E}}$$

$$\frac{\Gamma \vdash \mathbb{C}\ \mathrm{Cat} \qquad \Gamma \vdash \mathbb{D}\ \mathrm{Cat}}{\Gamma \vdash \mathrm{Prof}\,\mathbb{C}\,\mathbb{D}\ \mathrm{Type}} \quad \frac{\Gamma \mid \alpha : \mathbb{C}; \beta : \mathbb{D} \vdash R\ \mathrm{Set}}{\Gamma \vdash \lambda\alpha : \mathbb{C}; \beta : \mathbb{D}.R : \mathrm{Prof}\,\mathbb{C}\,\mathbb{D}} \quad \frac{\begin{array}{c}\Gamma \vdash M : \mathrm{Prof}\,\mathbb{C}\,\mathbb{D} \\ \Gamma \mid d^- \Xi \vdash A : \mathbb{C} \\ \Gamma \mid d^+ \Xi \vdash B : \mathbb{C}\end{array}}{\Gamma \mid \Xi \vdash MA\,B\ \mathrm{Set}}$$

$$\frac{\Gamma \mid \alpha : \mathbb{C} \vdash P\ \mathrm{Set}}{\Gamma \vdash \forall\alpha : \mathbb{C}.P\ \mathrm{Type}} \quad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash t : P}{\Gamma \vdash \lambda\alpha.t : \forall\alpha.P} \quad \frac{\Gamma \vdash M : \forall\alpha.P \qquad \Gamma \mid \beta : \mathbb{D} \vdash a : \mathbb{C}}{\Gamma \mid \beta : \mathbb{D} \vdash M^a : P[a/\alpha]}$$

**Fig. 5.** Type Connectives

## 3 Formal Category Theory in VETT

To demonstrate what formal category theory in VETT looks like, we demonstrate some basic definitions and theorems. While it is well known that much category theory can be formalized in virtual equipments, we show these examples to demonstrate how the VETT syntax gives a more familiar syntax to these constructions, while still avoiding the need for explicit naturality and functoriality side conditions. We have mechanized some of the results in this section (e.g. Lemma 2 and Lemma 3 and the maps in Lemma 4) in Agda.[5]

First, we using the elimination for the unit set, we can see that all constructions are (pro-)functorial:

**Construction 1** *For any small category $\mathbb{C}$, we can construct natural elements*

1. *Identity:* $\forall\alpha : \mathbb{C}.\alpha \to_{\mathbb{C}} \alpha$
2. *Composition:* $\forall\alpha_1 : \mathbb{C}.(\alpha_1 \to_{\mathbb{C}} \alpha_2) \triangleright^{\forall\alpha_2:\mathbb{C}} (\alpha_2 \to_{\mathbb{C}} \alpha_3) \triangleright^{\forall\alpha_3:\mathbb{C}} (\alpha_1 \to_{\mathbb{C}} \alpha_3)$
3. *Functoriality: for any* $F : Fun\,\mathbb{C}\,\mathbb{D}$, $\forall\alpha_1 : \mathbb{C}.(\alpha_1 \to_{\mathbb{C}} \alpha_2) \triangleright^{\forall\alpha_2:\mathbb{C}} (F(\alpha_1) \to_{\mathbb{D}} F(\alpha_2))$.
4. *Profunctoriality: for any* $R : Prof\,\mathbb{C}\,\mathbb{D}$ *if* $\mathbb{D}$ *is small then* $\forall\alpha_1 : \mathbb{C}.(\alpha_1 \to_{\mathbb{C}} \alpha_2) \triangleright^{\forall\alpha_2:\mathbb{C}} R\alpha_2\beta_2 \triangleright^{\forall\beta_2:\mathbb{D}} (\beta_2 \to_{\mathbb{D}} \beta_1) \triangleright^{\forall\beta_1:\mathbb{D}} R\alpha_1\beta_1$

Identity and Composition generalize the reflexivity and transitivity properties of equality, respectively, with the lack of symmetry being a key feature of the generalization. In addition, we can prove that the (pro)-functoriality axioms commute with the composition proof by the $\eta$ principle for the unit. (Pro-)Functoriality generalizes the statement that all functions and relations respect equality. Naturality is more complex to state, and it is a statement about the *proofs* so it has no analog in ordinary higher-order logic. The following version is stated for any *profunctor*, with the usual case of naturality arising when $R\alpha\beta = F\alpha \to_{\mathbb{C}} G\beta$.

**Lemma 1 (Naturality).** *For any $t : \forall\alpha : \mathbb{C}.R(\alpha;\alpha)$, by composing with profunctoriality, we can construct terms $\alpha_1 : \mathbb{C}, f : \alpha_1 \to_{\mathbb{C}} \alpha_2, \alpha_2 : \mathbb{C} \vdash lcomp(f, t^{\alpha_2})$ and $rcomp(t^{\alpha_1}, f) : R(\alpha_1;\alpha_2)$ that are both equal to $ind_{\to}(f,t)$.*

[5] https://github.com/maxsnew/virtual-equipments/blob/master/agda/Examples.agda

Next, we turn to some of the central theorems of category theory, the Yoneda and Co-Yoneda lemmas. Despite being ultimately quite elementary, these are notoriously abstract. In VETT, we view these as ordered generalizations of some very simple tautologies about equality. For instance, the Yoneda lemma generalizes the equivalence between the formulae $\forall y.x = y \Rightarrow Py$ and $Px$ for any $x$.

**Lemma 2.** *Let $\alpha : \mathbb{C}$ and $\pi : \mathcal{P}^+\mathbb{C}$. Then*

1. *(Yoneda) The profunctor $(\alpha \to_{\mathbb{C}} \alpha') \triangleright^{\forall \alpha'} (\pi \ni \alpha')$ is isomorphic to $\pi \ni \alpha$*
2. *(Co-Yoneda) The profunctor $(\pi \ni \alpha') \overset{\exists \alpha'}{\odot} (\alpha' \to \alpha)$ is isomorphic to $\pi \ni \alpha$*

The proofs both follow from the unit elimination rule, which is essentially the Yoneda lemma—the two cases of showing (1) is an isomorphism are precisely the $\beta$ and $\eta$ rules for the unit.

Next, we have the "Fubini" theorems, which relate the tensor and hom types. The statement and proofs for these theorems are analogous to proofs relating tensor and hom in ordered logic. For instance, the second isomorphism below is analogous to the equivalence $(P \odot Q) \multimap R \cong P \multimap Q \multimap R$ in ordered logic.

**Lemma 3 (Fubini).** *The following isomorphisms hold when the corresponding profunctors are well typed.*

1. $P(\alpha;\beta) \overset{\exists \beta}{\odot} (Q(\beta;\gamma) \overset{\exists \gamma}{\odot} R(\gamma;\delta)) \cong (P(\alpha;\beta) \overset{\exists \beta}{\odot} Q(\beta;\gamma)) \overset{\exists \gamma}{\odot} R(\gamma;\delta)$
2. $(P(\delta;\beta) \overset{\exists \beta}{\odot} Q(\beta;\gamma)) \triangleright^{\forall \gamma} S(\alpha;\gamma) \cong P(\delta;\beta) \triangleright^{\forall \beta} Q(\beta;\gamma) \triangleright^{\forall \gamma} S(\alpha;\gamma)$
3. $S(\gamma;\delta) \,{}^{\forall \gamma}\triangleleft (P(\gamma;\beta) \overset{\exists \beta}{\odot} Q(\beta;\alpha)) \cong S(\gamma;\delta) \,{}^{\forall \gamma}\triangleleft P(\gamma;\beta) \,{}^{\forall \beta}\triangleleft Q(\beta;\alpha)$
4. $Q(\delta;\gamma) \triangleright^{\forall \gamma} (S(\beta;\gamma) \,{}^{\forall \beta}\triangleleft P(\beta;\alpha)) \cong (Q(\delta;\gamma) \triangleright^{\forall \gamma} S(\beta;\gamma)) \,{}^{\forall \beta}\triangleleft P(\beta;\alpha)$
5. $\forall \alpha.P(\alpha;\beta) \triangleright^{\forall \beta} Q(\alpha;\beta) \cong \forall \beta.Q(\alpha;\beta) \,{}^{\forall \alpha}\triangleleft P(\alpha;\beta)$

*Proof.* We show one case as an example, the forward direction of (1) is given by $\lambda\alpha.\lambda^{\triangleright}(x,\delta).\mathrm{ind}_{\odot}(p,\beta,y.\mathrm{ind}_{\odot}(q,\gamma,r.((p,\beta,q),\gamma,r);y);x)$

Next, we can prove that two definitions of an adjunction are equivalent:

**Lemma 4.** *For $R : Fun\,\mathbb{D}\,\mathbb{C}$ and $L : Fun\,\mathbb{C}\,\mathbb{D}$, the following are in bijection:*

1. *An isomorphism of profunctors $(L\alpha \to_{\mathbb{D}} \beta) \cong (\alpha \to_{\mathbb{C}} R\beta)$*
2. *A unit $\eta : \forall \alpha.\alpha \to_{\mathbb{C}} R(L\alpha)$ and co-unit $\varepsilon : \forall \beta.L(R(\beta)) \to_{\mathbb{D}} \beta$ satisfying triangle identities.*

*Proof.* Given the forward homomorphism lr, we can construct $\eta = \lambda\alpha.\mathrm{lr}^{\alpha} \triangleright^{L\alpha} \mathrm{id}_{\alpha}$. Given the unit we can reconstruct the forward homomorphism using comp (composition) and fctor (functoriality) from Construction 1 as
$\mathrm{comp}^{\alpha} \triangleright^{R(L\alpha)} \eta^{\alpha} \triangleright^{R\beta} (\mathrm{fctor}(R)^{L\alpha} \triangleright^{\beta} f)$.

We can define weighted limits, which as special cases include ordinary limits and Kan extensions.

**Definition 1.** *For a functor $D : Fun\,\mathbb{J}\,\mathbb{C}$ and a profunctor $W : Prof\,\mathbb{K}\,\mathbb{J}$, the limit of $D$ weighted by $W$ is (if it exists) a functor $lim^W D : Fun\,\mathbb{K}\,\mathbb{C}$ with an isomorphism $\alpha \to_{\mathbb{C}} (lim^W D\,)k \cong Wkj \triangleright^{\forall j} (\alpha \to_{\mathbb{C}} Dj)$*

This generalizes the usual definition that a morphism into a limit is a cone over the diagram $(\alpha \to_{\mathbb{C}} Dj)$ to be parameterized by a weight $Wkj$. Then we can prove the well-known theorem that right adjoints preserve (weighted) limits:

**Theorem 1.** *If $lim^W D$ exists and is a limit and $R : Fun\,\mathbb{C}\,\mathbb{C}'$ has a left adjoint $L$, then $\lambda\kappa.R((lim^W D\,)\kappa)$ is the limit of $\lambda j.R(Dj)$ weighted by $W$.*

*Proof.*

$$\gamma \to R((\lim^W D\,)\kappa) \cong L\gamma \to (\lim^W D\,)\kappa \cong Wkj \triangleright^{\forall j} L\gamma \to Dj \cong Wkj \triangleright^{\forall j} \gamma \to R(Dj)$$

This is a high level proof in terms of isomorphisms that may be written in VETT. The first two steps are the instantiation of assumptions (adjointness, weighted limits). The last step uses the fact that a natural isomorphisms lift to natural isomorphism of homs of profunctors. The construction of this isomorphism illustrates how naturality need not be proved explicitly in VETT. For any $\phi : \forall\alpha.R'\alpha\beta \triangleright^{\forall\beta} R\alpha\beta$ and $\psi : \forall\gamma.S\gamma\beta \triangleright^{\forall\beta} S'\gamma\beta$ we can construct a natural transformation $\phi \triangleright \psi : \forall\gamma.(R\alpha\beta \triangleright^{\forall\beta} S\gamma\beta) \triangleright^{\forall\alpha} R'\alpha\beta \triangleright^{\forall\beta} S'\gamma\beta$ as $\lambda\gamma.\lambda^{\triangleright}(f,\alpha).\lambda^{\triangleright}(r,\beta).\psi^{\gamma} \triangleright^{\beta} (f \triangleright^{\beta} (\phi^{\alpha} \triangleright^{\beta} r))$. Furthermore if $\phi$ and $\psi$ have inverses, then $\phi^{-1} \triangleright \psi^{-1}$ is the inverse of $\phi \triangleright \psi$.

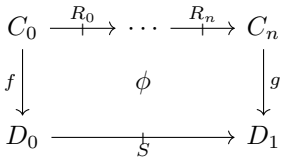
## 4 Semantics

Next, we develop the basics of the model theory for VETT. First, we define a sound and complete notion of categorical model based on hyperdoctrines of virtual equipments. Then we instantiate this general notion of model to show that the VETT can be interpreted in ordinary category theory as well as enriched, internal and indexed notions.

First, we can model the judgmental structure of the unary type theory and predicate logic in *virtual double categories* that are *split fibrant* and have a notion of *small object* [33, 18]. We briefly recount the structure present in a virtual double category, but see [18] for a precise definition of the composition rules for 2-cells and functor of virtual double categories.

**Definition 2.** *A virtual double category $\mathcal{V}$ consists of*

1. *A category $V_o$ of "objects and vertical arrows"*
2. *A set $\mathcal{V}_h$ of "horizontal arrows" with source and target functions $s, t : \mathcal{V}_h \to {\mathcal{V}_o}^2$*
3. *Sets of 2-cells of the following form, with appropriate "multi-categorical" notions of identity and composition:*



*We say that the 2-cell $\phi$ has $S$ as codomain, the sequence $R_0 \dots R_n$ as domain and call $f$ and $g$ the left and right "frames", or that $\phi$ is framed by $f$ and $g$.*

*We say a virtual double category is* split fibrant *when it has a choice of* restrictions*, that is, for any horizontal arrow $R : C \nrightarrow D$ and vertical arrows $f : C' \to C$ and $g : D' \to D$ there is a chosen horizontal arrow $R(f, g) : C' \nrightarrow D'$ with a cartesian 2-cell to $R$ framed by $f, g$ and these chosen cartesian lifts are functorial in $f, g$ ([47]). A* choice of small objects *is a subset of the objects $V_s \subseteq V_o$. A* morphism *of split fibrant virtual double categories with small objects is a functor of the virtual double categories that additionally preserves the restrictions and smallness of objects. This defines a category fVDCs.*

In the presence of restrictions, every 2-cell can be represented as a "globular" 2-cell where the left and right frame are identities [47]. For example the 2-cell $\phi$ above can be represented as one with the same domain but whose codomain is $S(f, g)$. This property is crucial for the completeness of our semantics as we only include a syntax for these globular terms (proof of Construction 2). Each component of this definition has a direct correspondence to a syntactic structure in VETT. The objects of $\mathcal{V}_o$ models the category judgment and the morphisms model the functor judgment. The set $\mathcal{V}_h$ models the profunctor judgment. A composable string $R_0 \cdots R_n$ models the profunctor contexts. The 2-cells correspond to the natural transformation judgment where we have taken the restriction $S(F, G)$ of the codomain. Note that Cruttwell and Shulman define a *virtual equipment* to be a virtual double category with all restrictions and all units. The units are the model of the unit of profunctors connective and so all of our models with the unit will be virtual equipments, hence the name VETT.

To model the dependent type theory and indexing of category-theoretic judgments by a $\Gamma$ with an action of substitution, we use a variation on Lawvere's notion of *hyperdoctrine* for modeling predicate logic[32][6]:

**Definition 3 (VETT Judgmental model).** *A VETT judgmental model ($VM_J$) is a pair of a category with families $\mathcal{C}$ and a functor $V^{(-)} : \mathcal{C}^o \to fVDCs$.*

Categories with families $\mathcal{C}$ model dependent type theory [23] and for each semantic context $\Gamma$, $V^\Gamma$ models the VETT judgments in context $\Gamma$, with the functoriality modeling the fact that all of these judgments admit a well-behaved action of substitution. A $\mathrm{VM}_J$ is then precisely the structure corresponding to the judgments and actions of substitution in VETT.

**Construction 2 (Syntactic Model)** *The syntax of VETT with with any subset of connectives are included presents a $VM_J$.*

*Proof.* Define the category of families using the dependent type structure and the virtual equipment structure having ($\alpha$-equivalence classes of) syntactic categories as objects, functors/sets as vertical/horizontal arrows and interpreting

[6] note that unlike in hyperdoctrines, we do not require *quantifiers* adjoint to substitution

compositions/restrictions as substitutions. The biggest gap between syntax and semantics is in the definition of the 2-cells. A 2-cell from
$(\alpha_1 : \mathbb{C}_1;\alpha_2 : \mathbb{C}_2 \vdash R_1), (\alpha_2 : \mathbb{C}_2;\alpha_3 : \mathbb{C}_3 \vdash R_2), \ldots$ to $(\beta_1 : \mathbb{D}_1;\beta_2 : \mathbb{D}_2 \vdash S)$ with frames $\alpha_1 : \mathbb{C}_1 \vdash b_1 : \mathbb{D}_1$ and $\alpha_n : \mathbb{C}_n \vdash b_2 : \mathbb{D}_2$ is given by a term $x_1 : R_1, x_2 : R_2 \ldots \vdash s : S[b_1/\beta_1; b_2/\beta_2]$. Composition is defined by substitution.

Then the *connectives* of VETT each precisely correspond to a universal construction in a $\mathrm{VM}_J$. The $\Pi, \Sigma, \mathrm{Id}$ types correspond to their standard semantics in a CwF and the connectives for categories and profunctors correspond to universal constructions in the virtual double categories. Products of categories are interpreted as products in the vertical category, and products of sets as products in the category of pro-arrows and 2-cells. The units, tensor and covariant and contravariant homs are modeled by the universal properties of the same names, as described in [47]. The graph of a profunctor is modeled by *tabulators* [26]. Finally, the covariant and contravariant presheaf categories can be described as a weakening of the definition of a Yoneda equipment from [20] to virtual double categories. More detailed descriptions of these universal properties are included in the extended version [38]. Then the soundness and completeness of this notion of categorical model is formalized by the following initiality theorem.

**Theorem 2 (Initiality).** *The syntax of VETT with any subset of connectives that includes the hom types presents a $VM_J$ that is initial in the category of $VM_J$ with the chosen instances of the universal properties and functors that preserve such chosen instances.*

*Proof.* The construction 2 can be extended for any connective modularly, with the exception that the unit relies on the presence of hom sets in order to satisfy the "distributivity" requirement that its elimination can occur in any context. Then we can construct the unique morphism to any HVE induction on syntax.

Now that we have a category-theoretic notion of model, we give some model construction theorems that can be used to justify our intuitive notion of semantics in (enriched, internal, indexed) category theory. First, we can extend any set-theoretic model of the category theoretic judgments to a hyperdoctrine of models where the category of families is the category of sets:

**Construction 3** *Given a $\mathcal{V} \in fVDCs$, we can construct a $VM_J$ $\mathcal{V}^- : Set \to vDbl_r$ by defining of $(\mathcal{V}^\Gamma)_o$ to be functions $\mathcal{V}_o^\Gamma$, and similarly for morphisms and 2-cells with all operations given pointwise.*

Then to define a model of VETT with a collection of connectives it is sufficient to construct a virtual equipment with the corresponding universal properties. The "standard model" is the virtual double category of locally small categories where the small objects are the small categories.

**Construction 4** *Fix a cardinal $\kappa$. The virtual double category $Cat_\kappa$ is defined to have as objects locally $\kappa$-small categories, small objects as $\kappa$-small categories, vertical morphisms as functors, horizontal arrows as functors $\mathbb{C}^o \times \mathbb{D} \to \kappa Set$*

*and 2-cells as morphisms of profunctors. Restriction of profunctors is given by composition, which is strictly associative and unital. $Cat_U$ has objects satisfying the universal properties of all connectives in VETT.*

More generally, categories internal to, enriched in and/or indexed by sufficiently nice categories define a virtual equipment that model the connectives of VETT. We highlight one example from the literature that is highly general: Shulman's enriched indexed categories [48]. Shulman's construction defines a virtual double category of large and small $\mathcal{V}$-categories for any pseudofunctor $\mathcal{V} : S^o \to \mathrm{MonCat}$ where $S$ is a category with finite products. He gives examples that show that this subsumes ordinary internal, enriched and indexed categories for suitable choices of $\mathcal{V}$, as well as more general categories that can be thought of as both indexed and enriched. This is slightly weaker then what we require: to have *split* restrictions, we need that $\mathcal{V}$ be a *strict* functor, not merely a pseudo-functor. This is analogous to the situation for dependent type theory, where syntactic substitution is strictly associative, but semantic substitution is typically given by pullback, which is only associative up to unique isomorphism. Shulman's construction carries over when the functor is strict but some of their example instances would require a strictification theorem.

**Construction 5 (Shulman [48])** *Given any functor $\mathcal{V} : S^o \to SymMonCat$ such that $S$ and $\mathcal{V}$ have sufficiently well-behaved (indexed) $\kappa$-products, then there is a virtual equipment $\mathcal{V} - Cat$ whose objects are locally $\kappa$-small $\mathcal{V}$-categories, small objects are $\kappa$-small $\mathcal{V}$-categories etc. This virtual equipment has objects satisfying all of the universal properties needed for a model of VETT.*

A final model that uses a CwF that is not Set would be given by taking extensional dependent type theory as the CwF and interpreting the category-theoretic constructions by their definitions inside type theory.

## 5 Related and Future Work

We now compare VETT with other calculi for formal category theory.

Cáccamo and Winskel [13] develop a formal language for defining categories, functors (of many variables) and proving existence of natural equivalences between them. Their system can encode profunctors as functors into Set. Their natural equivalence judgment does not have proof terms or equality between equivalences and they do not support natural transformations. Additionally, they only consider ordinary categories as the intended model and do not develop a more general semantics. Riehl and Verity [44] use a formal language of virtual equipments to prove results valid for $\infty$-categories without concrete manipulation of model categories. They formalize this language as a theory in Makkai's framework of first-order logic with dependent sorts (FOLDS). While this previous work has the same models as VETT, we believe that the syntax we propose in this paper formalizes informal arguments more directly, as shown in Section 3. This is because FOLDS approach approach is entirely relational,

whereas we formalize concepts like restriction of a profunctor or composition of natural transformations as functional operations (substitution). In particular, this means that our calculus requires only vertically degenerate squares (elements/natural transformations) as a "user-facing" notion, with general squares occurring only in the admissible substitution operations.

The coend calculus [34] is an informal syntax for manipulating profunctors involving ends and coends; an extension of VETT to treat profunctors of many variables of different variances may provide a formal treatment of it.

Myers [36] provides a string diagram calculus for double categories and proarrow equipments, generalizing string diagrams for monoidal categories. These are an alternative approach to type-theoretic calculi, with the string diagrams typically making tensor products simpler to work with, while a type-theoretic calculus like VETT makes the closed structure $P \triangleright^{\forall \alpha} Q$ simpler to work with by using bound variables.

*Cartesian bicategories* are similar to equipments but they axiomatize the bicategory of profunctors rather than the full double category of functors and profunctors [14]. Frey [24] describes preliminary work on a proof system for Cartesian bicateogires. Their profunctors are more general than in VETT in as they may have 0, 1 or more covariant or contravariant variables. But they do not have a term syntax for functors or natural transformations.

Our work in this paper fits broadly into a line of work on *directed dependent type theories*, a type theory where the identity type is interpreted as morphisms in a (possibly $\infty$-)category. In directed type theories based on a bisimplicial model [43, 12, 56, 55], morphism types are defined using an interval object, like in cubical type theory [9, 17, 5, 4], and universal properties like "morphism induction" are an internally definable property of certain types. Other type theories [39, 1] define morphism types via an induction principle, corresponding to the lifting properties of certain kinds of fibrations of categories. While these previous works can express some constructions on Cat that are not expressible in VETT, because VETT is more restricted, VETT contrariwise has more models, for instance categories enriched in non-cartesian monoidal categories, so the theorems that are provable in VETT apply in more settings.

Finally, some variations on double categories have been used to model the structure of certain program logics. GTT [37] is a logic for *vertically thin* proarrow equipments, where there is at most one vertical arrow or 2-cell of any tyepe, so their calculus does not include functor or transformation judgments. Another similar calculus is System P [22] which is an internal language of *reflexive graph categories*, which are like double categories without horizontal composition.

In future work, VETT could incorporate functor categories by generalizing the unary type theory of functors to functors of many variables, in which case ordinary $\lambda$ calculus can be used to define functor categories as function types, and incorporate multi-variable profunctors as in [24]. This would require to the models to have a monoidal structure. Ideas from coeffects and enriched category theory may be useful for defining opposite categories [49, 11].

**Acknowledgments.** This material is based on research sponsored by the National Science Foundation under agreement number CCF-1909517 and the United States Air Force Research Laboratory under agreement number FA9550-21-0009 (Tristan Nguyen, program manager). The authors would like to thank David Jaz Myers, Emily Riehl, Mike Shulman, Dominic Verity for helpful feedback on this work.

## References

1. Ahrens, B., North, P., van der Weide, N.: Semantics for two-dimensional type theory. In: ACM/IEEE Symposium on Logic in Computer Science (LICS) (2022)
2. Altenkirch, T., Capriotti, P., Kraus, N.: Extending homotopy type theory with strict equality. In: EACSL Annual Conference on Computer Science Logic (CSL) (2016)
3. Altenkirch, T., Kaposi, A.: Type theory in type theory using quotient inductive types. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 18–29. POPL '16 (2016). https://doi.org/10.1145/2837614.2837638
4. Angiuli, C., Brunerie, G., Coquand, T., Hou (Favonia), K.B., Harper, R., Licata, D.R.: Syntax and models of cartesian cubical type theory. Mathematical Structures in Computer Science (2021)
5. Angiuli, C., Hou (Favonia), K.B., Harper, R.: Cartesian cubical computational type theory: Constructive reasoning with paths and equalities. In: Computer Science Logic (CSL) (2018)
6. Awodey, S., Warren, M.: Homotopy theoretic models of identity types. Mathematical Proceedings of the Cambridge Philosophical Society (2009)
7. Bénabou, J.: Introduction to bicategories. In: Reports of the Midwest Category Seminar. pp. 1–77. Springer Berlin Heidelberg, Berlin, Heidelberg (1967)
8. Bénabou, J.: Distributors at work. Lecture notes written by Thomas Streicher **11** (2000)
9. Bezem, M., Coquand, T., Huber, S.: The univalence axiom in cubical sets. Journal of Automated Reasoning (June 2018). https://doi.org/10.1007/s10817-018-9472-6
10. Birkedal, L., Møgelberg, R.E., Schwinghammer, J., Støvring, K.: First steps in synthetic guarded domain theory: step-indexing in the topos of trees. Logical Methods in Computer Science **Volume 8, Issue 4** (Oct 2012). https://doi.org/10.2168/LMCS-8(4:1)2012
11. Brunel, A., Gaboardi, M., Mazza, D., Zdancewic, S.: A core quantitative coeffect calculus. In: Proceedings of the 23rd European Symposium on Programming Languages and Systems - Volume 8410. p. 351–370 (2014). https://doi.org/10.1007/978-3-642-54833-8˙19
12. Buchholtz, U., Weinberger, J.: Synthetic fibered $(\infty,1)$-category theory, higher Structures, to appear. arXiv:2105.01724
13. Cáccamo, M., Winskel, G.: A higher-order calculus for categories. In: Boulton, R.J., Jackson, P.B. (eds.) Theorem Proving in Higher Order Logics. pp. 136–153. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
14. Carboni, A., Walters, R.: Cartesian bicategories i. Journal of Pure and Applied Algebra **49**(1), 11–32 (1987). https://doi.org/https://doi.org/10.1016/0022-4049(87)90121-6

15. Cartmell, J.: Generalised algebraic theories and contextual categories. Annals of Pure and Applied Logic **32**, 209–243 (1986). https://doi.org/https://doi.org/10.1016/0168-0072(86)90053-9
16. Cervesato, I., Pfenning, F.: A linear logical framework. Information and Computation **179**(1), 19–75 (2002)
17. Cohen, C., Coquand, T., Huber, S., Mörtberg, A.: Cubical type theory: A constructive interpretation of the univalence axiom. In: Uustalu, T. (ed.) 21st International Conference on Types for Proofs and Programs (TYPES 2015). pp. 5:1–5:34 (2018). https://doi.org/10.4230/LIPIcs.TYPES.2015.5
18. Crutwell, G., Shulman, M.A.: A unified framework for generalized multicategories. Theory and Applications of Categories **24**, 580–655 (2010)
19. Curien, P.L.: Categorical combinators. Information and Control **69**(1), 188–254 (1986). https://doi.org/https://doi.org/10.1016/S0019-9958(86)80047-X
20. Di Liberti, I., Loregian, F.: On the unicity of formal category theories (2019). https://doi.org/10.48550/ARXIV.1901.01594
21. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. In: 2009 24th Annual IEEE Symposium on Logic In Computer Science. pp. 71–80 (2009). https://doi.org/10.1109/LICS.2009.34
22. Dunphy, B.P., Reddy, U.S.: Parametric limits. In: 19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings. pp. 242–251 (2004). https://doi.org/10.1109/LICS.2004.1319618
23. Dybjer, P.: Internal type theory. In: Berardi, S., Coppo, M. (eds.) Types for Proofs and Programs. pp. 120–134. Springer Berlin Heidelberg (1996)
24. Frey, J.: A language for closed cartesian bicategories (2019), category Theory 2019
25. Girard, J.Y.: Linear logic. Theoretical Computer Science **50**(1), 1–101 (1987). https://doi.org/https://doi.org/10.1016/0304-3975(87)90045-4
26. Grandis, M., Pare, R.: Limits in double categories. Cahiers de Topologie et Géométrie Différentielle Catégoriques **40**(3), 162–220 (1999)
27. Hofmann, M., Streicher, T.: The groupoid interpretation of type theory. In: Twenty-five years of constructive type theory. Oxford University Press (1998)
28. Isaev, V.: Indexed type theories. Mathematical Structures in Computer Science **31**(1), 3–63 (2021). https://doi.org/10.1017/S0960129520000092
29. Jung, R., Swasey, D., Sieczkowski, F., Svendsen, K., Turon, A., Birkedal, L., Dreyer, D.: Iris: Monoids and invariants as an orthogonal basis for concurrent reasoning. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 637–650. POPL '15, Association for Computing Machinery (2015). https://doi.org/10.1145/2676726.2676980
30. Krishnaswami, N.R., Pradic, P., Benton, N.: Integrating dependent and linear types. In: ACM Symposium on Principles of Programming Languages (2015)
31. Lambek, J., Scott, P.: Introduction to Higher-Order Categorical Logic. Cambridge University Press (1988)
32. Lawvere, F.W.: Adjointness in foundations. Dialectica **23** (1969)
33. Leinster, T.: Generalized enrichment of categories. Journal of Pure and Applied Algebra **168**(2), 391–406 (2002). https://doi.org/https://doi.org/10.1016/S0022-4049(01)00105-0, category Theory 1999: selected papers, conference held in Coimbra in honour of the 90th birthday of Saunders Mac Lane
34. Loregian, F.: (Co)end Calculus. London Mathematical Society Lecture Note Series, Cambridge University Press (2021). https://doi.org/10.1017/9781108778657
35. Moggi, E.: Notions of computation and monads. Information and Computation **93**(1), 55–92 (1991). https://doi.org/https://doi.org/10.1016/0890-

5401(91)90052-4, selections from 1989 IEEE Symposium on Logic in Computer Science
36. Myers, D.J.: String diagrams for double categories and equipments (2016). https://doi.org/10.48550/ARXIV.1612.02762
37. New, M.S., Licata, D.R.: Call-by-Name Gradual Type Theory. In: Kirchner, H. (ed.) 3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 108, pp. 24:1–24:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.FSCD.2018.24
38. New, M.S., Licata, D.R.: A formal logic for formal category theory (extended version) (2022). https://doi.org/10.48550/ARXIV.2210.08663, https://arxiv.org/abs/2210.08663
39. North, P.R.: Towards a directed homotopy type theory. In: Mathematical Foundations of Programming Semantics (MFPS) (2019)
40. Palmgren, E.: Categories with families and first-order logic with dependent sorts. Annals of Pure and Applied Logic **170**(12), 102715 (2019). https://doi.org/https://doi.org/10.1016/j.apal.2019.102715, https://www.sciencedirect.com/science/article/pii/S0168007219300727
41. Plotkin, G., Abadi, M.: A logic for parametric polymorphism. In: Bezem, M., Groote, J.F. (eds.) Typed Lambda Calculi and Applications. pp. 361–375. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)
42. Polakow, J., Pfenning, F.: Natural deduction for intuitionistic non-communicative linear logic. In: Girard, J. (ed.) Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1581, pp. 295–309. Springer (1999). https://doi.org/10.1007/3-540-48959-2"21
43. Riehl, E., Shulman, M.: A type theory for synthetic $\infty$-categories. Higher Structures **1**(1) (2018)
44. Riehl, E., Verity, D.: Elements of $\infty$-Category Theory. Cambridge Studies in Advanced Mathematics, Cambridge University Press (2022). https://doi.org/10.1017/9781108936880
45. Robinson, E., Rosolini, G.: Reflexive graphs and parametric polymorphism. In: Proceedings Ninth Annual IEEE Symposium on Logic in Computer Science. pp. 364–371 (1994). https://doi.org/10.1109/LICS.1994.316053
46. Seely, R.A.G.: Locally cartesian closed categories and type theory. Mathematical Proceedings of the Cambridge Philosophical Society **95**(1), 33–48 (1984). https://doi.org/10.1017/S0305004100061284
47. Shulman, M.: Framed bicategories and monoidal fibrations. Theory and Applications of Categories **20**, 650–738 (2008), http://www.tac.mta.ca/tac/volumes/20/18/20-18abs.html
48. Shulman, M.: Enriched indexed categories. Theory and Applications of Categories **28**, 616–695 (2013), http://www.tac.mta.ca/tac/volumes/28/21/28-21abs.html
49. Shulman, M.: Contravariance through enrichment. Theory and Applications of Categories **33**, 95–130 (2018), http://tac.mta.ca/tac/volumes/33/5/33-05abs.html
50. Street, R., Walters, R.: Yoneda structures on 2-categories. Journal of Algebra **50**(2), 350–379 (1978). https://doi.org/https://doi.org/10.1016/0021-8693(78)90160-6, https://www.sciencedirect.com/science/article/pii/0021869378901606
51. Voevodsky, V.: A very short note on homotopy $\lambda$-calculus (September 2006), unpublished.

52. Voevodsky, V.: A type system with two kinds of identity types (2013), talk at Andre Joyal's 70th birthday conference (IAS)
53. Vákár, M.: A categorical semantics for linear logical frameworks. In: Foundations of Software Science and Computation Structures (FoSSaCS) (2015)
54. Wand, M.: Fixed-point constructions in order-enriched categories. Theoretical Computer Science **8**(1), 13–30 (1979). https://doi.org/https://doi.org/10.1016/0304-3975(79)90053-7
55. Weaver, M.Z., Licata, D.R.: A constructive model of directed univalence in bicubical sets. In: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 915–928. LICS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3373718.3394794
56. Weinberger, J.: A Synthetic Perspective on (∞,1)-Category Theory: Fibrational and Semantic Aspects. Ph.D. thesis, TU Darmstadt (2022), arXiv:2202.13132
57. Wood, R.J.: Abstract pro arrows I. Cahiers de Topologie et Géométrie Différentielle Catégoriques **23**(3), 279–290 (1982)

# A Details of VETT Syntax and Syntactic Metatheory

## A.1 Contexts and Substitutions

In Figure 6 we include the formation rules and definitions of the three kinds of contexts that are used in VETT. In Figure 7 we give definitions for well-typedness of the corresponding three kinds of substitutions.

$$\Gamma \text{ type context } \textsc{TyCtxForm} \qquad \cdot \text{ type context } \textsc{MtTyCtx}$$

$$\frac{\Gamma \text{ type context} \qquad \Gamma \vdash A \text{ Type}}{\Gamma, X : A \text{ type context}} \textsc{TyCtxExt}$$

$$\frac{\Gamma \text{ type context}}{\Gamma \vdash \Xi \text{ set context}} \textsc{BoundaryForm} \qquad \frac{\Gamma \vdash \mathbb{C} \text{ Cat}}{\Gamma \vdash \alpha : \mathbb{C} \text{ set context}} \textsc{BoundarySingle}$$

$$\frac{\Gamma \vdash \mathbb{C} \text{ Cat} \qquad \Gamma \vdash \mathbb{D} \text{ Cat}}{\Gamma \vdash \alpha : \mathbb{C}; \beta : \mathbb{D} \text{ set context}} \textsc{BoundaryDbl}$$

$$\frac{\Gamma \text{ type context}}{\Gamma \vdash \Phi \text{ trans. context}} \textsc{TransCtxForm} \qquad \frac{\Gamma \vdash \mathbb{C} \text{ Cat}}{\Gamma \vdash \alpha : \mathbb{C} \text{ trans. context}} \textsc{TransCtxMt}$$

$$\frac{\Gamma \vdash \Phi \text{ trans. context} \qquad \Gamma \mid d^{+}\Phi; \beta : \mathbb{D} \vdash R \text{ Set}}{\Gamma \vdash \Phi, x : R, \beta : \mathbb{D} \text{ trans. context}} \textsc{TransCtxExt}$$

**Fig. 6.** Contexts

These definitions involve several operations $d^{\pm}\Phi$, $\underline{\Phi}$ and $\Phi \curlyvee \Psi$ on contexts (and their functorial lift to substitutions) that we now define.

$$\begin{aligned}\text{Term Substitution } \gamma, \delta &::= \cdot \mid \gamma, M/X \\ \text{Object Substitution } \xi, \zeta &::= a/\alpha \mid a/\alpha; b/\beta \\ \text{Transformation Substitution } \phi, \psi &::= a/\alpha \mid \phi, s/x, \psi\end{aligned}$$

$$\frac{\Delta \text{ type context} \qquad \Gamma \text{ type context}}{\Delta \vdash \gamma :: \Gamma}\ \textsc{TermSubstFormation}$$

$$\Delta \vdash \cdot :: \cdot\ \textsc{TermSubstMt} \qquad \frac{\Delta \vdash \gamma :: \Gamma \qquad \Delta \vdash M : A[\gamma]}{\Delta \vdash \gamma, M/X :: \Gamma, X : A}\ \textsc{TermSubstExt}$$

$$\frac{\Gamma \vdash Z \text{ set context} \qquad \Gamma \vdash \Xi \text{ set context}}{\Gamma \mid Z \vdash \xi :: \Xi}\ \textsc{BoundarySubstFormation}$$

$$\frac{\Gamma \mid \alpha : \mathbb{C} \vdash b : \mathbb{D}}{\Gamma \mid \alpha : \mathbb{C} \vdash b/\beta :: \beta : \mathbb{D}}\ \textsc{BoundarySubstSingle}$$

$$\frac{\Gamma \mid d^- \Xi \vdash a : \mathbb{C} \qquad \Gamma \mid d^+ \Xi \vdash b : \mathbb{D}}{\Gamma \mid \Xi \vdash a/\alpha; b/\beta :: (\alpha : \mathbb{C}; \beta : \mathbb{D})}\ \textsc{BoundarySubstDbl}$$

$$\frac{\Gamma \vdash \Psi \text{ trans. context} \qquad \Gamma \vdash \Phi \text{ trans. context}}{\Gamma \mid \Psi \vdash \phi :: \Phi}\ \textsc{ElementSubstFormation}$$

$$\frac{\Gamma \mid \beta : \mathbb{C} \vdash a : \mathbb{C}}{\Gamma \mid \beta : \mathbb{C} \vdash a/\alpha : (\alpha : \mathbb{C})}\ \textsc{ElementSubstMt}$$

$$\frac{\Gamma \mid \Psi \vdash \phi :: \Phi \qquad d^+\Psi = d^-\Psi'\Gamma \mid \Psi' \vdash t : R[d^+\phi; b/\beta] \qquad \Gamma \mid d^+\Psi' \vdash b : \mathbb{D}}{\Gamma \mid \Psi \curlyvee \Psi' \vdash \phi, t/x, b/\beta :: (\Phi, x : R, \beta : \mathbb{D})}\ \textsc{ElementSubstExt}$$

**Fig. 7.** Substitution

**Definition 4.** *We define operations $d^{\pm}$ that project out the covariant and contravariant boundary of a set context. This can be typed with the admissible rule*

$$\frac{\Gamma \vdash \Phi \text{ trans. context}}{\Gamma \vdash d^{\pm}\Phi \text{ set context}} \ (*)$$

*This is defined as*

$$\begin{aligned} d^{\pm}(\alpha : \mathbb{C}) &= \alpha : \mathbb{C} \\ d^{-}(\Phi, x : R, \Psi) &= d^{-}\Phi \\ d^{+}(\Phi, x : R, \Psi) &= d^{-}\Psi \end{aligned}$$

*This operation extends to the substitutions with admissible rule*

$$\frac{\Gamma \mid \Psi \vdash \phi :: \Phi}{\Gamma \mid d^{\pm}\Psi \vdash d^{\pm}\phi :: d^{\pm}\Phi} \ (*)$$

*defined as*

$$\begin{aligned} d^{\pm}(a/\alpha) &= a/\alpha \\ d^{-}(\phi, t/x, \psi) &= d^{-}\phi \\ d^{+}(\phi, t/x, \psi) &= d^{+}\psi \end{aligned}$$

Note that $d^{\pm}\Phi$ will always be a set context with a single variable $\alpha : \mathbb{C}$—we exploit the fact that we have these singleton set contexts to avoid introducing a separate syntactic class of category contexts $\alpha : \mathbb{C}$ and substitutions between them.

**Definition 5.** *We define the operation of restricting a set context to both sides of its boundary with admissible typing*

$$\frac{\Gamma \vdash \Phi \text{ trans. context}}{\Gamma \vdash \underline{\Phi} \text{ set context}} \ (*)$$

*and definition*

$$\begin{aligned} \underline{\alpha : \mathbb{C}} &= \alpha : \mathbb{C} \\ \underline{\Phi, x : R, \Psi} &= d^{-}\Phi; d^{+}\Psi \end{aligned}$$

*The extension to substitutions has admissible typing*

$$\frac{\Gamma \mid \Psi \vdash \phi :: \Phi}{\Gamma \mid \underline{\Psi} \vdash \underline{\phi} : \underline{\Phi}} \ (*)$$

*and definition*

$$\begin{aligned} \underline{b/\beta} &= b/\beta \\ \underline{\phi} &= d^{-}\phi; d^{+}\phi \text{ otherwise} \end{aligned}$$

Finally, we define the operation of "horizontal composition" of set contexts $\Phi \between \Psi$ and its functorial lift $\phi \between \psi$.

**Definition 6.** *We define horizontal composition of transformation contexts with the admissible typing rule*

$$\frac{\Gamma \vdash \Phi \text{ trans. context} \qquad \Gamma \vdash \Psi \text{ trans. context} \qquad d^+\Phi = d^-\Psi}{\Gamma \vdash \Phi \between \Psi \text{ trans. context}} \ (*)$$

*as follows*

$$\Phi \between \alpha : \mathbb{C} = \Phi$$
$$\Phi \between (\Psi, x : R, \alpha : \mathbb{C}) = (\Phi \between \Psi), x : R, \alpha : \mathbb{C}$$

*And we extend this to an operation on substitutions with the admissible rule*

$$\frac{\Gamma \mid \Psi \vdash \phi : \Phi \qquad \Gamma \mid \Psi' \vdash \phi' : \Phi' \qquad d^+\phi = d^-\phi'}{\Gamma \mid \Psi \between \Psi' \vdash \phi \between \phi' :: \Phi \between \Phi'} \ (*)$$

*Defined as follows*

$$\phi \between a/\alpha = \phi$$
$$\phi \between (\psi, s/x, a/\alpha) = (\phi \between \psi), s/x, a/\alpha$$

**Lemma 5 (Horizontal Category of Contexts/Substitutions).** *Horizontal composition of contexts is associative (when defined)*

$$(\Phi \between \Psi) \between \Sigma = \Phi \between (\Psi \between \Sigma)$$

*and unital with identity for $\mathbb{C}$ given by the single category variable context $\alpha : \mathbb{C}$:*

$$\alpha : \mathbb{C} \between \Phi = \Phi = \Phi \between \beta : \mathbb{D}$$

*when $d^-\Phi = \alpha : \mathbb{C}$ and $d^+\Phi = \beta : \mathbb{D}$.*

*These properties extend to the horizontal composition of element substitutions:*

$$\phi \between (\psi \between \sigma) = (\phi \between \psi) \between \sigma$$

*where the identity is the single variable substitution:*

$$a/\alpha \between \phi = \phi = \phi \between b/\beta$$

*when $d^-\phi = a/\alpha$ and $d^+\phi = b/\beta$.*

Next, we define the actions of substitutions on terms. We elide the obvious action of term substitutions $\gamma$ and include only the more unusual substructural substitutions.

**Definition 7 (Substitution Actions).** *For any $\Gamma \mid \alpha : \mathbb{C} \vdash a : \mathbb{D}$ and $\Gamma \mid \beta : \mathbb{D} \vdash b : \mathbb{E}$, we define $\Gamma \mid \alpha : \mathbb{C} \vdash b[a/\beta] : \mathbb{E}$ by recursion on $b$:*

$$
\begin{aligned}
\beta[a/\beta] &= a \\
(M\, b)[a/\beta] &= M\, (b[a/\beta]) \\
(b_1, b_2)[a/\beta] &= (b_1[a/\beta], b_2[a/\beta]) \\
(\pi_i b)[a/\beta] &= \pi_i b[a/\beta] \\
()[a/\beta] &= () \\
(\pi_\pm b)[a/\beta] &= \pi_\pm b[a/\beta] \\
(b_-, b_+, s)[a/\beta] &= (b_-[a/\beta], b_+[a/\beta], s[a/\beta]) \\
(\lambda\alpha.R)[a/\beta] &= \lambda\alpha.R[a/\beta]
\end{aligned}
$$

*Simultaneously, for $\Gamma \mid \Psi \vdash \phi : \Phi$ and $\Gamma \mid \Phi \vdash s : R$ we define $\Gamma \mid \Psi \vdash s[\phi] : R[\underline{\phi}]$ by recursion on $s$:*

$$
\begin{aligned}
x[a/\alpha, t/x, b/\beta] &= t \\
M^b[a/\alpha] &= M^{b[a/\alpha]} \\
ind_{\to}(\alpha.t, b_1, s, b_2)[\phi] &= ind_{\to}(\alpha.t, b_1[d^-\phi], s[\phi], b_2[d^+\phi]) \\
(id_b)[a/\alpha] &= id_{b[a/\alpha]} \\
ind_{\odot}(x, \beta, y.r; s)[\phi_l \between \phi_m \between \phi_r] &= ind_{\odot}(x, \beta, y.r[\phi_l, x/x, \beta/\beta \between \beta/\beta, y/y, \phi_r]; s[\phi_m]) \\
(s, b, t)[\phi_s \between \phi_t] &= (s[\phi_s], b[d^+\phi_s], t[\phi_t]) \\
(s \triangleright^a t)[\phi_f \between \phi_a] &= s[\phi_f] \triangleright^{a[d^+\phi_f]} t[\phi_a] \\
(\lambda^{\triangleright}(x, \alpha).s)[\phi] &= \lambda^{\triangleright}(x, \alpha).s[\phi, x/x, \alpha/\alpha] \\
(s\, {}^a\!\triangleleft t)[\phi_a \between \phi_f] &= s[\phi_f]\, {}^{a[d^-\phi_f]}\!\triangleleft t[\phi_a] \\
(\lambda^{\triangleleft}(\alpha, x).s)[\phi] &= \lambda^{\triangleleft}(\alpha, x).s[\alpha/\alpha, x/x, \phi] \\
(\pi_i s)[\phi] &= \pi_i s[\phi] \\
(s_1, s_2)[\phi] &= (s_1[\phi], s_2[\phi]) \\
()[\phi] &= ()
\end{aligned}
$$

*Several rules assume the substitution is in a particular form, such as the tensor elimination which expects an input context $\phi_s \between \phi_t$. The fact that the context can be uniquely decomposed in a well-typed way follows from an inversion principle (Lemma 6) for well-typed substitutions.*

*And finally, for $\Gamma \mid \Xi' \vdash \xi : \Xi$ and $\Gamma \mid \Xi \vdash P$ Set, we define $\Gamma \mid \Xi' \vdash P[\xi]$ Set by recursion on $P$:*

$$\begin{aligned}
(M\, a\, b)[\xi] &= M\, (a[d^- \xi])\, (b[d^+ \xi]) \\
(a \to_{\mathbb{C}} b)[\xi] &= a[d^- \xi] \to_{\mathbb{C}} b[d^+ \xi] \\
(P \overset{\exists \beta}{\odot} Q)[\xi] &= P[d^- \xi; \beta/\beta] \overset{\exists \beta}{\odot} P[\beta/\beta; d^+ \xi] \\
(R \triangleright^{\forall \alpha} P)[\xi] &= R[d^+ \xi; \alpha/\alpha] \triangleright^{\forall \alpha} P[d^- \xi; \alpha/\alpha] \\
(Q \, {}^{\forall \alpha}\!\triangleleft P)[\xi] &= Q[\alpha/\alpha; d^+ \xi] \, {}^{\forall \alpha}\!\triangleleft P[\alpha/\alpha; d^- \xi] \\
1[\xi] &= 1 \\
(P_1 \times P_2)[\xi] &= P_1[\xi] \times P_2[\xi]
\end{aligned}$$

**Lemma 6 (Inversion).**

1. *If $\Phi \vdash \psi :: (\alpha : \mathbb{C})$ then $\Phi = \beta : \mathbb{D}$ for some $\mathbb{D}$ and $\psi = a/\alpha$ where $\beta : \mathbb{D} \vdash a : \mathbb{C}$.*
2. *If $\Phi \vdash \psi :: \Psi_1 \between \Psi_2$, then there exists unique $\Phi_1, \Phi_2, \psi_1, \psi_2$ such that $\Phi = \Phi_1 \between \Phi_2$ and $\Phi_1 \vdash \psi_1 :: \Psi_1$ and $\Phi_2 \vdash \psi_2 : \Psi_2$ and $\psi = \psi_1 \between \psi_2$.*

### A.2 Equational Theory

Next we present the $\beta\eta$ rules that generate the equational theory of the terms. In keeping with the extensional style of the type theory, we do not present explicit transitivity, congruence, or transport rules, but rather consider these as inherent to the notion of equality. This can be formalized by modeling the terms of our type theory as a quotient inductive inductive type [3]. We elide the types on the $\beta$ rules, as they can be inferred from the shape of the term, but include them for clarity on the $\eta$ rules.

### A.3 Generalized Unit Elimination

The unit elimination rule presented in Section 2 is more restrictive than universal property of a unit in a virtual double category that we use in the semantics. So in order for our calculus to be complete for virtual equipments with units, we need to show that the more general unit elimination principle is admissible and satisfies the correct $\beta\eta$ rules.

The more general rules are as follows

$$\frac{\Phi[\alpha/\alpha_-] \between \Psi[\alpha/\alpha_+] \vdash t : P[\alpha/\alpha_-; \alpha/\alpha_+]}{\Phi, x : \alpha_- \to \alpha_+, \Psi \vdash \mathrm{ind}_{\to}^{\Phi;\Psi}(\alpha.t; x) : P} \textsc{UnitElim}$$

$$(\mathrm{ind}_{\to}^{\Phi;\Psi}(\alpha.t; x))[\phi, \mathrm{id}_\alpha, \psi] = t[\phi \between \psi]$$

$$\frac{\Phi, x : \alpha_- \to \alpha_+, \Psi \vdash s : P}{\Phi, x : \alpha_- \to \alpha_+, \Psi \vdash s = \mathrm{ind}_{\to}^{\Phi;\Psi}(\alpha.s[\alpha/\alpha_-, \mathrm{id}_\alpha/x, \alpha/\alpha_+]; x) : P}$$

$$\frac{}{\lfloor\lceil\mathbb{C}\rceil\rfloor = \mathbb{C}}\ \textsc{SmallCat}\beta \qquad \frac{\Gamma \vdash M : \mathrm{SmallCat}}{\Gamma \vdash \lceil\lfloor M\rfloor\rceil = M : \mathrm{SmallCat}}\ \textsc{SmallCat}\eta$$

$$\frac{}{\lfloor\lceil\mathbb{C}\rceil\rfloor = \mathbb{C}}\ \textsc{Cat}\beta \qquad \frac{\Gamma \vdash M : \mathrm{Cat}}{\Gamma \vdash \lceil\lfloor M\rfloor\rceil = M : \mathrm{Cat}}\ \textsc{Cat}\eta$$

$$\frac{}{(\lambda\alpha.b)\, a = b[a/\alpha]}\ \textsc{Fctor}\beta \qquad \frac{\Gamma \vdash M : \mathrm{Fun}\,\mathbb{C}\,\mathbb{D}}{\Gamma \vdash M = \lambda\alpha.M\,\alpha : \mathrm{Fun}\,\mathbb{C}\,\mathbb{D}}\ \textsc{Fctor}\eta$$

$$\frac{}{(\lambda\alpha\beta.P)\, a\, b = P[a/\alpha; b/\beta]}\ \textsc{Prof}\beta \qquad \frac{\Gamma \vdash M : \mathrm{Prof}\,\mathbb{C}\,\mathbb{D}}{\Gamma \vdash M = \lambda\alpha\,\beta.M\,\alpha\,\beta : \mathrm{Prof}\,\mathbb{C}\,\mathbb{D}}\ \textsc{Prof}\eta$$

$$\frac{}{(\lambda\alpha.s)a = s[a/\alpha]}\ \textsc{NatElt}\beta \qquad \frac{\Gamma \vdash M : \forall\alpha.P}{\Gamma \vdash M = \lambda\alpha.M\,a : \forall\alpha.P}\ \textsc{NatElt}\eta$$

$$\frac{}{(\lambda\alpha.R) \in a = R[a/\alpha]}\ \textsc{NegPresheaf}\beta \qquad \frac{\Gamma \mid \beta : \mathbb{D} \vdash p : \mathcal{P}^{-}\mathbb{C}}{p = \lambda\alpha.p \in \alpha}\ \textsc{NegPresheaf}\eta$$

$$\frac{}{(\lambda\beta.R) \ni b = R[b/\beta]}\ \textsc{NegPresheaf}\beta \qquad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash p : \mathcal{P}^{+}\mathbb{D}}{p = \lambda\beta.p \ni \beta}\ \textsc{PosPresheaf}\eta$$

$$\frac{}{\pi_{-}(a_{-}, a_{+}, s) = a_{-}}\ \textsc{Graph}\beta- \qquad \frac{}{\pi_{+}(a_{-}, a_{+}, s) = a_{+}}\ \textsc{Graph}\beta+$$

$$\frac{}{\pi_e(a_{-}, a_{+}, s) = s}\ \textsc{Graph}\beta e \qquad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash b : \sum_{\beta_{-};\beta_{+}} P}{\Gamma \mid \alpha : \mathbb{C} \vdash b = (\pi_{-}b, \pi_{+}b, \pi_e b) : \sum_{\beta_{-};\beta_{+}} P}\ \textsc{Graph}\eta$$

$$\frac{\Gamma \mid \alpha : \mathbb{C} \vdash a : 1}{\Gamma \mid \alpha : \mathbb{C} \vdash a = () : 1}\ 1\eta$$

$$\frac{}{\pi_i(a_1, a_2) = a_i}\ \times\beta \qquad \frac{\Gamma \mid \alpha : \mathbb{C} \vdash a : \mathbb{D}_1 \times \mathbb{D}_2}{\Gamma \mid \alpha : \mathbb{C} \vdash a = (\pi_1 a, \pi_2 a) : \mathbb{D}_1 \times \mathbb{D}_2}\ \times\eta$$

**Fig. 8.** $\beta\eta$ Equality for type and object connectives

$$\frac{}{(\lambda^{\triangleright}(x,\alpha).s)\,\triangleright^{a}\,t = s[t/x, a/\alpha]}\ \textsc{CovHom}\beta \qquad \frac{\Gamma \mid \Phi \vdash s : R \triangleright^{\forall\alpha} P}{\Gamma \mid \Phi \vdash s = \lambda^{\triangleright}(x,\alpha).s \triangleright^{\alpha} x}\ \textsc{CovHom}\eta$$

$$\frac{}{s\ ^{a}\triangleleft(\lambda^{\triangleleft}(\alpha,x).t) = t[a/\alpha, s/x]}\ \textsc{ConHom}\beta \qquad \frac{\Gamma \mid \Phi \vdash t : P\ ^{\forall\alpha}\triangleleft R}{\Gamma \mid \Phi \vdash t = \lambda^{\triangleleft}(\alpha,x).x\ ^{\alpha}\triangleleft t : P\ ^{\forall\alpha}\triangleleft R}\ \textsc{ConHom}\eta$$

$$\frac{}{(\mathrm{ind}_{\to}(\alpha.t, a, \mathrm{id}_a, a)) = t[a/\alpha]}\ \textsc{Unit}\beta$$

$$\frac{\Gamma \mid \alpha_1 : \mathbb{C}, z : \alpha_1 \to_{\mathbb{C}} \alpha_2, \alpha_2 : \mathbb{C} \vdash s : R}{\Gamma \mid \alpha_1 : \mathbb{C}, z : \alpha_1 \to_{\mathbb{C}} \alpha_2, \alpha_2 : \mathbb{C} \vdash s = \mathrm{ind}_{\to}(\alpha.s[\alpha/\alpha_1; \mathrm{id}_\alpha/z, \alpha/\alpha_2], \alpha_1, z, \alpha_2) : R}\ \textsc{Unit}\eta$$

$$\frac{}{\mathrm{ind}_{\odot}(x, \beta, y.r; (s, b, t)) = t[s/x; b/\beta; t/y]}\ \textsc{Tensor}\beta$$

$$\frac{\Gamma \mid \Phi, z : P \overset{\exists\beta}{\odot} Q, \Psi \vdash s : R}{\Gamma \mid \Phi, z : P \overset{\exists\beta}{\odot} Q, \Psi \vdash s = \mathrm{ind}_{\odot}(x, \beta, y.s[(x, \beta, y)/z]; s) : P \overset{\exists\beta}{\odot} Q}\ \textsc{Tensor}\eta$$

$$\frac{\Phi \vdash s : 1}{\Phi \vdash s = () : 1}\ 1\eta$$

$$\frac{}{\pi_i(s_1, s_2) = s_i}\ \times\beta \qquad\qquad \frac{\Phi \vdash s : P \times Q}{\Phi \vdash s = (\pi_1 s, \pi_2 s) : P \times Q}\ \times\eta$$

**Fig. 9.** $\beta\eta$ Equality for set connectives

The rule is more general because it allows the elimination of an input of the unit type with non-trivial contexts $\Phi, \Psi$ surrounding it, whereas the rule presented earlier would only allow this elimination if $x$ were the only variable. We did not include this more general rule as a basic inference rule because it requires an additional explicit substitution for the context $\Phi, x : \alpha_- \to \alpha_+, \Psi$, which would require making the substitutions part of the basic syntax. In the presence of hom types, we can prove this more general elimination is admissible, because the judgment

$$\Phi, x : \alpha_- \to \alpha_+, \Psi \vdash s : P$$

is in natural bijection with the judgment

$$\alpha_-, x : \alpha_- \to \alpha_+, \alpha_+ \vdash s : \Phi \triangleright P \triangleleft \Psi$$

where $\Psi \triangleright P \triangleleft \Phi$ (note the reversal of order) is a type constructed by recursion on $\Phi$ and $\Psi$ using uses the hom types. The function applications for the hom types then provide a more lightweight way to incorporate the explicit substitution into the definition of the type theory.

**Definition 8 (Generalized Unit Elimination).** *We define* $ind_{\to}^{\Phi;\Psi}(\alpha.t; x)$ *by induction on* $\Phi/\Psi$*.*

$$\begin{aligned} ind_{\to}^{\alpha_-;\alpha_+}(\alpha.t; x) &= ind_{\to}(\alpha.t, \alpha_-, x, \alpha_+) \\ ind_{\to}^{\alpha_-;\Psi,y,\beta}(\alpha.t; x) &= (ind_{\to}^{\alpha_-;\Psi}(\alpha.\lambda^{\triangleright}(y, \beta).t; x)) \triangleright^{\beta} y \\ ind_{\to}^{\beta,y,\Phi;\Psi}(\alpha.t; x) &= y \,{}^{\beta}\!\triangleleft (ind_{\to}^{\alpha_-;\Psi}(\alpha.\lambda^{\triangleleft}(\beta, y).t; x)) \end{aligned}$$

**Lemma 7 (Generalized Unit Elim $\beta\eta$).** *The admissible generalized unit elimination satisfies the described* $\beta\eta$ *equations.*

*Proof.* By induction on $\Phi/\Psi$ First, $\beta$

- If $\Phi = \Psi = \alpha$

$$\begin{aligned} (\mathrm{ind}_{\to}(\alpha.t, \alpha_-, x, \alpha_+))[\alpha, \mathrm{id}_\alpha, \alpha] &= \mathrm{ind}_{\to}(\alpha.t, \alpha, \mathrm{id}_\alpha) \\ &= t && (\mathrm{Unit}\beta) \end{aligned}$$

- If $\Phi = \alpha$ and $\Psi = \Psi, y, \beta$

$$\begin{aligned} ((\mathrm{ind}_{\to}^{\alpha_-;\Psi}(\alpha.\lambda^{\triangleright}(y, \beta).t; x)) \triangleright^{\beta} y)[\phi, \mathrm{id}_\alpha, \psi, s/y, b/\beta] &= ((\mathrm{ind}_{\to}^{\alpha_-;\Psi}(\alpha.\lambda^{\triangleright}(y, \beta).t; x)[[\phi, \mathrm{id}_\alpha, \psi]]) \triangleright^{b} s) \\ &\qquad (\text{Definition}) \\ &= (\lambda^{\triangleright}(y, \beta).t[\phi, \psi] \triangleright^{b} s) \\ &\qquad (\text{Induction}) \\ &= t[\phi, \psi, s/y, b/\beta] \\ &\qquad (\mathrm{Hom}\beta) \end{aligned}$$

- $\Phi = \beta, y, \Phi$ case is similar to previous.

Next $\eta$.

- $\Phi = \alpha_-$ and $\Psi = \alpha_+$:

  $\mathrm{ind}_{\to}^{\alpha_-;\alpha_+}(\alpha.s[\alpha/\alpha_-, \mathrm{id}_\alpha/x, \alpha/\alpha_+]; x) = \mathrm{ind}_{\to}(\alpha.s[\alpha/\alpha_-, \mathrm{id}_\alpha/x, \alpha/\alpha_+], \alpha_1, x, \alpha_2)$

  Which is equal to $s$ by the primitive unit $\eta$.
- $\Phi = \alpha_-$ and $\Psi = \Psi, y, \beta$:

$$\begin{aligned} s &= \lambda^{\triangleright}(y, \beta).s \triangleright^{\beta} y && \text{(Hom } \eta\text{)} \\ &= (\mathrm{ind}_{\to}^{\alpha_-;\Psi}(\alpha.\lambda^{\triangleright}(y, \beta).s[\mathrm{id}_\alpha/x]; x)) \triangleright^{\beta} y && \text{(Induction)} \\ &= \mathrm{ind}_{\to}^{\Phi;\Psi,y,\beta}(\alpha.s[\mathrm{id}_\alpha/x]; x) && \text{(Definition)} \end{aligned}$$

## B Details of Formal Category Theory Examples

Next, we provide some further details for some of the examples of the formal category theory constructions and theorems from Section 3.

**Definition 9 (Synthetic Composition/Functoriality).** *We provide the definitions of the terms in Construction 1*

1. *Identity* $id = \lambda\alpha.id_\alpha$
2. *Composition* $comp = \lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).ind_{\to}(\alpha.g \triangleright^{\forall\alpha_3} g, \alpha_1, f, \alpha_2)$
3. *Functoriality* $fctor(F) = \lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).ind_{\to}(\alpha.id_{F\,\alpha}, \alpha_1, f, \alpha_2)$
4. *Profunctoriality*

   $prof(R) = \lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).ind_{\to}(\alpha.\lambda^{\triangleright}(r, \beta_1).\lambda^{\triangleright}(g, \beta_2).r\ {}^{\alpha}\triangleleft(ind_{\to}(\beta.\lambda^{\triangleleft}(\alpha, r).r, \beta_1, g, \beta_2)), \alpha_1, f, \alpha_2)$

   *We can also define left and right composition for profunctors by applying the profunctorial action to an identity morphism on one side or the other:*

$$lcomp(R) = \lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).\lambda^{\triangleright}(r, \beta).prof(R)^{\alpha_1} \triangleright^{\alpha_2} f \triangleright^{\beta} r \triangleright^{\beta} id_\beta$$

$$rcomp(R) = \lambda\alpha.\lambda^{\triangleright}(r, \beta_1).\lambda^{\triangleright}(g, \beta_2).prof(R)^{\alpha} \triangleright^{\alpha} id_\alpha \triangleright^{\beta_1} r \triangleright^{\beta_2} g$$

*Associativity and unit follow by $\beta\eta$ for unit and homs.*

**Lemma 8 (Naturality).** *For any* $t : \forall\alpha : \mathbb{C}.R(\alpha; \alpha)$,

$\lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).lcomp(R)^{\alpha_1} \triangleright^{\alpha_2} f \triangleright^{\alpha_2} t^{\alpha_2} = \lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).rcomp(R)^{\alpha_1} \triangleright^{\alpha_1} t^{\alpha_1} \triangleright^{\alpha_2} f$

*Proof.* Expanding the definitions and applying $\beta$ reductions, both are equal to $\lambda\alpha_1.\lambda^{\triangleright}(f, \alpha_2).\mathrm{ind}_{\to}(\alpha.t^{\alpha}, \alpha_1, f, \alpha_2)$

**Lemma 9 (Yoneda, Co-Yoneda).** *Let* $\alpha^o : \mathbb{C}$ *and* $\pi : \mathcal{P}^+\mathbb{C}$. *Then*
*(Yoneda) The profunctor* $\alpha' \to_{\mathbb{C}} \alpha \triangleright^{\forall\alpha'} \alpha' \in \pi$ *is isomorphic to* $\alpha \in \pi$
*(Co-Yoneda) The profunctor* $\alpha \to \alpha' \overset{\exists\alpha'}{\odot} \alpha \in \pi$ *is isomorphic to* $\alpha \in \pi$

*Proof.* We show Yoneda in detail.

– The left-to-right homomorphism is defined as

$$M = \lambda\alpha.\lambda^{\triangleright}(\phi, \pi).\phi \triangleright^{\alpha} \mathrm{id}_{\alpha}$$

– The right-to-left homomorphism is defined as

$$N = \lambda\pi.\lambda^{\triangleright}(x, \alpha).\lambda^{\triangleright}(f, \alpha).'x \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi, x).x, \alpha', f, \alpha)$$

– First, right-to-left-to-right:

$$\begin{aligned}
\lambda\pi.\lambda^{\triangleright}(x,\alpha).M^{\alpha} \triangleright^{\pi} (N^{\pi} \triangleright^{\alpha} x) &= \lambda\pi.\lambda^{\triangleright}(x,\alpha).M^{\alpha} \triangleright^{\pi} (\lambda^{\triangleright}(f,\alpha').x \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha',f,\alpha)) \\
&= \lambda\pi.\lambda^{\triangleright}(x,\alpha).(\lambda^{\triangleright}(f,\alpha').x \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha',f,\alpha)) \triangleright^{\alpha} \mathrm{id}_{\alpha} \\
&= \lambda\pi.\lambda^{\triangleright}(x,\alpha).x \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha',\mathrm{id}_{\alpha},\alpha) \\
&\qquad\qquad (\mathrm{covhom}\beta) \\
&= \lambda\pi.\lambda^{\triangleright}(x,\alpha).x \,{}^{\pi}\!\triangleleft (\lambda^{\triangleleft}(\pi,x).x) \qquad (\mathrm{unit}\beta) \\
&= \lambda\pi.\lambda^{\triangleright}(x,\alpha).x \qquad (\mathrm{contrahom}\beta)
\end{aligned}$$

– Left-to-right-to-left

$$\begin{aligned}
&\lambda\alpha.\lambda^{\triangleright}(\phi,\pi).N^{\pi} \triangleright^{\alpha} (M^{\alpha} \triangleright^{\pi} \phi) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').(M^{\alpha} \triangleright^{\pi} \phi) \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha',f,\alpha) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').(\phi^{\alpha} \triangleright^{\alpha} \mathrm{id}_{\alpha}) \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha',f,\alpha) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').\mathrm{ind}_{\to}(\alpha.((\phi^{\alpha} \triangleright^{\alpha} \mathrm{id}_{\alpha}) \,{}^{\pi}\!\triangleleft \mathrm{ind}_{\to}(\alpha.\lambda^{\triangleleft}(\pi,x).x,\alpha,\mathrm{id}_{\alpha},\alpha)),\alpha',f,\alpha) \\
&\qquad\qquad (\mathrm{unit}\ \eta) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').\mathrm{ind}_{\to}(\alpha.((\phi^{\alpha} \triangleright^{\alpha} \mathrm{id}_{\alpha}) \,{}^{\pi}\!\triangleleft \lambda^{\triangleleft}(\pi,x).x),\alpha',f,\alpha) \\
&\qquad\qquad (\mathrm{unit}\ \beta) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').\mathrm{ind}_{\to}(\alpha.((\phi^{\alpha} \triangleright^{\alpha} \mathrm{id}_{\alpha})),\alpha',f,\alpha) \qquad (\mathrm{contrahom}\ \beta) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\lambda^{\triangleright}(f,\alpha').(\phi^{\alpha'} \triangleright^{\alpha} f) \qquad (\mathrm{unit}\ \eta) \\
&= \lambda\alpha.\lambda^{\triangleright}(\phi,\pi).\phi \qquad (\mathrm{covhom}\ \eta)
\end{aligned}$$

**Lemma 10 (Fubini).** *We show two of the Fubini cases in detail:*

1. $S(\gamma;\delta) \,{}^{\forall\gamma}\!\triangleleft (P(\gamma;\beta) \overset{\exists\beta}{\odot} Q(\beta;\alpha)) \cong S(\gamma;\delta) \,{}^{\forall\gamma}\!\triangleleft P(\gamma;\beta) \,{}^{\forall\beta}\!\triangleleft Q(\beta;\alpha)$
2. $\forall\alpha.P(\alpha;\beta) \triangleright^{\forall\beta} Q(\alpha;\beta) \cong \forall\beta.Q(\alpha;\beta) \,{}^{\forall\alpha}\!\triangleleft P(\alpha;\beta)$

*Proof.* 1. This is a form of Currying isomorphism, as the $\lambda$ term makes clear:

– Left to Right

$$\lambda\alpha.\lambda^{\triangleright}(h,\delta).\lambda^{\triangleleft}(\beta,q).\lambda^{\triangleleft}(\gamma,p).(p,\beta,q) \,{}^{\gamma}\!\triangleleft h$$

– Right to Left

$$\lambda\alpha.\lambda^{\triangleright}(k,\delta).\lambda^{\triangleleft}(\gamma,w).\mathrm{ind}_{\odot}(p,\beta,q.q\,{}^{\beta}\!\triangleleft\, p\,{}^{\gamma}\!\triangleleft\, k;w)$$

– Left to Right to Left

$$\begin{aligned}
&\lambda\alpha.\lambda^{\triangleright}(h,\delta).\lambda^{\triangleleft}(\gamma,w).\mathrm{ind}_{\odot}(p,\beta,q.(p\,{}^{\gamma}\!\triangleleft\, q\,{}^{\beta}\!\triangleleft(\lambda^{\triangleleft}(\beta,q).\lambda^{\triangleleft}(\gamma,p).(p,\beta,q)\,{}^{\gamma}\!\triangleleft\, h));w)\\
&= \lambda\alpha.\lambda^{\triangleright}(h,\delta).\lambda^{\triangleleft}(\gamma,w).\mathrm{ind}_{\odot}(p,\beta,q.(p\,{}^{\gamma}\!\triangleleft(\lambda^{\triangleleft}(\gamma,p).(p,\beta,q)\,{}^{\gamma}\!\triangleleft\, h));w) && (\mathrm{contrahom}\beta)\\
&= \lambda\alpha.\lambda^{\triangleright}(h,\delta).\lambda^{\triangleleft}(\gamma,w).\mathrm{ind}_{\odot}(p,\beta,q.((p,\beta,q)\,{}^{\gamma}\!\triangleleft\, h);w) && (\mathrm{contrahom}\beta)\\
&= \lambda\alpha.\lambda^{\triangleright}(h,\delta).\lambda^{\triangleleft}(\gamma,w).w\,{}^{\gamma}\!\triangleleft\, h && (\mathrm{tensor}\eta)\\
&= \lambda\alpha.\lambda^{\triangleright}(h,h). && (\mathrm{contrahom}\eta)
\end{aligned}$$

– Right to Left to Right

$$\begin{aligned}
&\lambda\alpha.\lambda^{\triangleright}(k,\delta).\lambda^{\triangleleft}(\beta,q).\lambda^{\triangleleft}(\gamma,p).(p,\beta,q)\,{}^{\gamma}\!\triangleleft(\lambda^{\triangleleft}(\gamma,w).\mathrm{ind}_{\odot}(p,\beta,q.p\,{}^{\gamma}\!\triangleleft\, q\,{}^{\beta}\!\triangleleft\, k;w))\\
&= \lambda\alpha.\lambda^{\triangleright}(k,\delta).\lambda^{\triangleleft}(\beta,q).\lambda^{\triangleleft}(\gamma,p).\mathrm{ind}_{\odot}(p,\beta,q.p\,{}^{\gamma}\!\triangleleft\, q\,{}^{\beta}\!\triangleleft\, k;(p,\beta,q))\\
&= \lambda\alpha.\lambda^{\triangleright}(k,\delta).\lambda^{\triangleleft}(\beta,q).\lambda^{\triangleleft}(\gamma,p).p\,{}^{\gamma}\!\triangleleft\, q\,{}^{\beta}\!\triangleleft\, k\\
&= \lambda\alpha.\lambda^{\triangleright}(k,\delta).\lambda^{\triangleleft}(\beta,q).q\,{}^{\beta}\!\triangleleft\, k\\
&= \lambda\alpha.\lambda^{\triangleright}(k,\delta).k
\end{aligned}$$

2. This isomorphism relates left and right homs. Unlike the previous cases, the isomorphism is of types, not sets/profunctors.
   – Left to right
   $$\lambda X.\lambda\beta.\lambda^{\triangleleft}(\alpha,p).X^{\alpha}\,{}^{\beta}\!\triangleright\, p$$
   – Right to left
   $$\lambda Y.\lambda\alpha.\lambda^{\triangleright}(p,\beta).p\,{}^{\alpha}\!\triangleleft\, Y^{\beta}$$
   – Left to right to left

$$\begin{aligned}
&\lambda X.\lambda\alpha.\lambda^{\triangleright}(p,\beta).p\,{}^{\alpha}\!\triangleleft(\lambda\beta.\lambda^{\triangleleft}(\alpha,p).X^{\alpha}\,{}^{\beta}\!\triangleright\, p)^{\beta}\\
&= \lambda X.\lambda\alpha.\lambda^{\triangleright}(p,\beta).p\,{}^{\alpha}\!\triangleleft(\lambda^{\triangleleft}(\alpha,p).X^{\alpha}\,{}^{\beta}\!\triangleright\, p) && (\mathrm{nat.elt}.\beta)\\
&= \lambda X.\lambda\alpha.\lambda^{\triangleright}(p,\beta).X^{\alpha}\,{}^{\beta}\!\triangleright\, p && (\mathrm{contrahom}\beta)\\
&= \lambda X.\lambda\alpha.X^{\alpha} && (\mathrm{contrahom}\eta)\\
&= \lambda X.X && (\mathrm{nat.elt}.\eta)
\end{aligned}$$

   – The other case is similar.

**Lemma 11 (Equivalent Definitions of Adjoints).** *We show that given a morphism of profunctors*

$$\forall\alpha.L\alpha \to \beta\,{}^{\forall\beta}\!\triangleright\, \alpha \to R\beta$$

*we can extract a unit natural transformation* $\eta : \forall\alpha.\alpha \to R(L\alpha)$ *and vice-versa.*

*Proof.* The construction is exactly the ordinary proof but formalized in VETT syntax. Given the morphism of profunctors $M$, we define the unit by evaluating at the identity:

$$\forall \alpha. M^{\alpha} \triangleright^{\alpha} \mathrm{id}_{\alpha}$$

and given the unit $\eta$, we can define a morphism of profunctors by composing the unit with the functorial lift of the input:

$$\forall \alpha. f \triangleright^{\forall \beta} \mathrm{comp}(\eta^{\alpha}, \mathrm{fctor}(R)(f))$$

That this is an isomorphism follows by a similar argument to the proof of the Yoneda lemma.

## C Details of Semantics

In this section, we provide the full descriptions of the universal properties in a virtual equipment corresponding to each connective in VETT.

**Definition 10 (Universal Properties for Category Connectives).** *Let $\mathcal{V}$ be a virtual equipment.*

1. *Let $C$ be a small object, then a contravariant presheaf object $\mathcal{P}^{-}C$ is an object with natural isomorphism $\mathcal{V}_o(A, \mathcal{P}^{-}C) \cong \{R \in V_h \mid s(R) = C \wedge t(R) = A\}$*
2. *Let $C$ be a small object, then a covariant presheaf object $\mathcal{P}^{+}C$ is an object with natural isomorphism $\mathcal{V}_o(A, \mathcal{P}^{+}C) \cong \{R \in V_h \mid t(R) = C \wedge s(R) = A\}$*
3. *Let $R$ be a horizontal arrow, then a tabulator $\int R$ is an object with natural isomorphism $\mathcal{V}_o(A, \int R) \cong \sum_{f:\mathcal{V}_o(A,s(R))} \sum_{g:\mathcal{V}_o(A,t(R))} \mathcal{V}_2(\cdot; f; g; R)$*
4. *A nullary product is an object $1$ with natural isomorphism $\mathcal{V}_o(A, 1) \cong 1$*
5. *A binary product of $B$ and $C$ is an object $B \times C$ with natural isomorphism $\mathcal{V}_o(A, B \times C) \cong \mathcal{V}_o(A, B) \times \mathcal{V}_o(A, C)$*

**Definition 11 (Universal Properties for Set Connectives).** *Let $\mathcal{V}$ be a virtual equipment.*

1. *A unit $U_C$ for an object $C$ is a horizontal arrow $U_C$ with $s(U_C) = t(U_c) = C$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{P}, U_c, \overrightarrow{Q}; f; g; R) \cong \mathcal{V}_2(\overrightarrow{P}, \overrightarrow{Q}; f; g; R)$*
2. *A tensor of horizontal arrows $P$ and $Q$ where $t(P) = s(Q)$ is a horizontal arrow $P \odot Q$ with $s(P \odot Q) = s(P)$ and $t(P \odot Q) = t(Q)$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{R}, P \odot Q, \overrightarrow{S}; f; g; T) \cong \mathcal{V}_2(\overrightarrow{R}, P, Q, \overrightarrow{S}; f; g; T)$.*
3. *A covariant hom of $P$ and $Q$ where $t(P) = t(Q)$ is a horizontal arrow $P \triangleright Q$ with $s(P \triangleright Q) = s(Q)$ and $t(P \triangleright Q) = s(P)$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{R}; f; id; P \triangleright Q) \cong \mathcal{V}_2(\overrightarrow{R}, P; f; id; Q)$*
4. *A contravariant hom of $P$ and $Q$ where $s(P) = s(Q)$ is a horizontal arrow $P \triangleleft Q$ with $s(P \triangleleft Q) = t(Q)$ and $t(P \triangleleft Q) = t(P)$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{R}; id; g; P \triangleleft Q) \cong \mathcal{V}_2(Q, \overrightarrow{R}; id; g; P)$*
5. *A nullary product for an object $C$ is a horizontal arrow $1_C$ with $s(1_C) = t(1_C) = C$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{P}; f; g; 1_C) \cong 1$*

6. *A binary product of horizontal arrows $P$ and $Q$ where $s(P) = s(Q)$ and $t(P) = t(Q)$ is a horizontal arrow $P \times Q$ with $s(P \times Q) = s(P)$ and $t(P \times Q) = t(P)$ with natural isomorphism $\mathcal{V}_2(\overrightarrow{R}; f; g; P \times Q) \cong \mathcal{V}_2(\overrightarrow{R}; f; g; P) \times \mathcal{V}_2(\overrightarrow{R}; f; g; Q)$*

*We require that in our models, units exist for all objects, tensors and homs overs small objects exist and all finite products exist. We additionally require that the choice of tensors, homs and products commute strictly with restrictions in that*

1. $(P \odot Q)(f, g) = (P(f, id) \odot Q(id, g))$
2. $(P \triangleright Q)(f, g) = (P(g, id) \triangleright Q(f, id))$
3. $(P \triangleleft Q)(f, g) = (P(id, g) \triangleleft Q(id, f))$
4. $1(f, g) = 1$
5. $(P \times Q)(f, g) = (P(f, g) \times Q(f, g))$

*Note that these equations necessarily hold up to isomorphism, even if we do not require them to commute strictly.*

### C.1 Completeness

Next we describe the syntactic properties of substitution that are needed in order to prove the *completeness* theorem, that is, that the syntax of VETT presents a hyperdoctrine of virtual equipments.

**Definition 12 (Syntactic Virtual Equipment).** *Fix a context $\Gamma$. Define a virtual equipment $Syn^{\Gamma}$ as follows:*

1. *The vertical category $Syn_o^{\Gamma}$ has categories $\Gamma \vdash \mathbb{C}$ Cat as objects, small categories as small objects and as arrows from $\mathbb{C}$ to $\mathbb{D}$ objects $\alpha : \mathbb{C} \vdash b : \mathbb{D}$ modulo renaming of the input variable. Composition is given by substitution and identity is the variable.*
2. *The horizontal arrows are the sets $\alpha : \mathbb{C}; \beta : \mathbb{D} \vdash R$ (up to renaming $\alpha$ and $\beta$) with source $\mathbb{C}$ and target $\mathbb{D}$.*
3. *Note that composable strings $\overrightarrow{R}$ of horizontal arrows are in bijection with contexts $\Phi$. Then we can define a 2-cell $Syn_2^{\Gamma}(\Phi, a, b, S)$ to be an element $\Gamma \mid \Phi \vdash s : S[a/\alpha; b/\beta]$.*
   *Composition is defined by substitution $t[\phi]$ as substitutions are in bijection with the "sequences of 2-cells" used in the definition of a virtual equipment. Associativity says that $t[\phi][\psi] = t[\phi[\psi]]$ where the composition $\phi[\psi]$ is defined below and corresponds exactly to the associativity rule in a virtual equipment. The unit is the variable, and they are unital as $x[s/x] = s$ and $s[\overrightarrow{x/x}] = s$.*
4. *Restriction along vertical arrows is given by substitution $R(a, b) = R[a/\alpha; b/\beta]$. This is strictly associative and unital, and the cartesian cell from $R$ to $R[a/\alpha; b/\beta]$ is just the identity $x : R[a/\alpha; b/\beta] \vdash x : R[a/\alpha; b/\beta]$.*

**Definition 13.** *We define the vertical composition of transformation substitutions $\phi[\psi]$ inductively on $\phi$.*

$$(a/\alpha)[b/\beta] = a[b/\beta]/\alpha$$
$$(\phi_1, t/x, a/\alpha)[\psi_1 \curlyvee \psi_2] = \phi_1[\psi_1], t[\psi_2], a[d^+\psi_2]$$

*This covers all cases by lemma 6.*

*We define the vertical identity $id_\Phi$ by induction on $\Phi$*

$$id_{\alpha:\mathbb{C}} = \alpha/\alpha$$
$$id_{\Phi,x:R,\alpha:\mathbb{C}} = id_\Phi, x/x, \alpha/\alpha$$

*By induction this is seen to be associative:*

$$\phi[\psi][\sigma] = \phi[\psi[\sigma]]$$

*and unital*

$$id_\Phi[\phi] = \phi = \phi[id_\Psi]$$