TYPE THEORIES IN CATEGORY THEORY

TESLA ZHANG

July 26, 2021

ABSTRACT. We introduce basic notions in category theory to type theorists, including comprehension categories, categories with attributes, contextual categories, type categories, and categories with families along with additional discussions that are not very closely related to type theories by listing definitions, lemmata, and remarks. By doing so, this introduction becomes more friendly as a referential material to be read in random order (instead of from the beginning to the end). In the end, we list some mistakes made in the early versions of this introduction.

The interpretation of common type formers in dependent type theories are discussed based on existing categorical constructions instead of mechanically derived from their type theoretical definition. Non-dependent type formers include unit, products (as fiber products), and functions (as *fiber exponents*), and dependent ones include extensional equalities (as equalizers), dependent products, and the universe of (all) propositions (as the subobject classifier).

CONTENTS

1. Preface

Warning 1.1 (Prerequisites)**.** We assume the familiarity of the following concepts:

(1) Set theory: sets, subsets, elements, functions, image, fiber, injectivity, and cartesian products.
(2) Type theory: dependent types, lambda calculus, (telescopic) contexts, typing judgments, and substitutions.
(3) Category theory: categories, objects, morphisms, isomorphisms, commutative diagrams, functors, and natural transformations.

Typing judgments and substitutions are slightly discussed (in §2), but not in an introductory way. ☠

Notation 1.2. We introduce some notational conventions in category theory.

- We write $A \in \mathcal{C}$ to say that "A is an object in the category $\mathcal{C}$", and $f \in \mathcal{C}(A, B)$ to say that "f is a morphism in the category $\mathcal{C}$ from A towards B". In the literature, the former is also written as $A \in \mathrm{Ob}(\mathcal{C})$ and the latter is also written as $f \in \mathrm{Hom}_{\mathcal{C}}(A, B)$.
- We write $\mathrm{Ob}(\mathcal{C})$ for the set of objects in $\mathcal{C}$.
- We write $\mathrm{id}_A \in \mathcal{C}(A, A)$ for the identity morphism, $dom(f)$ and $cod(f)$ for the domain projection and the codomain projection of morphisms. We also say a morphism f is a morphism *from* $dom(f)$ and *towards* $cod(f)$.
- We write $f \circ g \in \mathcal{C}(dom(g), cod(f))$ for composition of morphisms.

▲

Terminology 1.3 (NatIncl)**.** For two sets $B \subseteq A$, the "identity" function $\iota : B \to A$ is called a *natural inclusion*. ▲

Convention 1.4 (Sterling [Ste20])**.** In the typing rules, some judgments might be *entailed* by another. We write the entailed judgment(s) out explicitly in gray if they worth it. ▲

Convention 1.5 (Cockx [CA18])**.** We use boxes to distinguish type theory or category theory notations from natural language text, e.g. "we combine a term $\boxed{a}$ with a term $\boxed{b}$ to get a term $\boxed{a\ b}$". ▲

2. Background and motivation

Remark 2.1. The notion of *substitution* in type theory is the process of replacing a variable with a term. We sometimes refer to the action of replacement with the verb *apply*, like "apply this substitution to that term". We generalize this notion to lists, so that one substitution deals with multiple replacements. ✎

Definition 2.2 (SubstObj)**.** The data carried by a substitution is called a *substitution object* – a list of mapping from variables to terms. Substitution objects can be *applied* on terms. ⛰

Demonstration 2.3. Consider a lambda calculus term $(\lambda x.\lambda y.u)\ a\ b$, we denote its reduced form as $u[a/x, b/y]$ (a notation similar to the one used by Curry [CFC59]). The $[a/x, b/y]$ part is an example of a substitution object (definition 2.2). ⛰

Remark 2.4 (Nameless)**.** Contexts, types, terms, and substitution objects are considered up to α-renaming throughout this introduction. That is to say, there are no "names" in these structures – α-equivalent terms are considered the same. We can imagine that we are using a nameless representation of binding structures, such as de Bruijn indices [de 72]. ✎

Notation 2.5. We introduce some notational conventions in type theory.

We use Γ, Δ to denote *contexts*, $\boxed{\sigma, \gamma}$ (recall convention 1.5) for substitution objects (definition 2.2), lowercase letters for terms (preferably u, v), and uppercase letters for types (preferably A, B).

We write $u\sigma$ for "applying the substitution object σ to term u". ⛰

Remark 2.6. In a simple type theory, terms are typed in a context and types are canonical mathematical objects. However, since we are dealing with a dependent type theory, types will be formed in a context too.

We will have two basic judgments: *context formation* $\Gamma \vdash$ (the types in Γ are well-typed) and *type formation* $\boxed{\Gamma \vdash A \text{ type}}$ (A is a well-typed type in Γ). Based on these judgments, we define the judgment for terms (using convention 1.4):

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type}}{\Gamma \vdash u : A} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A}{\Gamma \vdash u = v : A}$$

It reads "term u has type A in context Γ, presupposing $\boxed{\Gamma \vdash A \text{ type}}$". We also have the judgment for substitution objects:

$$\frac{\Gamma \vdash \qquad \Delta \vdash}{\Gamma \vdash \sigma : \Delta}$$

It reads "in the context Γ, the i-th term in σ have the i-th type in Δ, presupposing $\Gamma \vdash, \Delta \vdash$". It is assumed that σ and Δ have the same length. ✎

Remark 2.7 (Substitutions)**.** Although we are going to treat substitution informally, we would like to emphasize some important insights about substitutions.

- Consider a substitution object $\Gamma \vdash \sigma : \Delta$ (a judgment already described in remark 2.6): it is a fact that "terms in σ can refer to bindings in Γ", and we say "σ *instantiates* the context Δ".
- Consider a term $\Delta \vdash u : A$, both u and A can refer to bindings in Δ.

We can turn u into another term, typed in the context Γ by applying the substitution σ since the open references to Δ can be replaced by the terms in σ, which can refer to bindings in Γ. This process can be described by the following deduction:

$$\frac{\dfrac{\Gamma \vdash \qquad \Delta \vdash}{\Gamma \vdash \sigma : \Delta} \qquad \dfrac{\Delta \vdash A \text{ type}}{\Delta \vdash u : A}}{\Gamma \vdash A\sigma \text{ type} \qquad \text{and} \qquad \Gamma \vdash u\sigma : A\sigma}$$

Similarly, for contexts $\Gamma, \Gamma', \Gamma''$ and substitutions objects σ, γ, there is:

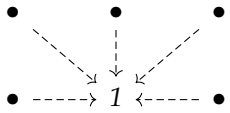
$$\dfrac{\dfrac{\Gamma \vdash \quad \Gamma' \vdash}{\Gamma \vdash \sigma : \Gamma'} \quad \dfrac{\Gamma' \vdash \quad \Gamma'' \vdash}{\Gamma' \vdash \gamma : \Gamma''}}{\Gamma \vdash \gamma\sigma : \Gamma''}$$

The above deduction tree induces the definition of a category (definition 2.25). ✎

2.1. **Pullbacks and products.**

Notation 2.8. We refer to the category of small sets[1] as **Set**. ▲

Definition 2.9 (TermObj)**.** For a category $\mathcal{C}$, an object $1 \in \mathcal{C}$ is called *terminal* if for every $A \in \mathcal{C}$ there is a unique morphism $1_A \in \mathcal{C}(A, 1)$. The visualization is trivial:

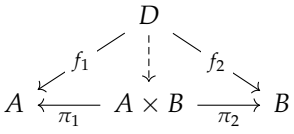


▲

Lemma 2.10 (Uniqueness)**.** *If a category $\mathcal{C}$ has several terminal objects, they are (uniquely) isomorphic to each other.*

Proof. By definition 2.9, consider two terminal objects $X, Y \in \mathcal{C}$, there are unique morphisms $X_Y \in \mathcal{C}(Y, X)$ and $Y_X \in \mathcal{C}(X, Y)$ whose compositions cannot be any morphism other than the identity morphism. Thus an isomorphism. □

Definition 2.11 (Product)**.** For a category $\mathcal{C}$ and $A, B \in \mathcal{C}$, the *product object* of A and B, denoted $(A \times B) \in \mathcal{C}$, is an object in $\mathcal{C}$ equipped with two morphisms $\pi_1 \in \mathcal{C}(A \times B, A)$ and $\pi_2 \in \mathcal{C}(A \times B, B)$ such that for every object $D \in \mathcal{C}$ with two morphisms $f_1 \in \mathcal{C}(D, A)$ and $f_2 \in \mathcal{C}(D, B)$, there is a unique morphism in $\mathcal{C}(D, A \times B)$ (called the *product morphism* of f_1 and f_2) that commutes the following diagram:



☆

Lemma 2.12 (Uniqueness)**.** *In a category $\mathcal{C}$ and $A, B \in \mathcal{C}$, the product object is unique up to unique isomorphism.*

Proof. By the definition of product objects, they have unique morphisms towards each other, and these two morphisms are inverse to each other by their uniqueness. □

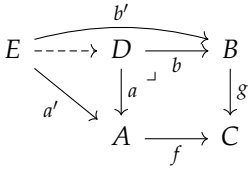
Exercise 2.13. Prove that in **Set** the cartesian product of sets $A, B \in$ **Set** is the product object.

Exercise 2.14. For readers who are familiar with the notion of *groups* and *direct product of groups*, prove that in the category of groups and group homomorphisms the direct product of groups is the product object.

[1] If you do not know what are *small* sets, think of them as sets.

Exercise 2.15. For readers who are familiar with the notion of *topological spaces* and *product space*, prove that in the category of topological spaces and continuous functions the product space of topological spaces is the product object.

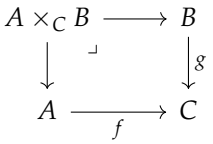
Definition 2.16 (Pullback)**.** For a category $\mathcal{C}$, objects $A, B, C \in \mathcal{C}$, and morphisms $f \in \mathcal{C}(A, C)$, $g \in \mathcal{C}(B, C)$, the *pullback* of f and g is an object $D \in \mathcal{C}$ equipped with two morphisms $a \in \mathcal{C}(D, A)$ and $b \in \mathcal{C}(D, B)$ where $f \circ a = g \circ b \in \mathcal{C}(D, C)$, such that every object $E \in \mathcal{C}$ with $a' \in \mathcal{C}(E, A)$ and $b' \in \mathcal{C}(E, B)$ where $f \circ a' = g \circ b' \in \mathcal{C}(E, C)$, there is a unique morphism in $\mathcal{C}(E, D)$ commuting the following diagram:



If in a category $\mathcal{C}$, for all objects $A, B, C \in \mathcal{C}$ and pairs of morphisms $f \in \mathcal{C}(A, C)$, $g \in \mathcal{C}(B, C)$ the pullback of f and g exists, we say that $\mathcal{C}$ *has all pullbacks*. ☆

Remark 2.17. Note how similar the diagrams in definitions 2.11 and 2.16 are. We sometimes denote the pullback D in definition 2.16 as $A \times_C B$ and refer to D as the *fiber product* of A and B. ✎

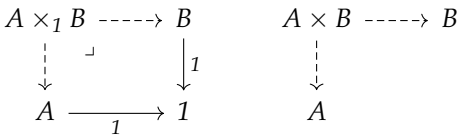
Notation 2.18. In definition 2.16, the objects $A, B, C, D \in \mathcal{C}$ constitute a commutative square. If a commutative square characterizes a pullback, we mark it with a ⌟ and refer to the square as a *pullback square*:



▲

Lemma 2.19. *In a category $\mathcal{C}$ with all pullbacks (definition 2.16) and a terminal object (definition 2.9) $1 \in \mathcal{C}$, for every $A, B \in \mathcal{C}$ we have $A \times_1 B = A \times B$.*

Proof. We prove by constructing both commutative squares:

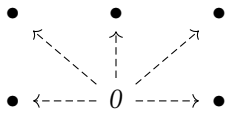


The fact that the right-bottom vertex of the pullback square is 1 makes the square always commutative, so being a pullback means being terminal in all such squares, which is a product object. The other side of the implication holds in a similar way. □

2.2. **Duality and coproducts.**

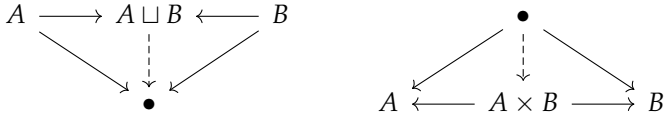
Definition 2.20 (Opposite)**.** For a category $\mathcal{C}$, the *opposite category* $\mathcal{C}^{\mathrm{op}}$ of $\mathcal{C}$ is $\mathcal{C}$ but the domain and codomain of every morphism are swapped. ▲

Definition 2.21 (InitObj)**.** For a category $\mathcal{C}$, an object $0 \in \mathcal{C}$ is called *initial* if it is the terminal object (definition 2.9) in $\mathcal{C}^{\mathrm{op}}$ (definition 2.20), visualized below:



Demonstration 2.22 (Parker)**.** The empty set $\emptyset$ is the initial object of **Set** since the maps from $\emptyset$ are all trivial.

Definition 2.23 (Coproduct)**.** For a category $\mathcal{C}$ and $A, B \in \mathcal{C}$ we say take the product object $(A \times B) \in \mathcal{C}^{\mathrm{op}}$ to be the *coproduct object* $(A \sqcup B) \in \mathcal{C}$, visualized along with the product object $(A \times B) \in \mathcal{C}$ below:



Note that flipping the arrows in one of the diagrams results in the other.

Convention 2.24 (Duality)**.** If one construction in a category $\mathcal{C}$ is another construction in the category $\mathcal{C}^{\mathrm{op}}$, we say that these two constructions are *dual* to each other. For example, product objects (definition 2.11) are dual to coproduct objects (definition 2.23), terminal objects (definition 2.9) are dual to initial objects (definition 2.21).

2.3. **The category of contexts.**

Definition 2.25 (ContextCat)**.** A *category of contexts* $\mathcal{C}$ is a category whose objects are (telescopic) contexts and morphisms are substitutions objects. For instance, if there are objects $\Gamma, \Delta \in \mathcal{C}$ that correspond to some contexts in type theory, the morphism $\sigma \in \mathcal{C}(\Gamma, \Delta)$ corresponds to a substitution object that is typed in Γ and instantiates (remark 2.7) Δ.

For an object $\Gamma \in \mathcal{C}$, we define id_Γ to be the identity substitution and $\sigma \circ \gamma$ as "applying the substitution γ on the terms in σ".

The category of contexts of a type theory T is also known as the *syntactical category* [Sco, definition 2] of T.

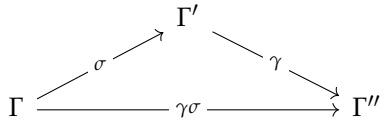
Definition 2.26 (ClCat)**.** The *classifying category* $Cl(T)$ of a type theory T is the *most general* interpretation of T. The name is due to Pitts [Pit01, §4.2]. In most cases, not only all constructions in T are interpreted in $Cl(T)$, but also all the constructions in $Cl(T)$ are interpreted in T. This property is also known as *completeness*.

Demonstration 2.27. The syntactical category (definition 2.25) of a type theory is a classifying category (definition 2.26).

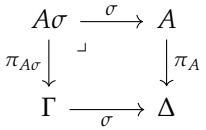
Notation 2.28. We write $[\![\Gamma]\!]$ for the semantical interpretation of a type theoretical construct Γ in a category. This notation will be overloaded for contexts and substitution objects. Later, in §4, the same notation will be used on types and terms, too.

Demonstration 2.29. Recall the last deduction in remark 2.7. It corresponds to the composition of substitution objects in the category of contexts $\mathcal{C}$: for $[\![\gamma]\!] \in$

$\mathcal{C}(\llbracket\Gamma'\rrbracket, \llbracket\Gamma''\rrbracket)$ and $\llbracket\sigma\rrbracket \in \mathcal{C}(\llbracket\Gamma\rrbracket, \llbracket\Gamma'\rrbracket)$, there is $\llbracket\gamma\sigma\rrbracket = \llbracket\gamma\rrbracket \circ \llbracket\sigma\rrbracket \in \mathcal{C}(\llbracket\Gamma\rrbracket, \llbracket\Gamma''\rrbracket)$. Visualization omitting the brackets:



Remark 2.30. Recall the second last deduction in remark 2.7. Consider an operation π_A for a dependent type A (we did not say that A belong to any category, but we can still talk about functions acting on types) that gives the context where A is formed within, the following diagram commutes in a syntactical category (all of the vertices are from the second last judgment in remark 2.7):



Exercise 2.31. Prove the universal property of the pullback (definition 2.16) square in remark 2.30. A referential proof is in [Sco, lemma 6].

Remark 2.32 (STLC)**.** The category of contexts (definition 2.25) already provides semantics for simple type theories. Types are just contexts of length one, while terms are just substitution objects of length one. This is because in a simple type theory, types are formed *without* a context:

$$\frac{A \text{ type} \qquad B \text{ type}}{(A \times B) \text{ type}} \qquad \frac{A \text{ type} \qquad B \text{ type}}{(A \to B) \text{ type}} \qquad \frac{A \text{ type}}{(\text{List } A) \text{ type}} \qquad \frac{A \text{ type}}{(\text{Maybe } A) \text{ type}}$$

$$\text{Int type} \qquad \text{Bool type} \qquad \top \text{ type} \qquad \bot \text{ type}$$

The formation of types is interpreted as objects in a category of contexts, and the typing judgments $\Gamma \vdash u : A$, $\Gamma \vdash \sigma : \Delta$ are interpreted as morphisms in a category of contexts.

Remark 2.33. It is possible to describe a *dependent type* in a category of contexts $\mathcal{C}$ by interpreting the judgment $\boxed{\Gamma \vdash A \text{ type}}$ as $\boxed{\Gamma, A \vdash}$ so that it corresponds to an object in $\mathcal{C}$. However, by constructing a categorical model directly from the classifying category, we will learn nothing about the relationship between category theory and type theory.

To find a better interpretation, we construct a category theoretical construction directly in category theory and interpret type theoretical constructions with it.

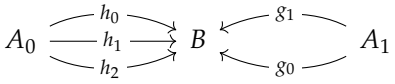
3. Basic category theory

Definition 3.1 (Presheaf)**.** For a category $\mathcal{C}$, functor $T : \mathcal{C}^{\text{op}} \to \mathbf{Set}$ is called a *presheaf*. Sometimes we also say that functor $T' : \mathcal{C}^{\text{op}} \to \mathcal{V}$ to be a $\mathcal{V}$-valued presheaf.

Convention 3.2 (Presheaves)**.** Not all categories $\mathcal{V}$ are suitable as the codomain of presheaves (definition 3.1). We normally use those whose objects are some kind of

"containers" (like sets, categories, etc.) so that for each object and morphism in $\mathcal{C}$, we obtain information about them from the presheaves over $\mathcal{C}$. ▲

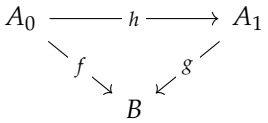
Definition 3.3 (HomFunctor)**.** For a category $\mathcal{C}$ and an object $B \in \mathcal{C}$, the *hom functor* $\mathcal{C}(-, B) : \mathcal{C}^{\text{op}} \to \mathbf{Set}$ sends each object $A \in \mathcal{C}$ to the hom set $\mathcal{C}(A, B)$. The morphism $h \in \mathcal{C}(X, Y)$ is sent to the function who sends $g \in \mathcal{C}(Y, B)$ to $g \circ h \in \mathcal{C}(X, B)$. A poor visualization:



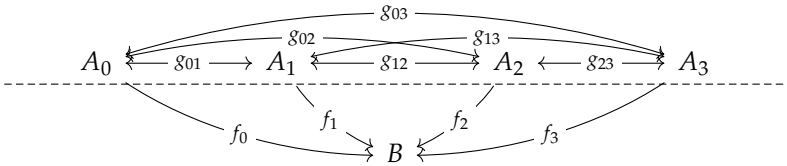
The hom functor sends A_0 to the set $\{h_0, h_1, h_2\}$ and A_1 to the set $\{g_0, g_1\}$. Similarly, we also say $\mathcal{C}(B, -) : \mathcal{C} \to \mathbf{Set}$ to be a hom functor. ▲

Demonstration 3.4. Hom functors (definition 3.3) are presheaves (definition 3.1). For a category $\mathcal{C}$, a presheaf $T : \mathcal{C}^{\text{op}} \to \mathbf{Set}$ is called a *representable presheaf* if it is naturally isomorphic to $\mathcal{C}(-, B)$. ▲

Definition 3.5 (Overcategory)**.** For a category $\mathcal{C}$ and $B \in \mathcal{C}$, we define a category $\mathcal{C}_{/B}$, called the *category over* B, whose objects are morphisms in $\mathcal{C}$ towards B. Morphisms in $\mathcal{C}_{/B}$ are inherited from $\mathcal{C}$ in the following way: given morphisms $f \in \mathcal{C}(A_0, B)$ and $g \in \mathcal{C}(A_1, B)$, a morphism in $\mathcal{C}_{/B}$ between objects f and g is a morphism $h \in \mathcal{C}(A_0, A_1)$ (note proposition 3.6) such that $g \circ h = f$. This process is visualized below:



Identities and compositions of $\mathcal{C}_{/B}$ are also inherited from $\mathcal{C}$. We visualize $\mathcal{C}_{/B}$ with the following diagram with $A_0, A_1, A_2, A_3 \in \mathcal{C}$:



The morphisms below the dashed line (f_i for $i \in [0,3]$) are objects of $\mathcal{C}_{/B}$, while the others (g_{ij} for $i, j \in [0,3]$) are morphisms of $\mathcal{C}_{/B}$. We use bidirectional arrows to mean "morphisms of both directions". $\mathcal{C}_{/B}$ is also known as an *overcategory* or the *slice category over* B. ▲

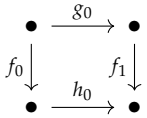
Proposition 3.6.
$$g \in \mathcal{C}_{/B}(f, f') \implies g \in \mathcal{C}(dom(f), dom(f'))$$
$$g \in \mathcal{C}(dom(f), dom(f')) \not\Longrightarrow g \in \mathcal{C}_{/B}(f, f')$$
▲

Lemma 3.7. *For a category $\mathcal{C}$ and its terminal object (definition 2.9) 1, $\mathcal{C} \simeq \mathcal{C}_{/1}$.*

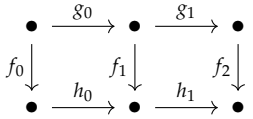
Proof. We take the objects in $\mathcal{C}$ to its unique morphism towards the terminal object in the forward direction, and take the domain of these morphisms in the backward direction. □

Demonstration 3.8. For $\mathcal{C}$ a category of contexts (definition 2.25) and $\Gamma \in \mathcal{C}$, the category of "substitutions typed in Γ" is the overcategory (definition 3.5) $\mathcal{C}_{/\Gamma}$. ▲

Definition 3.9 (ArrowCat)**.** For an arbitrary category $\mathcal{C}$, the *arrow category* $\mathcal{C}^{\to}$ (or $\mathrm{Arr}(\mathcal{C})$) is the category whose objects are morphisms in $\mathcal{C}$. The morphisms in $\mathcal{C}^{\to}$ are commutative diagrams in $\mathcal{C}$. For example, for $f_0, f_1 \in \mathcal{C}^{\to}$, the hom set $\mathcal{C}^{\to}(f_0, f_1)$ are pairs of morphisms (g_0, h_0) in $\mathcal{C}$ such that the following square commutes:



Composition in $\mathcal{C}$ is the composition of commutative diagrams. For example, the composition of $(g_0, h_0) \in \mathcal{C}^{\to}(f_0, f_1)$ and $(g_1, h_1) \in \mathcal{C}^{\to}(f_1, f_2)$ can be visualized in this way:



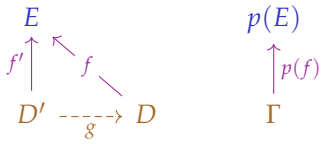
▲

Lemma 3.10 (CodProj)**.** *The codomain projection* $cod : \mathcal{C}^{\to} \to \mathcal{C}$ *and the domain projection* $dom : \mathcal{C}^{\to} \to \mathcal{C}$ *(notation 1.2) are functors from the arrow category (definition 3.9) of* $\mathcal{C}$ *to* $\mathcal{C}$*.*

Proof. The functorality follows directly from morphism composition in $\mathcal{C}$. □

3.1. **Grothendieck fibrations.**

Definition 3.11 (Cartesianness)**.** For arbitrary categories $\mathcal{E}, \mathcal{C}$, a functor $p : \mathcal{E} \to \mathcal{C}$, and objects $\Gamma \in \mathcal{C}$ and $D, E \in \mathcal{E}$, a morphism $f \in \mathcal{E}(D, E)$ is called *Grothendieck cartesian* if for all $D' \in \mathcal{E}$ and $f' \in \mathcal{E}(D', E)$ such that $p(f) = p(f') \in \mathcal{C}(\Gamma, p(E))$ (this implies $p(D) = p(D') = \Gamma \in \mathcal{C}$), there is a unique $g \in \mathcal{E}(D', D)$ such that $p(g) = \mathrm{id}_\Gamma$ and $f' = f \circ g$. We visualize the Grothendieck cartesianness of f as:

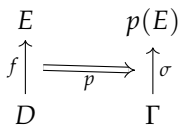


The diagram shows why f is sometimes known as a *terminal lifting*. This definition is due to Jacobs [Jac93, definition 2.1 (i)] and Grothendieck [Gro71]. ▲

History 3.12. The definition 3.11 is called *cartesianness* in [Jac93], but we prefix it with "Grothendieck" because there is another (newer) definition of cartesianness of morphisms in category theory, which is discussed in definition 7.8. Grothendieck cartesianness is also known as *weak cartesianness*, where definition 7.8 is also known as *strong cartesianness*. ▲

Definition 3.13 (Fibration)**.** For arbitrary categories $\mathcal{E}, \mathcal{C}$, objects $\Gamma \in \mathcal{C}$ and $D \in \mathcal{E}$, a functor $p : \mathcal{E} \to \mathcal{C}$ is called a *Grothendieck fibration* or a *fibered category* if composition in $\mathcal{E}$ preserves Grothendieck cartesianness and for every $E \in \mathcal{E}$ and $\sigma \in \mathcal{C}(\Gamma, p(E))$, there exists a Grothendieck cartesian morphism $f \in \mathcal{E}(D, E)$ such

that $p(f) = \sigma$. We visualize the definition as:

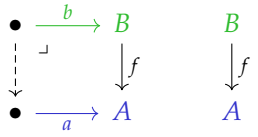


This definition is due to Jacobs [Jac93, definition 2.1 (iii)] and Grothendieck [Gro71]. ⛰

Remark 3.14. A Grothendieck fibration (definition 3.13) is a functor $p : \mathcal{E} \to \mathcal{C}$ such that given a morphism f in $\mathcal{C}$, we can determine some morphisms in $\mathcal{E}$ that are sent to f by p. ✎

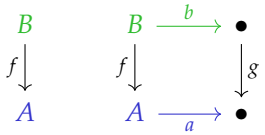
Convention 3.15. We will refer to "Grothendieck fibrations" as "fibrations" in the rest of this introduction since we do not discuss other kinds of fibrations. ⛰

Demonstration 3.16. The codomain projection functor $cod : \mathcal{C}^{\to} \to \mathcal{C}$ (lemma 3.10) is a fibration (definition 3.13) if $\mathcal{C}$ has all pullbacks (definition 2.16). For $a, b \in \mathcal{C}^{\to}$ such that $cod(a) = A \in \mathcal{C}$ and $cod(b) = B \in \mathcal{C}$, we visualize the fact that the cartesianness of $f \in \mathcal{C}(B, A)$ is equivalent to the existence of a pullback:



There is a square and a line in the above visualization. The square is a morphism in $\mathcal{C}^{\to}$ (from a towards b), and the line is a morphism in $\mathcal{C}$. ⛰

Demonstration 3.17. The domain projection functor $dom : \mathcal{C}^{\to} \to \mathcal{C}$ (lemma 3.10) is a fibration (definition 3.13). For $a, b \in \mathcal{C}^{\to}$ such that $dom(a) = A \in \mathcal{C}$ and $dom(b) = B \in \mathcal{C}$, we visualize the fact that $f \in \mathcal{C}(B, A)$ is cartesian:

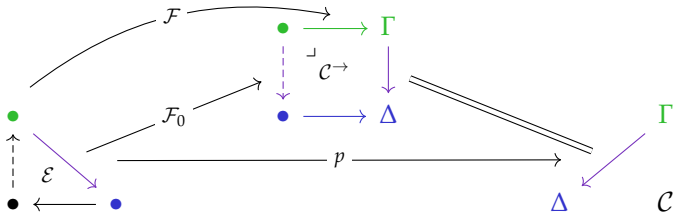


Unlike in demonstration 3.16, $\mathcal{C}$ does not need to have pullbacks because the cartesianness of f only requires the commutativity of the square. ⛰

3.2. **Towards a comprehension category.**

Definition 3.18 (Jacobs' CompCat)**.** A *comprehension category* is a structure consisting of a category (definition 2.25) $\mathcal{C}$, a functor $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ preserving Grothendieck cartesianness of morphisms, and $p := cod \circ \mathcal{F} : \mathcal{E} \to \mathcal{C}$ a fibration (definition 3.13). We denote $\mathcal{F}_0 = dom \circ \mathcal{F}$ due to Jacobs [Jac93, notation 4.2]. The definition of a

comprehension category commutes the following diagram:



We refer to the functor $\mathcal{F}$ as the *(context) comprehension*. The intuition of this name is elaborated in definition 3.20. ☆

Remark 3.19. Although $p := cod \circ \mathcal{F}$ is a fibration in definition 3.18, *cod* is not necessarily a fibration because $\mathcal{C}$ may not have all pullbacks (see demonstration 3.16). We require the functor $\mathcal{F}$ to preserve Grothendieck cartesianness so that only the image of $\mathcal{F}$ are Grothendieck cartesian morphisms. ✎

Definition 3.20 (CtxExt)**.** For a comprehension (definition 3.18) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ and morphism $\sigma \in \mathcal{C}(\Gamma, \Delta)$ such that σ can be uniquely lifted (definition 3.11) to both $\mathcal{E}$ and $\mathcal{C}^{\to}$ since $\mathcal{F}$ preserves Grothendieck cartesianness. In case σ is lifted to a morphism in $\mathcal{E}(A, B)$, we denote $\mathcal{F}(A)$ as π_A so that $\pi_A \in \mathcal{C}_{/\Gamma}$ and refer to them as *display maps*.

For all such object $A \in \mathcal{E}$, we can uniquely determine an object $\pi_A \in \mathcal{C}_{/\Gamma}$ with $\mathcal{F}$, whose domain is another object in $\mathcal{C}$. We refer to this uniquely determined object $dom(\pi_A)$ as the *context extended from* Γ *with* A, denoted $(\Gamma, A) \in \mathcal{C}$.

A visualization of (Γ, A) in the (sub)diagram in definition 3.18, recall that $\mathcal{F}_0 := dom \circ \mathcal{F}$ as mentioned in definition 3.18:

$$\begin{array}{ccccc} A & & \Gamma, A & \xrightarrow{\pi_A} & \Gamma \\ \downarrow & & \downarrow & \lrcorner \; \mathcal{C}^{\to} & \downarrow \sigma \\ B & & \Delta, B & \xrightarrow{\pi_B} & \Delta \end{array}$$

Note that the right-half of the diagram is similar to a rotated version of the diagram in remark 2.30. ☆

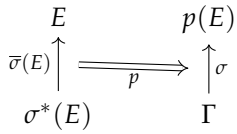
Remark 3.21 (Spoiler I)**.** The definition 3.18 of a comprehension category gives us a basic sketch of a (dependent) type theory:

(1) Recall remark 2.32, we already interpreted contexts and substitutions in $\mathcal{C}$.
(2) Objects in $\mathcal{E}$ should be considered "dependent types". It is in $\mathcal{E}$ that type formation happens.
(3) Introduction and elimination happen in $\mathcal{C}$.
(4) $p(A) \in \mathcal{C}$ is the context where the type A is formed within. So, A is a type formed in the context $p(A)$.
(5) $\mathcal{F}$ allows us to add new types to a context, see definition 3.20.

We complete the interpretation formally in §4. ✎

Definition 3.22 (CartLift)**.** In the diagram in definition 3.13, we define D as $\sigma^*(E)$ since it can be uniquely characterized by σ and E. We refer to f as the *cartesian lifting* of σ to E, denoted $\overline{\sigma}(E) : \mathcal{E}(\sigma^*(E), E)$. The diagram can be redrawn in this

way, using much fewer variables:

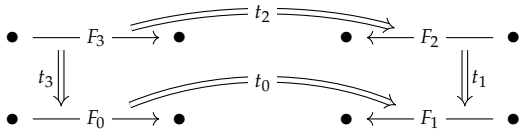


This entire diagram is uniquely determined by only two variables: σ and E. ▲

3.3. **Slightly higher categories.**

Definition 3.23 (2-Cat)**.** A *strict 2-category* $\mathcal{C}$ is a category such that for objects $A, B \in \mathcal{C}$, the hom sets $f, g \in \mathcal{C}(A, B)$ are also categories (called *hom-categories*), and the morphisms in all these categories compose if their domain and codomain match (this includes a lot of cases).

The objects of $\mathcal{C}$ are called *0-cells*, the morphisms of $\mathcal{C}$ are called *1-cells* or *1-morphisms*, and the morphisms in $\mathcal{C}(A, B)$ are called *2-cells* or *2-morphisms*. To visualize a strict 2-category, we draw a random commutative square in a strict 2-category, where for $i \in [0, 3]$, F_i is a 1-cell and t_i is a 2-cell:



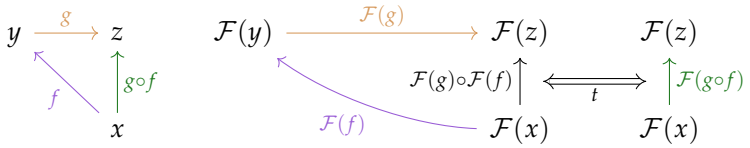
▲

Remark 3.24. A formal definition of strict 2-categories is actually much more complex than definition 3.23, where we omit the preservation of identities and the detailed definition of compositions. By that, we capture the essential features of strict 2-categories and get rid of the bureaucracy of dealing with associators. ✎

Remark 3.25 (Enrichment)**.** A strict 2-category (definition 3.23) is a special case of an *enriched category*. We will define the notion of *enrichment* in §7, demonstration 7.10. ✎

Demonstration 3.26 (Cat)**.** The category of small categories[2], **Cat**, is a strict 2-category (definition 3.23), whose objects are small categories, 1-morphisms are functors, and 2-morphisms are natural transformations. ▲

Definition 3.27 (Pseudofunctor)**.** For strict 2-categories (definition 3.23) C, D and $x, y, z \in C$, a *pseudofunctor* $\mathcal{F} : C \to D$ maps $\mathrm{Ob}(C)$ to $\mathrm{Ob}(D)$ and $C(x, y)$ to $D(\mathcal{F}(x), \mathcal{F}(y))$. In particular, identities in C are mapped to some morphisms in D that are 2-isomorphic to identities, and the natural transformation t in the following diagram must be a natural isomorphism:



▲

[2]If you do not know what are *small* categories, think of them as categories.

Definition 3.28. The "functorality" condition of a pseudofunctor (definition 3.27) is not up to strict equality, but up to natural isomorphism. We refer to this notion as *pseudofunctorality*.

Warning 3.29. A **Cat**-valued (demonstration 3.26) pseudofunctor may not be a presheaf (definition 3.1) since presheaves must be functors (definition 3.27). Maybe we can call it a *pseudopresheaf*, but this terminology is unpopular (yet).

3.4. **Preparation for a category with attributes.**

Definition 3.30 (Subcategory)**.** For a category C, a *subcategory* H of C is a category such that $\mathrm{Ob}(H) \subset \mathrm{Ob}(C)$ and for $A, B \in H$, $H(A,B) \subset C(A,B)$. The identities must exist in the hom sets of H, and the hom sets of H must close under composition.

Definition 3.31 (FullSub)**.** For a category C, a full subcategory H of C is a subcategory (definition 3.30) of C such that the natural inclusion (terminology 1.3) functor $\iota : H \to C$ is fully faithful.

Definition 3.32 (FullSub)**.** This is an alternative to definition 3.31. For a category C, a full subcategory H of C is a subcategory of C such that for every $A, B \in H$ and every $f \in C(A,B)$, $f \in H(A,B)$.

Definition 3.33 (Fiber)**.** For categories $\mathcal{E}, \mathcal{C}$ and a functor $\mathcal{F} : \mathcal{E} \to \mathcal{C}$, the *fiber* of $\mathcal{F}$ over an object $\Gamma \in \mathcal{C}$ (denoted as $\mathcal{E}_\Gamma$) is a full subcategory (definitions 3.31 and 3.32) of $\mathcal{E}$ whose objects are mapped to Γ by $\mathcal{F}$ and morphisms are mapped to id_Γ by $\mathcal{F}$.

Terminology 3.34. In [Jac93, definition 2.2], Jacobs said that $\mathcal{E}_\Gamma$ (definition 3.33) has objects *above* $\mathcal{C}$, and it has *vertical* morphisms. The intuition of these terminologies is visualized in a rotated version of the diagram in definition 3.11, where the (E, D, D')-triangle is "above" the $(\Gamma, p(E))$-line.

Definition 3.35 (Reindex)**.** For a fibration $p : \mathcal{E} \to \mathcal{C}$, morphisms $\sigma \in \mathcal{C}(\Gamma, \Delta)$, and $f \in \mathcal{E}_\Delta(D, E)$ (definition 3.33) (so that $p(D) = p(E) = \Delta \in \mathcal{C}$ and $p(f) = \mathrm{id}_\Delta$), we can take the cartesian lifting (definition 3.22) of σ to both D and E – the objects $\sigma^*(D), \sigma^*(E) \in \mathcal{E}_\Gamma$, to characterize the morphism $\sigma^*(f)$ in the following pullback square:

$$\begin{array}{ccc} \sigma^*(D) & \xrightarrow{\sigma^*(f)} & \sigma^*(E) \\ {\scriptstyle \overline{\sigma}(D)}\downarrow & \lrcorner & \downarrow{\scriptstyle \overline{\sigma}(E)} \\ D & \xrightarrow{\quad f \quad} & E \end{array} \qquad \begin{array}{c} \Gamma \\ \downarrow{\scriptstyle \sigma} \\ \Delta \end{array}$$

We can observe that the cartesian lifting of $\sigma \in \mathcal{C}(\Gamma, \Delta)$ to objects in $\mathcal{E}_\Delta$ are objects in $\mathcal{E}_\Gamma$, and the morphisms in $\mathcal{E}_\Delta$ can also be lifted. Therefore we obtain a functor $\sigma^* : \mathcal{E}_\Delta \to \mathcal{E}_\Gamma$ for every $\sigma \in \mathcal{C}(\Gamma, \Delta)$, called the *reindexing* functor.

Exercise 3.36. Define and prove the universal property of the pullback square in definition 3.35.

Remark 3.37 (Spoiler II)**.** In definition 3.35, the morphism $\sigma \in \mathcal{C}(\Gamma, \Delta)$ induces the functor $\sigma^* : \mathcal{E}_\Delta \to \mathcal{E}_\Gamma$. The second last deduction in remark 2.7 and the diagram in remark 2.30 show that given a substitution $\Gamma \vdash \sigma : \Delta$, it can bring the type A in Δ to Γ. Recall the previous spoiler remark 3.21, We can think of σ^* as the "action to apply the substitution σ on a type".

Definition 3.38 (Cleavage)**.** For a fibration $p : \mathcal{E} \to \mathcal{C}$, a particular choice of σ^* and $\overline{\sigma}$ (definition 3.22) for every $\sigma \in \mathcal{C}(\Gamma, \Delta)$ is called a *cleavage* of p. In other words, a cleavage of p is a mapping from σ to the pair $(\sigma^*, \overline{\sigma})$.

A cleavage of p is equivalent to a subcategory (definition 3.30) of $\mathcal{E}$. ⛰

Remark 3.39. In definition 3.13, a fibration $p : \mathcal{E} \to \mathcal{C}$ is equipped with an operation that takes a morphism σ and returns $p^{-1}(\sigma)$ a Grothendieck cartesian morphism. To some extent, p is an "invertible" functor: for a morphism f' in $\mathcal{E}$, we can send it to $\mathcal{C}$ by p and take it back to a cleavage (definition 3.38) f in $\mathcal{E}$. The relationship between f' and f is similar to the morphisms of the same name in definition 3.11. ✎

Lemma 3.40 (Faker)**.** *For a fibration $p : \mathcal{E} \to \mathcal{C}$, we have a pseudofunctor (definition 3.27) $\Psi : \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$ by $\Psi(\Gamma) := \mathcal{E}_\Gamma$ and for $\sigma \in \mathcal{C}(\Gamma, \Delta)$, $\Psi(\sigma) := \sigma^*$.*

Proof. We verify the pseudofunctorality of Ψ:

- For $\Gamma \in \mathcal{C}$, $\Psi(\mathrm{id}_\Gamma) = \mathrm{id}_\Gamma^* \simeq \mathrm{id}_{\mathcal{E}_\Gamma}$ a natural isomorphism.
- For $\sigma \in \mathcal{C}(\Gamma', \Gamma'')$ and $\gamma \in \mathcal{C}(\Gamma, \Gamma')$ (so that $\sigma \circ \gamma \in \mathcal{C}(\Gamma, \Gamma'')$), we know $\Psi(\sigma \circ \gamma) = (\sigma \circ \gamma)^*$ and $\sigma^* \circ \gamma^* = \Psi(\sigma) \circ \Psi(\gamma)$, and there is $(\sigma \circ \gamma)^* \simeq \sigma^* \circ \gamma^*$ a natural isomorphism. □

Definition 3.41 (Split)**.** We say a fibration $p : \mathcal{E} \to \mathcal{C}$ is *split* or *cloven* if the pseudofunctor Ψ (lemma 3.40) defined on it satisfies functorality instead of pseudofunctorality. This requires the following strict equalities:

- For $\Gamma \in \mathcal{C}$, $\mathrm{id}_\Gamma^* = \mathrm{id}_{\mathcal{E}_\Gamma}$.
- For $\sigma \in \mathcal{C}(\Gamma', \Gamma'')$ and $\gamma \in \mathcal{C}(\Gamma, \Gamma')$, $(\sigma \circ \gamma)^* = \sigma^* \circ \gamma^*$.

⛰

Demonstration 3.42. Ψ in a split fibration (definition 3.41) is a **Cat**-valued presheaf (definition 3.1). ⛰

Remark 3.43 (Coherence)**.** If our comprehension category is not split, we will have substitutions up to isomorphism instead of strict equality as discussed by Curien [Cur90]. ✎

Terminology 3.44. We say a comprehension category (definition 3.18) to be *full* if the comprehension $\mathcal{F}$ is fully faithful, and *split* if the underlying fibration p is split (definition 3.41). ⛰

3.5. **Cartmell's artifacts.**

Definition 3.45 (Cartmell's CwA)**.** A *category with attributes* (CwA) is a full split (terminology 3.44) comprehension category. Note that the codomain $\mathcal{C}$ of the comprehension has a terminal object.

We refer to the domain $\mathcal{E}$ of the comprehension as the *attributes*. ☆

History 3.46. CwA was already well-established when comprehension category was discussed. The original definition of CwA is more similar to type categories (definition 5.6) where the dependent types indexed by an object in $\mathcal{C}$ forms a set instead of a fiber in another category $\mathcal{E}$. In this case, it is *not* (but similar to) a full split comprehension category. We discuss the difference in remark 5.16.

Treating a CwA as a full split comprehension category is accepted in recent years, though. ⛰

Definition 3.47 (Contextuality)**.** A CwA (definition 3.45) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ is *contextual* if there exists a *length* function $\ell : \mathcal{C} \to \mathbb{N}$ such that:

(1) $\ell(1) = 0$, and for every $\Gamma \in \mathcal{C}$, $\ell(\Gamma) = 0 \iff \Gamma = 1$ holds. The terminal object 1 must be unique.
(2) For every $\Gamma \in \mathcal{C}$ and $A \in \mathcal{E}_\Gamma$, $\ell(\Gamma, A) = \ell(\Gamma) + 1$ holds.
(3) For every $\Delta \in \mathcal{C}$ where $\ell(\Delta) > 0$, there exists uniquely $\Gamma \in \mathcal{C}$ and $A \in \mathcal{E}_\Gamma$ such that $\Delta = (\Gamma, A)$.

In other words, the objects in $\mathcal{C}$ are *generated* from the terminal object and context extension (definition 3.20), and we can do *induction* on it.

A similar definition can be found in [CCD19, definition 2]. ⛰

Definition 3.48 (Cartmell's CxlCat)**.** A CwA satisfying contextuality (definition 3.47) is called a *contextual category*. This is essentially the definition in [Car86, §14], where objects are described as a tree. ☆

Remark 3.49. In [Sco, definition 7], the length function (definition 3.47) is described as a *grading* of objects in $\mathcal{C}$, where the objects are indexed by a natural number. It is denoted $\mathrm{Ob}(\mathcal{C}) := \coprod_{n:\mathbb{N}} \mathrm{Ob}_n(\mathcal{C})$. ✎

Definition 3.50 (Democracy)**.** A CwA (definition 3.45) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ is *democratic* if for every $\Gamma \in \mathcal{C}$ there is an object $A \in \mathcal{E}_1$ such that $\Gamma \simeq (1, A)$. This definition is due to [CCD19, definition 3]. ⛰

4. Categorical semantics

Definition 4.1 (TT)**.** A type theory consists of the following constructions, defined by *typing judgments*:

(1) Contexts, whose well-formedness is denoted $\Gamma \vdash$. There is a well-formed *empty context* 1.
(2) Types formed in contexts, denoted $\boxed{\Gamma \vdash A \text{ type}}$, and type equality, denoted $\boxed{\Gamma \vdash A = B \text{ type}}$.
(3) Context extension. Given $\boxed{\Gamma \vdash A \text{ type}}$, we can add A to the end of Γ to form a new context. We denote this process using the following rule:
$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type}}{\Gamma, A \vdash}$$
(4) Substitution objects, whose well-formedness is denoted $\Gamma \vdash \sigma : \Delta$. There is a well-formed *empty substitution* $\Gamma \vdash [] : 1$.
(5) Typing of terms. Given A formed in Γ, "term u is of type A" and "terms u and v are equal instances of type A" are described as:
$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type}}{\Gamma \vdash u : A} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A}{\Gamma \vdash u = v : A}$$
(6) Substitution extension. This is similar to context extension:
$$\frac{\Gamma \vdash \qquad \Delta \vdash \qquad \Delta \vdash A \text{ type} \qquad \Gamma \vdash \sigma : \Delta \qquad \Gamma \vdash a : A\sigma}{\Gamma \vdash (\sigma, a) : (\Delta, A)}$$

⛰

Remark 4.2. The type theory defined in definition 4.1 can be extended in many ways. We only discuss the essential parts here: the formation of contexts, dependent types, terms, and substitutions.

In definition 4.16 we extend the type theory with the Cartesian product type, and more constructions in the literature of type theory will be added in §6. ✎

Warning 4.3 (Universe)**.** We do *not* have any type that represents the *universe* type that usually appear in a type theory, such as the *intuitionistic type theory* due to Martin-Löf [Mar75], the *calculus of constructions* due to Coquand [Coq85], and the *extended calculus of constructions* due to Luo [Luo90; Luo94].

We *cannot* have a type $\boxed{\vdash \mathcal{U} \text{ type}}$ for all types (including $\vdash \mathcal{U} : \mathcal{U}$) due to the paradox construction by Girard [Gir72] (and later simplified by Hurkens [Hur95]). ☠

Definition 4.4. We interpret contexts and substitutions in a CwA (definition 3.45).

(1) A context Γ is interpreted as an object in $\mathcal{C}$, denoted $[\![\Gamma]\!] \in \mathcal{C}$. In particular, $[\![1]\!] := 1$, the terminal object in $\mathcal{C}$ which is said to exists in definition 2.25.
(2) A substitution $\Gamma \vdash \sigma : \Delta$ is interpreted as a morphism in $\mathcal{C}$, denoted $[\![\sigma]\!] : \mathcal{C}([\![\Gamma]\!], [\![\Delta]\!])$. The empty substitution is the morphism from an object to the terminal object 1, which always exists by the universal property of terminal objects. ⛰

Definition 4.5. We interpret types and terms with their equalities in a CwA (definition 3.45). For a context Γ in type theory and its interpretation $[\![\Gamma]\!] \in \mathcal{C}$,

(1) The judgment $\boxed{\Gamma \vdash A \text{ type}}$ is interpreted as an object $[\![A]\!] \in \mathcal{E}_{[\![\Gamma]\!]}$.
(2) Two types A, B formed in context Γ are said to be equal (the judgment $\boxed{\Gamma \vdash A = B \text{ type}}$) if their interpretations are uniquely isomorphic objects in the fiber (definition 3.33) $\mathcal{E}_{[\![\Gamma]\!]}$.
(3) Context extension $[\![\Gamma, A]\!]$ is defined as $([\![\Gamma]\!], [\![A]\!]) \in \mathcal{C}$ using the construction in definition 3.20.
(4) We interpret judgments $\Gamma \vdash a : A$ as $\Gamma \vdash \sigma : (\Gamma, A)$ for some σ. We require σ to satisfy that $\pi_{[\![A]\!]} \circ [\![\sigma]\!] = \mathrm{id}_{[\![\Gamma]\!]}$ (note that $[\![\sigma]\!] \in \mathcal{C}([\![\Gamma]\!], [\![\Gamma, A]\!])$).
(5) Two terms u, v of the same type A are said to be equal (the judgment $\Gamma \vdash u = v : A$) if their interpretations are uniquely isomorphic objects in the overcategory $\mathcal{C}_{/[\![\Gamma]\!],[\![A]\!]}$ (note that their domains are both $[\![\Gamma]\!]$). ⛰

Remark 4.6. The interpretation of term/type equality in definition 4.5 is slightly different from the one in [Pit01, §6.4], where they require *sameness* instead of unique isomorphism. ✎

Terminology 4.7. In definition 4.5, the substitutions $\sigma \in \mathcal{C}(\Gamma, (\Gamma, A))$ such that $\pi_A \circ \sigma = \mathrm{id}_\Gamma$ are called the *global sections* of A. ⛰

Exercise 4.8. Interpret substitution extension in a CwA.

Exercise 4.9 (Weakening)**.** Interpret in a CwA the context weakening rule:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash B \text{ type}}{\Gamma, B \vdash u : A}$$

Remark 4.10 (Strictness)**.** The substitution operation is interpreted as morphisms in $\mathcal{C}$. So, if two contexts are equivalent up to substitution, their cartesian lifting (definition 3.22) will be the same (instead of being naturally isomorphic categories) thanks to the strictness condition in definition 3.41. ✎

4.1. **Star at the top: the unit type.**

Definition 4.11 (Unit)**.** We extend definition 4.1 with the unit type:

$$\frac{}{\vdash \top \text{ type}} \qquad \frac{}{\vdash \star : \top} \qquad \frac{\vdash u : \top}{\vdash u = \star : \top}$$

We do not define the $\top$ type with an elimination rule. Instead, we directly give it a uniqueness-rule (the η-equality) for convenience. ▲

Definition 4.12. We interpret the unit type in definition 4.11 in a CwA:

(1) The type $\top$ corresponds to an object $[\![\top]\!] \in \mathcal{E}$ that is sent to the terminal object $1 \in \mathcal{C}$ by the comprehension (definition 3.18). In other words, $\mathcal{F}_0([\![\top]\!]) = 1 \in \mathcal{C}$.
(2) $\mathcal{F}_0([\![\top]\!])$ interprets the *context 1 extended by* $\top$, denoted $(1, [\![\top]\!]) \in \mathcal{C}$. Note that $(1, [\![\top]\!]) = 1$.
(3) The introduction rule $\vdash \star : \top$ is interpreted as the identity morphism id_1.
(4) The uniqueness rule of $\top$ is derived from the uniqueness of identity morphisms: for all morphisms in $\mathcal{C}(1, (1, [\![\top]\!]))$, they are the same as the given identity morphism.

$$[\![\top]\!] \longrightarrow \mathcal{F}_0 \longrightarrow 1, [\![\top]\!] \underset{\pi_1}{\overset{[\![\star]\!]}{\rightleftarrows}} 1$$

The above diagram shows the interpretations of $\top$ and $\star$. ▲

Remark 4.13. The fiber (definition 3.33) $\mathcal{C}_1$ corresponds to the notion of *closed types* in a type theory – types defined in the empty context. The $\top$ type (definition 4.12) is an example of such type. For any $\Delta \in \mathcal{C}$ and $A \in \mathcal{E}_\Delta$, a morphism $\sigma \in \mathcal{C}(1, \Delta)$ is a *closed substitution*, and the diagram in definition 4.12 can be added with Δ and σ to form the following commutative diagram:

$$\begin{array}{ccccccc}
[\![\top]\!] & \longrightarrow \mathcal{F}_0 \longrightarrow & 1, [\![\top]\!] & \underset{\pi_1}{\overset{[\![\star]\!]}{\rightleftarrows}} & 1 \\
 & & \downarrow & \lrcorner & \downarrow \sigma \\
A & \longrightarrow \mathcal{F}_0 \longrightarrow & \Delta, A & \longrightarrow \pi_A \longrightarrow & \Delta
\end{array}$$

✎

Exercise 4.14. Modify the interpretation in definition 4.12 to adapt the following, alternative definition of the $\top$ type:

$$\frac{\Gamma \vdash}{\Gamma \vdash \top} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \star : \top} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash u : \top}{\Gamma \vdash u = \star : \top}$$

The diagram is similar to the one in remark 4.13:

$$\begin{array}{ccccccc}
[\![\top]\!] & \longrightarrow \mathcal{F}_0 \longrightarrow & \Gamma, [\![\top]\!] & \underset{\pi_{[\![\top]\!]}}{\overset{[\![\star]\!]}{\rightleftarrows}} & \Gamma \\
 & & \downarrow & \lrcorner & \downarrow \\
A & \longrightarrow \mathcal{F}_0 \longrightarrow & \Delta, A & \longrightarrow \pi_A \longrightarrow & \Delta
\end{array}$$

This version of $\top$ is similar to a combination of definition 4.12 and exercise 4.9.

Remark 4.15. In definition 4.12, we interpret the formation, introduction, and elimination rules for a type theoretical construction. Then, we *prove* the uniqueness rule. If there are other equalities that we can prove in the categorical model but cannot be derived from the type theory, we may add these equalities to make the type theory more convenient to work with. This is a benefit we can gain from categorical models of type theory.

Unfortunately, the $\top$ type is too simple. In §4.2, we are going to see such an equality in theorema 4.18. ✎

4.2. **Like a semiring: the (co)product type.**

Definition 4.16 (Product)**.** We extend definition 4.1 with the (non-dependent) product type, using the same notation as product objects (definition 2.11):

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash (A \times B) \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash (A \times B) \text{ type} \qquad \Gamma \vdash u : A \times B}{\Gamma \vdash u.1 : A \quad \text{and} \quad \Gamma \vdash u.2 : B}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash \langle u, v \rangle : A \times B}$$

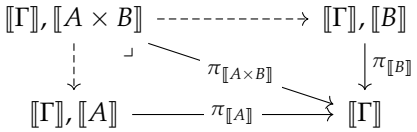
$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash \langle u, v \rangle.1 = u : A \quad \text{and} \quad \Gamma \vdash \langle u, v \rangle.2 = v : B}$$

▲

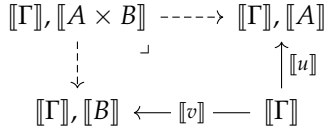
Definition 4.17. The *context* Γ *extended by the product type* (definition 4.16) $A \times B$, $(\Gamma, A \times B)$, is interpreted as the pullback (definition 2.16) $(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket) \times_\Gamma (\llbracket \Gamma \rrbracket, \llbracket B \rrbracket) \in \mathcal{C}$, denoted $(\llbracket \Gamma \rrbracket, \llbracket A \times B \rrbracket) \in \mathcal{C}$ in a contextual category (definition 3.48).

Note that we are not talking about the product type itself, but instead, we define contexts *extended by* a product type. By contextuality (definition 3.47), there is an object $\llbracket A \times B \rrbracket \in \mathcal{E}$ that corresponds to this extended context.

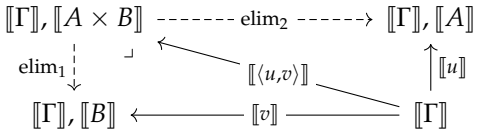
The interpretation of the formation rule is visualized below:



In the introduction rule, the inputs are $\llbracket u \rrbracket$, a global section (terminology 4.7) of $\llbracket A \rrbracket$, and $\llbracket v \rrbracket$, a global section of $\llbracket B \rrbracket$.



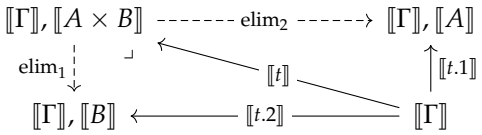
The introduction rule itself corresponds to the product morphism of $[\![u]\!]$ and $[\![v]\!]$. The elimination rules are directly derived from the property of the product object.



Theorema 4.18 (Uniqueness)**.** *The interpretation in definition 4.17 satisfies the η-rule for the product type:*

$$\frac{\Gamma \vdash (A \times B)\ type \qquad \Gamma \vdash t : A \times B}{\Gamma \vdash t = \langle t.1, t.2 \rangle : A \times B}$$

Proof. By doing some replacements of terms in the last diagram in definition 4.17:



□

Definition 4.19 (Coproduct)**.** We extend definition 4.1 with the coproduct type:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A\ \text{type} \qquad \Gamma \vdash B\ \text{type}}{\Gamma \vdash (A \sqcup B)\ \text{type}}$$

$$\frac{\Gamma \vdash (A \sqcup B)\ type \qquad \Gamma \vdash A\ type \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{inl}(u) : A \sqcup B}$$

$$\frac{\Gamma \vdash (A \sqcup B)\ type \qquad \Gamma \vdash B\ type \qquad \Gamma \vdash u : B}{\Gamma \vdash \mathsf{inr}(u) : A \sqcup B}$$

$$\frac{\begin{array}{c}\Gamma \vdash (A \sqcup B)\ type \qquad \Gamma \vdash A\ type \qquad \Gamma \vdash B\ type \\ \Gamma \vdash C\ \text{type} \qquad \Gamma, A \vdash u : C \qquad \Gamma, B \vdash v : C \qquad \Gamma \vdash t : A \sqcup B\end{array}}{\Gamma \vdash \mathsf{match}(t, u, v) : C}$$

$$\frac{\begin{array}{ccc}\Gamma \vdash A\ type & \Gamma \vdash B\ type & \Gamma \vdash C\ type \\ \Gamma, A \vdash u : C & \Gamma, B \vdash v : C & \Gamma \vdash t : A\end{array}}{\Gamma \vdash \mathsf{match}(\mathsf{inl}(t), u, v) = \boxed{ut} : C}$$

$$\frac{\begin{array}{ccc}\Gamma \vdash A\ type & \Gamma \vdash B\ type & \Gamma \vdash C\ type \\ \Gamma, A \vdash u : C & \Gamma, B \vdash v : C & \Gamma \vdash t : B\end{array}}{\Gamma \vdash \mathsf{match}(\mathsf{inr}(t), u, v) = \boxed{vt} : C}$$

Note that in the two rules, $\boxed{ut}$ and $\boxed{vt}$ denote the operation of "applying the singleton substitution object t" to the term u and v, respectively.

Remark 4.20. Type theoretically, the coproduct type is dual to the product type. Since product types are interpreted as pullbacks (see definition 4.17) which are

"products in the overcategory", the coproduct types should look like "coproducts in the overcategory". ✎

Exercise 4.21. Interpret the coproduct type (definition 4.19) in a contextual category (definition 3.48). Beware of warning 7.14.

4.3. **Internal homs and the evaluation map.**

Definition 4.22 (StrMonCat)**.** A category $\mathcal{C}$ is *strictly monoidal* if there is an operation (called the *tensor product*) $\otimes$ such that for $A, B \in \mathcal{C}$, $A \otimes B \in \mathcal{C}$. Apart from that, the following additional properties must hold:

(1) For $A, B, C, D \in \mathcal{C}$, $f \in \mathcal{C}(A, C)$, and $g \in \mathcal{C}(B, D)$, there is a morphism $f \otimes g \in \mathcal{C}(A \otimes B, C \otimes D)$.
(2) There is an object $1 \in \mathcal{C}$, called the *tensor unit*, satisfying that for every $A \in \mathcal{C}$, $A \otimes 1 = 1 \otimes A = A$.
(3) The tensor product of objects and morphisms are strictly associative.

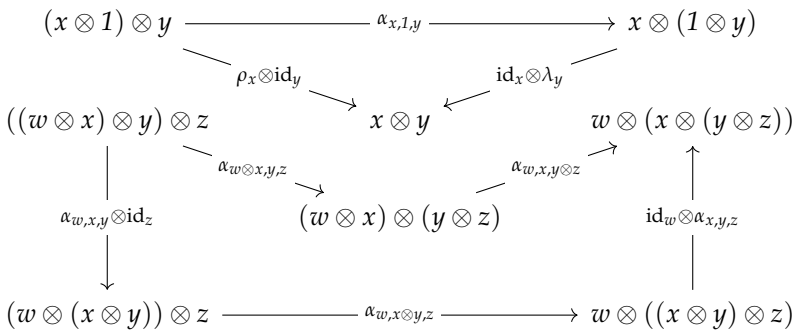
▲

Definition 4.23 (Bifunctor)**.** For (small) categories $\mathcal{C}_1, \mathcal{C}_2, \mathcal{D} \in \mathbf{Cat}$, we take the product object (definition 2.11) $\mathcal{C}_1 \times \mathcal{C}_2$ and refer to a functor $\mathcal{F}$ as a *bifunctor* if it of form $\mathcal{F} : \mathcal{C}_1 \times \mathcal{C}_2 \to \mathcal{D}$. We say that $\mathcal{F}$ is a bifunctor *from* $\mathcal{C}_1, \mathcal{C}_2$ *to* $\mathcal{D}$. ▲

Definition 4.24 (MonCat)**.** The definition 4.22 can be loosen into the general notion of *monoidal categories* in order to allow tensor products that are not strictly commutative (but up to isomorphism). We define $\otimes$ as a bifunctor (definition 4.23) $\mathcal{C} \otimes \mathcal{C} \to \mathcal{C}$, also called the *tensor product*, with the following additional structures:

(1) The *tensor unit* object $1 \in \mathcal{C}$.
(2) For $x, y, z \in \mathcal{C}$, an isomorphism $\alpha_{x,y,z} \in \mathcal{C}((x \otimes y) \otimes z, x \otimes (y \otimes z))$.
(3) For $x \in \mathcal{C}$, two isomorphisms $\rho_x \in \mathcal{C}(x \otimes 1, x)$ and $\lambda_x \in \mathcal{C}(1 \otimes x, x)$.

These structures commute the following diagrams:



▲

Definition 4.25 (CartMonCat)**.** We say a monoidal category (definition 4.24) to be *cartesian* if its tensor products are the product objects (definition 2.11) and its tensor unit is the terminal object (definition 2.9). ▲

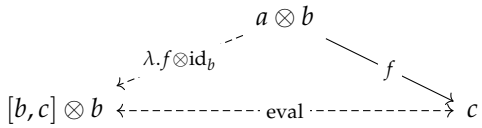
Definition 4.26 (SymMonCat)**.** We say a monoidal category (definition 4.24) to be *symmetric* if its tensor product is commutative up to isomorphism. This definition is slightly weaker than the conventional one, but it is simpler. ▲

Definition 4.27 (ClosedSMCat)**.** We say a symmetric monoidal category $\mathcal{C}$ (definition 4.26) to be *closed* if for objects $a, b, c \in \mathcal{C}$, there is an object $[b, c] \in \mathcal{C}$ (called an *internal hom*) such that following three pairs of hom presheaves (definition 3.3) are naturally isomorphic:

$$\begin{aligned} \mathcal{C}(- \otimes b, c) &\simeq \mathcal{C}(-, [b, c]) \\ \mathcal{C}(a \otimes -, c) &\simeq \mathcal{C}(a, [-, c]) \\ \mathcal{C}(a \otimes b, -) &\simeq \mathcal{C}(a, [b, -]) \end{aligned}$$

This is just an unnecessarily complicated way of saying that there is an equivalence $\mathcal{C}(a \otimes b, c) \simeq \mathcal{C}(a, [b, c])$. ▲

Definition 4.28 (EvalMap)**.** In a symmetric monoidal closed category (definition 4.27) $\mathcal{C}$, its objects $a, b, c \in \mathcal{C}$, and a morphism $f \in \mathcal{C}(a \otimes b, c)$, by the equivalences in definition 4.27 we can obtain the unique morphism $\lambda.f \in \mathcal{C}(a, [b, c])$ commuting the following diagram:



The morphism eval is therefore a unique isomorphism. We refer to this morphism as the *evaluation map* for the internal hom $[b, c]$. ▲

Lemma 4.29 (Internalization)**.** *In a symmetric monoidal closed category (definition 4.27) $\mathcal{C}$, its tensor unit (definition 4.24) $1 \in \mathcal{C}$, and objects $b, c \in \mathcal{C}$, there is an isomorphism $\mathcal{C}(b, c) \simeq \mathcal{C}(1, [b, c])$.*

Proof. By specializing the object a in the equivalence in definition 4.27 to 1:

$$\mathcal{C}(1, [b, c]) \simeq \mathcal{C}(1 \otimes b, c) \simeq \mathcal{C}(b, c)$$

□

Lemma 4.30. *The tersor unit used in lemma 4.29 can be replaced with any object $a \in \mathcal{C}$ such that $a \otimes b \simeq b$.*

4.4. Locally cartesian closed contextual categories and fiber exponents.

Definition 4.31 (CCC)**.** We say a symmetric closed monoidal category (definition 4.27) to be *cartesian* if it is also cartesian monoidal (definition 4.25). In this case, the category is called a cartesian closed category, or CCC. The internal homs of a CCC are also known as *exponential objects* and we denote $[b, c]$ as c^b. ☆

Demonstration 4.32 (FunctionSet)**.** Since functions are sets too and they satisfy the natural isomorphisms in definition 4.27, **Set** is a CCC (definition 4.31). The evaluation map (definition 4.28) in **Set** is just function application: for $A, B \in \mathbf{Set}$, the internal hom (definition 4.27) $A \to B$ has the evaluation map $\mathrm{eval} \in \mathbf{Set}((A \to B) \times A, B)$. ▲

Lemma 4.33. *In a CwA (definition 3.45) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ where $\mathcal{C}$ has pullbacks (in the sense of product types as in definition 4.17), for $\Gamma \in \mathcal{C}$ and $A \in \mathcal{E}_\Gamma$, $\Gamma \times_\Gamma (\Gamma, A) \simeq \Gamma, A$.*

Proof. By the second projection of the pullback in the forward direction and $\pi_A \times_\Gamma \mathrm{id}_{\Gamma, A}$ in the backward direction. □

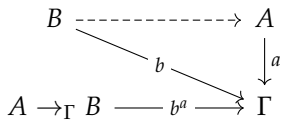
Definition 4.34 (LCCC)**.** We say a category $\mathcal{C}$ to be a *locally cartesian closed* category or an LCCC if the overcategory $\mathcal{C}_{/\Gamma}$ for every $\Gamma \in \mathcal{C}$ is cartesian closed. The relation between LCCC and type theories has been explored by Seely and Hofmann [See84; Hof94]. ▲

Lemma 4.35. *An LCCC (definition 4.34) is a CCC (definition 4.31) if it has a terminal object.*

Proof. Consider an LCCC $\mathcal{C}$ with a terminal object 1, we know that $\mathcal{C}_{/1}$ is a CCC. By lemma 3.7, $\mathcal{C}$ is a CCC. □

Lemma 4.36. *In an LCCC (definition 4.34) $\mathcal{C}$ and $\Gamma, A, B \in \mathcal{C}$, the cartesian product in $\mathcal{C}_\Gamma$ is given by the pullback $A \times_\Gamma B \in \mathcal{C}$.*

Definition 4.37 (FiberExp)**.** In an LCCC (definition 4.34) $\mathcal{C}$, for $\Gamma, A, B \in \mathcal{C}$, $a \in \mathcal{C}(A, \Gamma)$, and $b \in \mathcal{C}(B, \Gamma)$ (so that $a, b \in \mathcal{C}_{/\Gamma}$), the *fiber exponent* $A \to_\Gamma B$ is the domain of the exponential object $b^a \in \mathcal{C}_{/\Gamma}$. The name is inspired from the alias "fiber product" of pullbacks (definition 2.16) and "fiber coproduct" of pushouts (definition 7.1). Visualization:



Note that by lemma 4.29, there is an isomorphism $\mathcal{C}_{/\Gamma}(a, b) \simeq \mathcal{C}_{/\Gamma}(1_{\mathcal{C}_{/\Gamma}}, b^a)$. ▲

4.5. **Simple type theory: the function type.**

Definition 4.38 (Function)**.** We extend definition 4.1 with the function type, defined by the following rules:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash (A \to B) \text{ type}}$$

$$\frac{\Gamma \vdash (A \to B) \text{ type} \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma, A \vdash u : B}{\Gamma \vdash \lambda.u : A \to B}$$

$$\frac{\begin{array}{c}\Gamma \vdash (A \to B) \text{ type} \\ \Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash u : A \to B \qquad \Gamma \vdash v : A\end{array}}{\Gamma \vdash \mathsf{ap}(u, v) : B}$$

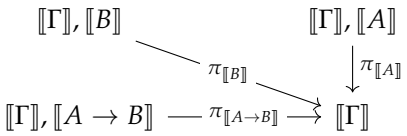
$$\frac{\Gamma, A \vdash \qquad \Gamma \vdash B \text{ type} \qquad \Gamma, A \vdash u : B \qquad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{ap}(\lambda.u, v) = uv : B}$$

▲

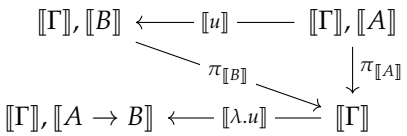
Definition 4.39. In a contextual category (definition 3.48) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^\to$ where $\mathcal{C}$ is an LCCC, we can interpret definition 4.38.

Consider types $\boxed{\Gamma \vdash A \text{ type}, \Gamma \vdash B \text{ type}, \Gamma \vdash C \text{ type}}$. The formation of *context* Γ *extended by the function type* $A \to B$ is the fiber exponent (definition 4.37) $(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket) \to_{\llbracket \Gamma \rrbracket}$

$(\llbracket\Gamma\rrbracket, \llbracket B\rrbracket)$, denoted $(\llbracket\Gamma\rrbracket, \llbracket A \to B\rrbracket) \in \mathcal{C}$, whose display map (definition 3.20) is given by the exponential object $\pi_{\llbracket A\to B\rrbracket} = \pi_{\llbracket B\rrbracket}^{\pi_{\llbracket A\rrbracket}} \in \mathcal{C}_{/\llbracket\Gamma\rrbracket}$.

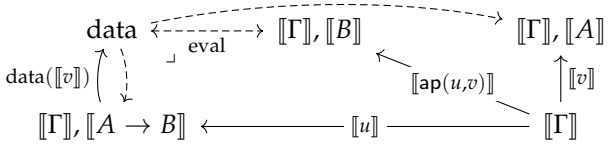


The introduction rule takes a morphism $\llbracket u\rrbracket \in \mathcal{C}((\llbracket\Gamma\rrbracket, \llbracket A\rrbracket), (\llbracket\Gamma\rrbracket, \llbracket B\rrbracket))$ such that $\pi_{\llbracket B\rrbracket} \circ \llbracket u\rrbracket = \pi_{\llbracket A\rrbracket}$ to the morphism $\llbracket\lambda.u\rrbracket \in \mathcal{C}(\llbracket\Gamma\rrbracket, (\llbracket\Gamma\rrbracket, \llbracket A \to B\rrbracket))$ by the isomorphism in definition 4.37:

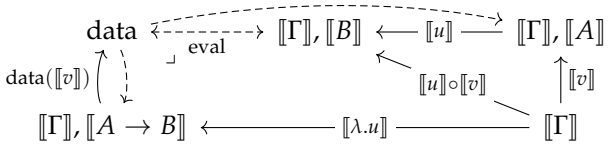


The rest of the rules involve the fiber product $(\llbracket\Gamma\rrbracket, \llbracket A \to B\rrbracket) \times_{\llbracket\Gamma\rrbracket} (\llbracket\Gamma\rrbracket, \llbracket A\rrbracket) \in \mathcal{C}$. For simplicity, we denote it as "data", and we denote $\mathrm{id}_{\llbracket\Gamma\rrbracket, \llbracket A\to B\rrbracket} \times_{\llbracket\Gamma\rrbracket} (f \circ \pi_{\llbracket A\to B\rrbracket})$ as $\mathrm{data}(f)$ for morphism $f \in \mathcal{C}(\llbracket\Gamma\rrbracket, (\llbracket\Gamma\rrbracket, \llbracket A\rrbracket))$.

The elimination rule is interpreted by morphism composition and the evaluation map (definition 4.28), visualized as the following diagram which commutes everywhere:



The fact that the evaluation map (definition 4.28) is a unique isomorphism shows that the β-equality of this interpretation holds:



5. Alternative constructions

Definition 5.1 (DiscreteCat)**.** We say a category $\mathcal{C}$ to be *discrete* if it has no morphism other than identity morphisms.

Proposition 5.2. *$\mathcal{C}$ is discrete* $\iff \mathcal{C} \simeq Ob(\mathcal{C})$.

Definition 5.3 (Discreteness)**.** We say a fibration $p : \mathcal{E} \to \mathcal{C}$ to be *discrete* if each fiber (definition 3.33) is discrete (definition 5.1).

Remark 5.4. In the literature, we sometimes ask the fibration p of a CwA (definition 3.45) to be a discrete fibration (definition 5.3). This is what Pitts [Pit01, §6] has done implicitly (without mentioning the notion of fibrations (definition 3.13) and comprehension categories (definition 3.18)).

Definition 5.5 (Fam)**.** The category of indexed families of sets, **Fam**, consists of the following:

(1) Objects $(U_x)_{x\in X}$, where X is a set and for each $x \in X$, U_x is a set.
(2) Morphisms $(f, (g_x)_{x\in X}) \in \mathbf{Fam}((U_x)_{x\in X}, (V_y)_{y\in Y})$, where $f : X \to Y$ is a function (called the *reindexing function*, similar to definition 3.35) and for each $x \in X$, $g_x : U_x \to V_{f(x)}$ is a function.

5.1. Type categories.

Definition 5.6 (Pitts' TypeCat)**.** A *type category* consists of:

(1) A category $\mathcal{C}$, in analog to the category of contexts (definition 2.25).
(2) A presheaf (definition 3.1) $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$. We refer to $\mathrm{Ty}(\Gamma)$ as the set of *dependent types indexed by* Γ.
(3) A *context extension* operation, constructing $(\Gamma, A) \in \mathcal{C}$ from $A \in \mathrm{Ty}(\Gamma)$.
(4) A *projection morphism* $\pi_A \in \mathcal{C}((\Gamma, A), \Gamma)$, the inverse of context extension.

We also require the object $(\Delta, \sigma^* A) \in \mathcal{C}$ (where $\sigma^* A \in \mathrm{Ty}(\Delta)$ is called the *pullback of A along Δ*) and the morphism $\sigma^A : \mathcal{C}((\Delta, \sigma^* A), (\Gamma, A))$ to exist (assuming $\Gamma \in \mathcal{C}, \Delta \in \mathcal{C}, A \in \mathrm{Ty}(\Gamma)$) by the following pullback square:

$$\begin{array}{ccc} \Delta, \sigma^* A & \xrightarrow{\sigma^A} & \Gamma, A \\ {\scriptstyle \pi_{\sigma^* A}}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_A} \\ \Delta & \xrightarrow[\sigma]{} & \Gamma \end{array}$$

The following *strictness condition* must hold in the pullback square:

$$\mathrm{id}_\Gamma^* A = A \qquad \mathrm{id}_\Gamma^A = \mathrm{id}_{\Gamma, A}$$
$$\gamma^*(\sigma^* A) = (\sigma \circ \gamma)^* A \qquad \sigma^A \circ \gamma^{\sigma^* A} = (\sigma \circ \gamma)^A$$

This definition is due to [Pit01, definition 6.3.3].

Remark 5.7. The "projection morphism" in definition 5.6 is similar to the "display maps" in definition 3.20, and the pullback square in definition 5.6 is similar to the pullback square in remark 2.30.

Lemma 5.8. *For $\Gamma \in \mathcal{C}$, the set $\mathrm{Ty}(\Gamma)$ has an embedding in the objects in the overcategory (definition 3.5) $\mathcal{C}_{/\Gamma}$.*

Proof. By mapping $A \in \mathrm{Ty}(\Gamma)$ to $\pi_A \in \mathcal{C}((\Gamma, A), \Gamma)$. □

Remark 5.9. By lemma 5.8, we may add morphisms between the elements of $\mathrm{Ty}(\Gamma)$ to form a category in order to put more categorical structures into it. This will make Ty into a functor $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$, similar to the functor $\Psi : \mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}$ (lemma 3.40) in a split (definition 3.41 and terminology 3.44) comprehension category (since we have the strictness conditions in definition 5.6).

Notation 5.10. We define two syntactical shorthands for type categories (definition 5.6) for convenience.

- For $A \in \mathrm{Ty}(1)$, we write $A \in \mathcal{C}$ for $(1, A) \in \mathcal{C}$.
- $\mathrm{Ty}_\Gamma(A, B)$ for the hom set $\mathrm{Ty}(\Gamma)(A, B)$.

Definition 5.11 (TyMor)**.** The hom set $\mathrm{Ty}_\Gamma(A, B)$ is defined as the hom set $\mathcal{C}_{/\Gamma}(\pi_A, \pi_B)$. Identities and compositions are inherited from $\mathcal{C}$. This realizes remark 5.9.

Lemma 5.12. *$Ty(\Gamma)$ is a full subcategory (definitions 3.31 and 3.32) of $\mathcal{C}_{/\Gamma}$ if for every $A \neq B \in Ty(\Gamma)$, $(\Gamma, A) \neq (\Gamma, B) \in \mathcal{C}$.*

Proof. By definition 5.11. □

Theorema 5.13. *A discrete (definition 5.3) CwA (definition 3.45) is equivalent to a type category (definition 5.6).*

Proof. We try to equate a CwA with a type category by identifying:

(1) The category interpreting contexts $\mathcal{C}$. Both of them have such categories.
(2) The category of Γ-indexed dependent types for $\Gamma \in \mathcal{C}$. This is called $\Psi(\Gamma)$ in a CwA and $\mathrm{Ty}(\Gamma)$ in a type category. $\mathrm{Ty}(\Gamma)$ is a set, while $\Psi(\Gamma)$ is also a set (since being discrete).
(3) The context comprehension operation. We have this operation in the definition of a type category, and we have constructed the same operation in a CwA in definition 3.20.
(4) See lemma 5.14.

□

Lemma 5.14. *The context comprehension in a CwA satisfies the pullback square in definition 5.6.*

Proof. We define the following category Σ_Ψ (recall lemma 3.40) on a CwA (definition 3.45), which is a construction due to Pitts [Pit01, example 6.3.9] and Grothendieck:

- Objects are pairs (Γ, A) for $\Gamma \in \mathcal{C}$ and $A \in \Psi(\Gamma)$ a category.
- Morphisms are pairs $(\sigma, f) \in \Sigma_\Psi((\Gamma, A), (\Delta, B))$ for $\sigma \in \mathcal{C}(\Gamma, \Delta)$ and $f : A \to \Psi(\sigma)(B)$ (note $\Psi(\sigma)(B) = \sigma^*(B)$) a functor whose domain and codomain are both in $\Psi(\Gamma)$.

We construct the following pullback square:

$$\begin{array}{ccc} (\Gamma, A) & \xrightarrow{(\sigma, f)} & (\Delta, B) \\ {\scriptstyle \pi_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi_B} \\ \Gamma & \xrightarrow[\sigma]{} & \Delta \end{array}$$

The (fully faithful) functorality (definition 3.41) of Ψ says that f is an equivalence and the composition of substitution objects σ is therefore strictly associative. This satisfies the strictness conditions required by the square in definition 5.6. □

Exercise 5.15. Define the composition and identities of Σ_Ψ in lemma 5.14.

Remark 5.16. The original definition of a CwA *is* discrete (and is almost the same as a type category), which means that the morphisms in the fibers in $\mathcal{E}$ are all identity morphisms. In definition 3.45, on the other hand, we allow arbitrary morphisms in $\mathcal{E}$ as long as $\mathcal{F}$ preserves cartesianness and p is a full split fibration.

The benefit of defining a CwA based on a comprehension category is that comprehension categories are appreciated by mathematicians since they have their own well-established fibrations. Curious readers may refer to [Jac93, example 4.10]. ✎

5.2. **Categories with families.**

Definition 5.17 (Dybjer's CwF)**.** A *category with families* (CwF) [CCD19; Dyb96; Hof97] is a structure consisting of:

(1) A category interpreting contexts (definition 2.25) $\mathcal{C}$.
(2) A **Fam**-valued presheaf $T : \mathcal{C}^{\mathrm{op}} \to \mathbf{Fam}$ (definitions 3.1 and 5.5).
(3) A context comprehension operation defined in definition 5.21. It is similar to definitions 3.20 and 5.6.

Remark 5.18. We can also have a contextual (definition 3.47) CwF by requiring an ℓ operation on its $\mathcal{C}$ category, or a democratic (definition 3.50) CwF by requiring such types to exists.

Notation 5.19. We have the following syntactical shorthands:

(1) $\mathrm{Ty}(\Gamma) = X$ for $\Gamma \in \mathcal{C}$ if $T(\Gamma) = (U_x)_{x \in X}$. In fact, Ty is a presheaf.
(2) $\mathrm{Tm}(\Gamma, A) = U_A$ for $A \in \mathrm{Ty}(\Gamma)$ if $T(\Gamma) = (U_x)_{x \in X}$.
(3) For a substitution object $\sigma \in \mathcal{C}(\Gamma, \Delta)$, there exist the following function: $T(\sigma) : (\mathrm{Tm}(\Delta, A))_{A \in \mathrm{Ty}(\Delta)} \to (\mathrm{Tm}(\Gamma, B))_{B \in \mathrm{Ty}(\Gamma)}$.

Definition 5.20 (Substitution)**.** The morphism $T(\sigma)$ in notation 5.19 consists of two maps:

(1) A reindexing map $\sigma^* : \mathrm{Ty}(\Delta) \to \mathrm{Ty}(\Gamma)$ called the *substitution on types*, similar to definition 3.35.
(2) A map $\sigma^* : \mathrm{Tm}(\Delta, A) \to \mathrm{Tm}(\Gamma, \sigma^*(A))$ called the *substitution on terms*.

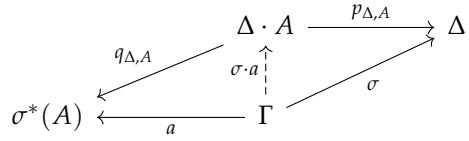
Note that the names of these two maps are overloaded. These maps constitute the following commutative square:

$$\begin{array}{ccc} \Gamma & \mathrm{Ty}(\Gamma) & \mathrm{Tm}(\Gamma, \sigma^*(A)) \\ \downarrow{\scriptstyle \sigma} & \uparrow{\scriptstyle \sigma^*} & \uparrow{\scriptstyle \sigma^*} \\ \Delta & \mathrm{Ty}(\Delta) & \mathrm{Tm}(\Delta, A) \end{array}$$

Beware of the directions of the arrows. A substitution $\sigma \in \mathcal{C}(\Gamma, \Delta)$ brings types formed in Δ to Γ, as discussed in remark 3.37.

Definition 5.21 (Comprehension)**.** For $\Gamma \in \mathcal{C}$ and $A \in \mathrm{Ty}(\Gamma)$, the *context comprehension* of a CwF consists of an assigned context $\Gamma \cdot A$ and two functions $p_{\Gamma,A} : \Gamma \cdot A \to \Gamma$ (which is also a substitution (definition 5.20)) and $q_{\Gamma,A} \in \mathrm{Tm}(\Gamma \cdot A, (p_{\Gamma,A})^*(A))$, such that the following property holds:

For every $\sigma \in \mathcal{C}(\Gamma, \Delta)$, $A \in \mathrm{Ty}(\Delta)$, and $a \in \mathrm{Tm}(\Gamma, \sigma^*(A))$, a substitution object $\sigma \cdot a \in \mathcal{C}(\Gamma, \Delta \cdot A)$ is uniquely identified such that $p_{\Delta,A} \circ (\sigma \cdot a) = \sigma$ and $(\sigma \cdot a)^* q_{\Delta,A} = a$. The characterization is visualized in the following commutative parallelogram:



Theorema 5.22. *A CwF is equivalent to a type category.*

Proof. We try to equate a CwF with a type category by identifying:

(1) The category interpreting contexts $\mathcal{C}$. Both of them have such categories.
(2) The (discrete) category of Γ-indexed dependent types for $\Gamma \in \mathcal{C}$. They are called $\mathrm{Ty}(\Gamma)$ in both categories, both are sets.
(3) The set of terms of type A in a CwF is $\mathrm{Tm}(\Gamma, A)$, while in a type category it is just the set of global sections (terminology 4.7) of $A \in \mathcal{C}$.

□

Exercise 5.23. Prove that the context comprehension operation in a CwF (definition 5.21) and a type category (definition 5.6) are equivalent.

Corollary 5.24. *A CwF is equivalent to a CwA.*

Proof. By theoremata 5.13 and 5.22. □

Remark 5.25. We consider a CwF (definition 5.17) as a CwA (definition 3.45) or a type category (definition 5.6) with additional but unnecessary structures. ✎

Remark 5.26. The structure of a CwF is *very* close to the syntax of type theory – both the formation of contexts and types and the typing of substitutions and terms have a direct correspondence to a "belong to" relation in a CwF. In [CCD19], Castellan even used the notation of substitution from type theory directly ($\_[\gamma]$ instead of γ^*), but we avoid their notation for consistency with the rest of this introduction. ✎

Terminology 5.27. The model of type theory based on a CwF (definition 5.17) is known as the *presheaf* model since the notion of "dependent types defined in a context" is interpreted as a presheaf. ▲

6. Dependent type theory

Definition 6.1 (Monic)**.** We say a morphism $f \in \mathcal{C}(A, B)$ to be a *monomorphism*, a *mono*, or *monic* if for every $C \in \mathcal{C}$ and $g_1, g_2 \in \mathcal{C}(C, A)$ such that:

$$f \circ g_1 = f \circ g_2 \implies g_1 = g_2$$

Holds. A monomorphism is also known as a *left-cancallative* morphism. The above equality is visualized below:

$$C \underset{g_2}{\overset{g_1}{\rightrightarrows}} A \xrightarrow{f} B$$

▲

Demonstration 6.2 (Injection)**.** In **Set**, a morphism is monic (definition 6.1) if and only if it is an injective function. ▲

Definition 6.3 (Monic)**.** This is an alternative to definition 6.1. In a category $\mathcal{C}$, if for every $C \in \mathcal{C}$ the hom functor (definition 3.3) $\mathcal{C}(C, -)$ takes $f \in \mathcal{C}(A, B)$ to injective functions $\mathcal{C}(C, f) : \mathcal{C}(C, A) \to \mathcal{C}(C, B)$, we say f to be *monic*. ▲

Lemma 6.4 (MonoIso)**.** *If for a mono (definitions 6.1 and 6.3) f, there exist a morphism g such that $f \circ g = id_{cod(f)}$, then f is an isomorphism.*

Proof. $f \circ g \circ f = \mathrm{id}_{cod(f)} \circ f = f = f \circ \mathrm{id}_{dom(f)}$ and by the definition of mono and the associativity of composition $g \circ f = \mathrm{id}_{dom(f)}$. Thus f an isomorphism. □

Remark 6.5. There is an intuitive justification of lemma 6.4: in **Set**, invertible and injective (see demonstration 6.2) functions are isomorphisms. ✎

Lemma 6.6 (IsoMono)**.** *If for an isomorphism f, there exist morphisms g_1, g_2 such that $g_1 \circ f = g_2 \circ f$, then $g_1 = g_2$.*

Proof. $g_1 = g_1 \circ f \circ f^{-1} = g_2 \circ f \circ f^{-1} = g_2$. □

6.1. **Equalizers: the extensional equality type.**

Remark 6.7 (McBride)**.** We begin this section with a quotation by Conor McBride from his suspended Twitter account[3]:

> Never trust a type theorist who has not changed their mind about equality.

Definition 6.8 (Id)**.** We extend definition 4.1 with the following type:

$$\frac{\Gamma \vdash \quad \Gamma \vdash A \text{ type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : A}{\Gamma \vdash (\mathrm{Id}_A\ a\ b) \text{ type}} \qquad \frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : \mathrm{Id}_A\ a\ a}$$

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : A \quad \Gamma \vdash p : \mathrm{Id}_A\ a\ b}{\Gamma \vdash a = b : A}$$

This type is known as the *extensional equality type*, in contrast to the *intensional equality type*. We will only talk about the extensional one for convenience. ▲

Terminology 6.9. The introduction rule in definition 6.8 is known as the rule for *reflexivity*, and the elimination rule is known as *equality reflection*. ▲

Definition 6.10 (Equalizer)**.** In a category $\mathcal{C}$ and for $f, g \in \mathcal{C}(X, Y)$ (we refer to pairs of morphisms like this as *parallel morphisms*), there might be some objects $E \in \mathcal{C}$ together with morphisms of form $e \in \mathcal{C}(E, X)$ commuting the following diagram (so that $f \circ e = g \circ e \in \mathcal{C}(E, Y)$):

$$E \xrightarrow{\ e\ } X \underset{g}{\overset{f}{\rightrightarrows}} Y$$

We take the terminal object of the full subcategory (definitions 3.31 and 3.32) of $\mathcal{C}_{/X}$ commuting the above diagram which consists of morphisms like e, and refer to the object E (denoted $\mathrm{Eq}(f, g)$) and the morphism e (denoted $\mathrm{eq}(f, g)$) as the *equalizer* of f and g. We bring these notations into the diagram above:

$$\mathrm{Eq}(f,g) \xrightarrow{\ \mathrm{eq}(f,g)\ } X \underset{g}{\overset{f}{\rightrightarrows}} Y$$

In case each parallel morphisms in $\mathcal{C}$ has an equalizer, we say that $\mathcal{C}$ *has all equalizers*. The definition of equalizers is from [EH63]. ▲

Lemma 6.11. *A category $\mathcal{C}$ has all equalizers (definition 6.10) if it has a terminal object and all pullbacks (definition 2.16).*

[3] https://twitter.com/pigworker

Proof. By lemma 2.19 we know $\mathcal{C}$ has products. We construct the equalizer for arbitrary parallel morphisms $f, g \in \mathcal{C}(X, Y)$:

$$\begin{array}{ccc} \mathrm{Eq}(f,g) & \xrightarrow{\mathrm{eq}(f,g)} & X \\ \downarrow & \lrcorner & \downarrow{\scriptstyle (f,g)} \\ Y & \xrightarrow{(\mathrm{id}_Y, \mathrm{id}_Y)} & Y \times Y \end{array}$$

□

Lemma 6.12 ([EH63, Proposition 1.3])**.** *For a category $\mathcal{C}$ and $f, g \in \mathcal{C}(X, Y)$, $\mathrm{eq}(f, g)$ is monic (definition 6.1).*

Lemma 6.13. *In a category $\mathcal{C}$ and a morphism $f \in \mathcal{C}(X, Y)$, the equalizer of f and f is id_X.*

Definition 6.14. We interpret the extensional identity type in a contextual category (definition 3.48) as an equalizer (definition 6.10). The formation of the *context* Γ *extended by* $\mathrm{Id}_A\ a\ b$ (denoted $(\llbracket\Gamma\rrbracket, \llbracket \mathrm{Id}_A\ a\ b\rrbracket) \in \mathcal{C}$) is by taking the equalizer $\mathrm{Eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)$, where the display map (definition 3.20) is given by the equalizer: $\pi_{\llbracket \mathrm{Id}_A\ a\ b\rrbracket} = \mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)$:

$$\llbracket\Gamma\rrbracket, \llbracket A\rrbracket \overset{\llbracket a\rrbracket}{\underset{\llbracket b\rrbracket}{\leftarrow}} \llbracket\Gamma\rrbracket \xleftarrow{\mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)} \llbracket\Gamma\rrbracket, \llbracket \mathrm{Id}_A\ a\ b\rrbracket$$

By lemma 6.13, $\mathrm{eq}(\llbracket a\rrbracket, \llbracket a\rrbracket) = \mathrm{id}_{\llbracket\Gamma\rrbracket}$. So, we define $\llbracket \mathsf{refl}_a\rrbracket = \mathrm{id}_{\llbracket\Gamma\rrbracket}$:

$$\llbracket\Gamma\rrbracket, \llbracket A\rrbracket \xleftarrow{\llbracket a\rrbracket} \llbracket\Gamma\rrbracket \xrightarrow{\llbracket \mathsf{refl}_a\rrbracket} \llbracket\Gamma\rrbracket, \llbracket \mathrm{Id}_A\ a\ a\rrbracket$$

In case we have a morphism f corresponding to an instance of the identity type, it is an inverse to the equalizer ($\mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket) \circ f = \mathrm{id}_{\llbracket\Gamma\rrbracket, \llbracket \mathrm{Id}_A\ a\ b\rrbracket}$), and by lemma 6.4 f is an isomorphism. Then, by lemma 6.6 $\llbracket a\rrbracket = \llbracket b\rrbracket$ holds, and that makes sense of the elimination rule:

$$\llbracket\Gamma\rrbracket, \llbracket A\rrbracket \overset{\llbracket a\rrbracket}{\underset{\llbracket b\rrbracket}{\xleftarrow{\ \pi_{\llbracket A\rrbracket}\ }}} \llbracket\Gamma\rrbracket \xrightarrow[f]{\mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)} \llbracket\Gamma\rrbracket, \llbracket \mathrm{Id}_A\ a\ b\rrbracket$$

▲

Lemma 6.15 (UIP)**.** *The interpretation in definition 6.14 gives rise to the following "uniqueness of the identity proof" rule:*

$$\frac{\Gamma \vdash A\ type \qquad \Gamma \vdash a : A \qquad \Gamma \vdash p : \mathrm{Id}_A\ a\ a}{\Gamma \vdash p = \mathit{refl}_a : \mathrm{Id}_A\ a\ a}$$

Proof. By lemma 6.13 and the uniqueness of the identity morphism. For visualization, see the second diagram in definition 6.14. □

Theorema 6.16 (Uniqueness)**.** *The interpretation in definition 6.14 gives rise to the following "uniqueness rule" of the identity type:*

$$\frac{\Gamma \vdash A\ type \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : \mathrm{Id}_A\ a\ b \qquad \Gamma \vdash q : \mathrm{Id}_A\ a\ b}{\Gamma \vdash p = q : \mathrm{Id}_A\ a\ b}$$

Proof. (Type theory perspective) By equality reflection, the existence of p and q implies $a = b$, so p and q are also instances of $\mathrm{Id}_A\ a\ a$, and by lemma 6.15 they are both equal to refl_a. □

Proof. (Categorical perspective) $[\![p]\!] \circ \mathrm{eq}([\![a]\!], [\![b]\!]) = [\![q]\!] \circ \mathrm{eq}([\![a]\!], [\![b]\!])$, and then $[\![p]\!] = [\![q]\!]$ since $\mathrm{eq}([\![a]\!], [\![b]\!])$ is monic. □

6.2. **Locally cartesian closed categories: the dependent product type.**

Definition 6.17 (Pi)**.** We extend definition 4.1 with the dependent product type, defined by the following typing rules:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma, A \vdash B \text{ type}}{\Gamma \vdash (\Pi_A B) \text{ type}}$$

$$\frac{\Gamma \vdash (\Pi_A B) \text{ type} \qquad \Gamma, A \vdash B \text{ type} \qquad \Gamma, A \vdash u : B}{\Gamma \vdash \Lambda.u : \Pi_A B}$$

$$\frac{\Gamma \vdash (\Pi_A B) \text{ type} \qquad \Gamma \vdash A \text{ type} \qquad \Gamma, A \vdash B \text{ type} \qquad \Gamma \vdash u : \Pi_A B \qquad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{ap}(u, v) : Bv}$$

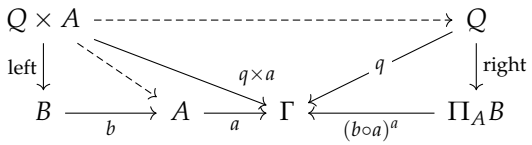
$$\frac{\Gamma \vdash (\Pi_A B) \text{ type} \qquad \Gamma \vdash A \text{ type} \qquad \Gamma, A \vdash B \text{ type} \qquad \Gamma, A \vdash u : B \qquad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{ap}(\Lambda.u, v) = uv : Bv}$$

▲

Definition 6.18 (DepProd)**.** Consider an LCCC (definition 4.34) $\mathcal{C}$, $\Gamma, A, B \in \mathcal{C}$, $a \in \mathcal{C}(A, \Gamma)$, $b\mathcal{C}(B, A)$. Observe that $b \circ a \in \mathcal{C}(B, \Gamma)$.

Then, since $\mathcal{C}$ is an LCCC, $\mathcal{C}_{/\Gamma}$ is a CCC (definition 4.31), so we can take the exponential object of a and $b \circ a$, denoted $(b \circ a)^a \in \mathcal{C}_{/\Gamma}$. We denote the fiber exponent (definition 4.37) as $\Pi_A B = dom((b \circ a)^a) \in \mathcal{C}$, and for every $Q \in \mathcal{C}, q \in \mathcal{C}(Q, \Gamma)$, we have $\mathcal{C}_{/\Gamma}(q \times a, b \circ a) \simeq \mathcal{C}_{/\Gamma}(q, (b \circ a)^a)$.

By the definition of overcategories (definition 3.5), the above isomorphism is also an isomorphism in $\mathcal{C}$, written as $\mathcal{C}(Q \times A, B) \simeq \mathcal{C}(Q, \Pi_A B)$. Diagrammatically, the morphisms "left" and "right" below are in isomorphic hom sets:

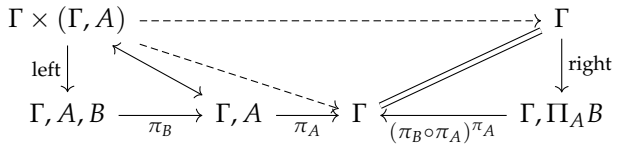


We refer to $\Pi_A B$ as a *dependent product*. ☆

Notation 6.19. In case we are working in a contextual category (definition 3.48) and A, B, Q corresponds to the extended contexts $(\Gamma, A'), (\Gamma, B'), (\Gamma, Q') \in \mathcal{C}$ and a, b, q corresponds to the display maps (definition 3.20) $\pi_{A'}, \pi_{B'}, \pi_{Q'}$, we write $\Pi_{A'} B'$ for $\Pi_A B$ and refer to $(\pi_{B'} \circ \pi_{A'})^{\pi_{A'}}$ as $\pi_{\Pi_{A'} B'}$ ▲

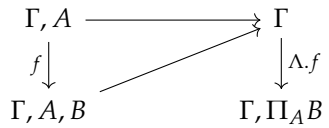
Lemma 6.20. *In a CwA (definition 3.45) $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ where $\mathcal{C}$ is an LCCC (definition 4.34), for $\Gamma \in \mathcal{C}$, $A \in \mathcal{E}_\Gamma$, and $B \in \mathcal{E}_{\Gamma, A}$, there is an isomorphism $\mathcal{C}((\Gamma, A), (\Gamma, A, B)) \simeq \mathcal{C}(\Gamma, (\Gamma, \Pi_A B))$.*

Proof. By specializing the object Q to Γ, the morphisms a, b, q to π_A, π_B, π_Q in the diagram in definition 6.18:



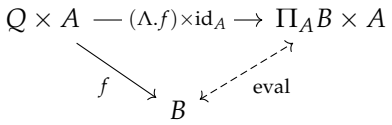
Now, we have an isomorphism $\mathcal{C}(\Gamma \times (\Gamma, A), (\Gamma, A, B)) \simeq \mathcal{C}(\Gamma, (\Gamma, \Pi_A B))$. By lemma 4.33, we can replace $\Gamma \times (\Gamma, A)$ with (Γ, A). □

Notation 6.21. In an LCCC $\mathcal{C}$, by the equivalence in lemma 6.20, for a morphism $f \in \mathcal{C}((\Gamma, A), (\Gamma, A, B))$ we can uniquely obtain a morphism $\Lambda.f \in \mathcal{C}(\Gamma, (\Gamma, \Pi_A B))$. The diagram in the proof of lemma 6.20 can be simplified as:



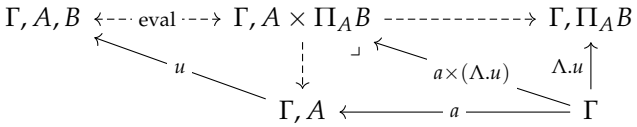
Lemma 6.22 (EvalMap). *In an LCCC $\mathcal{C}$ and for $Q, A, B \in \mathcal{C}$, $f \in \mathcal{C}(Q \times A, B)$, there is a unique isomorphism $\Pi_A B \times A \simeq B$.*

Proof. The following diagram is uniquely determined by the choice of f (using notation 6.21):



□

Definition 6.23. We interpret the dependent product type (definition 6.17) in a contextual category $\mathcal{F} : \mathcal{E} \to \mathcal{C}^{\to}$ where $\mathcal{C}$ is an LCCC (definition 4.34). The introduction and elimination rules are just morphism compositions in the following diagram, omitting the semantic brackets since we are not distinguishing the interpretations and their type theoretical counterparts:



The input of the introduction rule is u, and we take it to $\Lambda.u \in \mathcal{C}(\Gamma, (\Gamma, \Pi_A B))$. In the elimination rule, the input is the product $a \times (\Lambda.u)$, and the commutativity of the diagram justifies the β-equality of the dependent product type.

6.3. **Subobject classifiers: the universe of all propositions.**

Definition 6.24 (Prop). We extend definition 4.1 with a type Prop, defined by the following typing rules:

$$\frac{}{\vdash \mathrm{Prop}\ \mathrm{type}} \qquad \frac{\Gamma \vdash p : \mathrm{Prop}}{\Gamma \vdash \mathrm{El}(p)\ \mathrm{type}} \qquad \frac{\Gamma \vdash p : \mathrm{Prop} \qquad \Gamma \vdash u : \mathrm{El}(p) \qquad \Gamma \vdash v : \mathrm{El}(p)}{\Gamma \vdash u = v : \mathrm{El}(p)}$$

$$\frac{\Gamma \vdash A\ \mathrm{type} \qquad \Gamma \vdash p : \Pi_{x:A}\Pi_{y:A}\mathrm{Id}_A\ x\ y}{\Gamma \vdash R(A,p) : \mathrm{Prop}}$$

We can think of R as an injective function that maps the unique instance (when exists) of a type into an instance of Prop. ⛰

Demonstration 6.25 (Truth)**.** The unit type (definition 4.11), $\top$, has a corresponding proposition (definition 6.24) $R(\top, \Lambda.\Lambda.\mathsf{refl}_\star) : \mathrm{Prop}$. ⛰

Terminology 6.26. When talking about the instances of a type, we say that the type is *inhabited* if it has at least one instance, and these instances are called *inhabitants*.

If the type has no instance, we say it is *uninhabited*. ⛰

Remark 6.27. The definition 6.24 is an imitation of the Prop universe in the (extended) calculus of constructions [Coq85; Luo90; Luo94], which can be regarded as the type of all types (called *propositions*) that "have either one single inhabitant (terminology 6.26) or have no instance at all". The second last rule in definition 6.24 captures such *uniquely inhabited* property of propositions.

By having a universe for *some* types (propositions), we partially addresses warning 4.3. ✎

Definition 6.28 (SubObj)**.** For a category $\mathcal{C}$ with a terminal object (definition 2.9) 1, all pullbacks (definition 2.16), a selected object $\Omega \in \mathcal{C}$ called the *truth value object*, and a mono (definition 6.1) $\mathsf{true} \in \mathcal{C}(1, \Omega)$, called the *subobject classifier*, such that:

Every mono $\pi_U \in \mathcal{C}(U, \Gamma)$ (for some $U, \Gamma \in \mathcal{C}$) uniquely determines a morphism $\chi_U \in \mathcal{C}(\Gamma, \Omega)$ such that the following square is a pullback square:

$$\begin{array}{ccc} U & \xrightarrow{\pi_U} & \Gamma \\ {\scriptstyle 1}\downarrow & \lrcorner & \downarrow{\scriptstyle \chi_U} \\ 1 & \xrightarrow{\mathsf{true}} & \Omega \end{array}$$

The morphism χ_U is also known as the *characteristic map* or *classifying map* of the *subobject* π_U. ⛰

Demonstration 6.29 (Set)**.** In **Set**, the truth value object (definition 6.28) Ω is the 2-valued set $\mathbf{2} := \{t, f\}$, and $\mathsf{true} \in \mathbf{Set}(1, \mathbf{2}), \mathsf{true}(x) := t$. For every $A \in \mathbf{Set}$ and $B \subseteq A \in \mathbf{Set}$ a subobject (that π_B is the natural inclusion (terminology 1.3)), the characteristic map is defined as $\chi_B(a) := \left\{ \begin{array}{ll} t & \text{if } a \in B \\ f & \text{otherwise} \end{array} \right\}$. For each natural inclusion morphism in **Set** can we find a subobject classifier for it. We specialize the diagram in definition 6.28 into **Set**:

$$\begin{array}{ccc} B & \xrightarrow{\subseteq} & A \\ {\scriptstyle 1}\downarrow & \lrcorner & \downarrow{\scriptstyle \chi_B} \\ 1 & \xrightarrow{t} & \{t, f\} \end{array}$$

The pullback square characterizes χ_B to be a function that "returns f as much as possible" because it is terminal in $\mathbf{Set}(B, A)$. ⛰

Definition 6.30. We interpret the universe of propositions (definition 6.24) in a contextual category (definition 3.48) with a subobject classifier (definition 6.28) defined as demonstration 6.25.

Consider a type $\boxed{\Gamma \vdash A \text{ type}}$, it is a proposition when uniquely inhabited (terminology 6.26) by $\Gamma \vdash u : A$ or uninhabited. In the following diagram, *the context* Γ *extended by the proposition* A is the pullback $(\llbracket\Gamma\rrbracket, \llbracket A\rrbracket) \in \mathcal{C}$, and the interpretation of the Prop universe is the truth object:

$$\begin{array}{ccc} \llbracket\Gamma\rrbracket, \llbracket A\rrbracket & \xrightarrow{\pi_{\llbracket A\rrbracket}} & \llbracket\Gamma\rrbracket \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \llbracket R(A,p)\rrbracket} \\ 1 & \xrightarrow{\llbracket R(\top,\Lambda.\Lambda.\star)\rrbracket} & \llbracket\mathrm{Prop}\rrbracket \end{array}$$

(1) Since $\pi_{\llbracket A\rrbracket}$ is mono, its inverse $\llbracket u\rrbracket \in \mathcal{C}(\llbracket\Gamma\rrbracket, (\llbracket\Gamma\rrbracket, \llbracket A\rrbracket))$ (if exists) must be unique. This uniqueness justifies the uniqueness rule of propositions and is guaranteed by the proof p.
(2) For every proposition A, we define a (unique) term $R(A, p) \in \mathrm{Prop}$ that corresponds to A. The uniqueness justifies the extensionality (see theorema 6.31) of Prop and the existence justifies the completeness.
(3) The $\mathrm{El}(-)$ operation takes a morphism $A \in \mathcal{C}(\llbracket\Gamma\rrbracket, \llbracket\mathrm{Prop}\rrbracket)$ (which is an instance of Prop) on the left of the above diagram and returns the pullback. To some extent, $\mathrm{El}(-)$ and $R(-, p)$ are inverse to each other.

▲

Theorema 6.31 (Extensionality)**.** *The extensionality of propositions holds in the interpretation in definition 6.30:*

$$\frac{\begin{array}{cc}\Gamma \vdash A \textit{ type} & \Gamma \vdash p : \Pi_{x:A}\Pi_{y:A}\mathrm{Id}_A\ x\ y \\ \Gamma \vdash B \textit{ type} & \Gamma \vdash q : \Pi_{x:B}\Pi_{y:B}\mathrm{Id}_B\ x\ y\end{array}}{\Gamma \vdash R(A,p) = R(B,q) : \mathrm{Prop}}$$

Theorema 6.32. *For any type* A *and its instances* $x : A, y : A$, $\boxed{\mathrm{Id}_A\ x\ y}$ *is a proposition. In other words, the following type is inhabited:*

$$\Pi_{x:A}\Pi_{y:A}\Pi_{i:\mathrm{Id}_A\ x\ y}\Pi_{j:\mathrm{Id}_A\ x\ y}\mathrm{Id}_{\mathrm{Id}_A\ x\ y}\ i\ j$$

Proof. By theorema 6.16 we can prove it by reflexivity: $\mathsf{uip} := \Lambda x.\Lambda.\Lambda.\Lambda.\mathsf{refl}_{\mathsf{refl}_x}$ (alternatively we can write $\mathsf{uip} := \Lambda.\Lambda y.\Lambda.\Lambda.\mathsf{refl}_{\mathsf{refl}_y}$). □

6.4. **Internalized constructions.**

Definition 6.33 (Bool)**.** We define a type Bool in the type theory in definition 4.1 with the extensions (definitions 4.11 and 4.19) as:

(1) $\mathsf{Bool} := \top \sqcup \top$.
(2) $\mathsf{true} := \mathsf{inl}(\star)$.
(3) $\mathsf{false} := \mathsf{inr}(\star)$.
(4) $\langle t\ ?\ u : v\rangle := \mathsf{match}(t, u, v)$.

From the above definition, we can derive the following typing rules for Bool:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Bool}\ \mathsf{type}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{true} : \mathsf{Bool} \quad \text{and} \quad \Gamma \vdash \mathsf{false} : \mathsf{Bool}}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A \qquad \Gamma \vdash t : \mathsf{Bool}}{\Gamma \vdash \langle t\ ?\ u : v \rangle : A}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A \qquad \Gamma \vdash t : \mathsf{Bool}}{\Gamma \vdash \langle \mathsf{true}\ ?\ u : v \rangle = u : A \quad \text{and} \quad \Gamma \vdash \langle \mathsf{false}\ ?\ u : v \rangle = u : A}$$

▲

Definition 6.34 (PropTrunc)**.** For every well-formed type $\boxed{\Gamma \vdash A\ \mathsf{type}}$, we define a type $\boxed{\Gamma \vdash ||A||\ \mathsf{type}}$, called the *propositional truncation of* A:

(1) $||A|| := \prod_{P:\mathrm{Prop}} (A \to \mathrm{El}(P)) \to \mathrm{El}(P)$ (using definitions 4.38, 6.17 and 6.24).
(2) For $\Gamma \vdash u : A$, $\Gamma \vdash |u| : ||A||$ (unfolds to $\Gamma \vdash |u| : \prod_{P:\mathrm{Prop}} (A \to \mathrm{El}(P)) \to \mathrm{El}(P)$), defined as $|u| := \Lambda P.\lambda f.\mathsf{ap}(f, u)$.
(3) The elimination, that for $\Gamma \vdash u : ||A||$ we can eliminate it to another proposition $\Gamma \vdash P : \mathrm{Prop}$ with the map $\Gamma \vdash f : A \to \mathrm{El}(P)$ by $\mathsf{ap}(\mathsf{ap}(u, P), f)$.

From the above definition, we can derive the following typing rules for $|| - ||$:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A\ \mathsf{type}}{\Gamma \vdash ||A||\ \mathsf{type}} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash u : A}{\Gamma \vdash |u| : ||A||}$$

$$\frac{\begin{array}{c}\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash ||A||\ \mathsf{type} \\ \Gamma \vdash P : \mathrm{Prop} \qquad \Gamma \vdash f : A \to \mathrm{El}(P) \qquad \Gamma \vdash u : ||A||\end{array}}{\Gamma \vdash ||\_||\text{-}\mathsf{elim}(u, P, f) : \mathrm{El}(P)}$$

▲

Definition 6.35. We construct an alternative definition of Id that lives in Prop (see theorema 6.32) by $\mathrm{Id}'_A\ a\ b := R(\mathrm{Id}_A\ a\ b, \mathsf{ap}(\mathsf{ap}(\mathsf{uip}, a), b))$. We derive the following judgment:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash \mathrm{Id}'_A\ a\ b : \mathrm{Prop}}$$

▲

Definition 6.36 (Void)**.** We define the empty type $\bot := \prod_{P:\mathrm{Prop}} P$ using definitions 6.17 and 6.24. ▲

7. More category theory

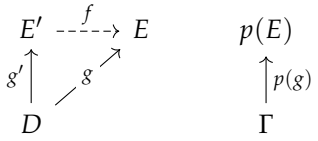
Definition 7.1 (Pushout)**.** The notion dual to pullback (definition 2.16) is *pushout* (or *pushforward*), characterizing objects by taking pullbacks in the opposite category (definition 2.20), denoted and diagramed below:

$$\begin{array}{ccc} C & \longrightarrow & B \\ \downarrow & \ulcorner & \downarrow \\ A & \longrightarrow & A \sqcup_C B \end{array}$$

The pushout $(A \sqcup_C B) \in \mathcal{C}$ is also known as the *fiber coproduct* of A and B. ▲

Definition 7.2 (Undercategory)**.** For a category $\mathcal{C}$ and $X \in \mathcal{C}$, the *undercategory* ${}_{/X}\mathcal{C}$ (also known as the *coslice category under* X) is the overcategory $\mathcal{C}^{\mathrm{op}}{}_{/X}$. The objects of an undercategory are certain morphisms *from* X in $\mathcal{C}$ commuting the (reversed version of the) diagrams in definition 3.5. ▲

Definition 7.3 (Cocartesianness)**.** Similar to definition 3.11, where we have a functor $p : \mathcal{E} \to \mathcal{C}$ and objects $\Gamma \in \mathcal{C}$, $E, E', D \in \mathcal{E}$. We say a morphism $g \in \mathcal{E}(D, E)$ to be *Grothendieck cocartesian* if for all $g' \in \mathcal{E}(D, E')$ there is a unique morphism $f \in \mathcal{E}(E', E)$ such that $g' \circ f = g$. This universal property is visualized below:



The diagram shows why g is sometimes known as an *initial lifting*. This definition is due to Jacobs [Jac93, definition 2.1 (i)] due to Grothendieck [Gro71]. ▲

Definition 7.4 (Opfibration)**.** Similar to definition 3.13, we define a functor $p : \mathcal{E} \to \mathcal{C}$ to be an *opfibration* if composition preserves Grothendieck cocartesianness and for every $\sigma \in \mathcal{C}(p(D), \Gamma)$ there is a Grothendieck cocartesian morphism $g \in \mathcal{E}(D, E)$ such that $p(g) = \sigma$. The visualization is almost identical to the one in definition 3.13. ▲

Corollary 7.5. *$p : \mathcal{E} \to \mathcal{C}$ is an opfibration $\iff$ $p^{\mathrm{op}} : \mathcal{E}^{\mathrm{op}} \to \mathcal{C}^{\mathrm{op}}$ is a fibration.*

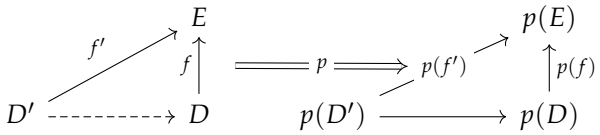
Definition 7.6 (Bifibration)**.** A functor that is a fibration (definition 3.13) and a opfibration (definition 7.4) is called a *bifibration*. ▲

Demonstration 7.7. Extending demonstrations 3.16 and 3.17:

- The domain projection fibration $dom : \mathcal{C}^{\to} \to \mathcal{C}$ is a bifibration if it has all pushouts (definition 7.1).
- The codomain projection functor $dom : \mathcal{C}^{\to} \to \mathcal{C}$ is an opfibration (definition 7.4).

We can also replace the word "fibration" with "bifibration" in demonstration 3.16. ▲

Definition 7.8 (Cartesianness)**.** For a functor $p : \mathcal{E} \to \mathcal{C}$ and $f \in \mathcal{E}(D, E)$, we say that f is (strongly, see history 3.12) *p-cartesian* if for every $f' \in \mathcal{E}(D', E)$ there is a commutative triangle $p(f) \circ \overline{g} = p(f')$ for some $\overline{g}$ that can uniquely determine a commutative triangle $f \circ g = f'$ for some g. The visualization is:

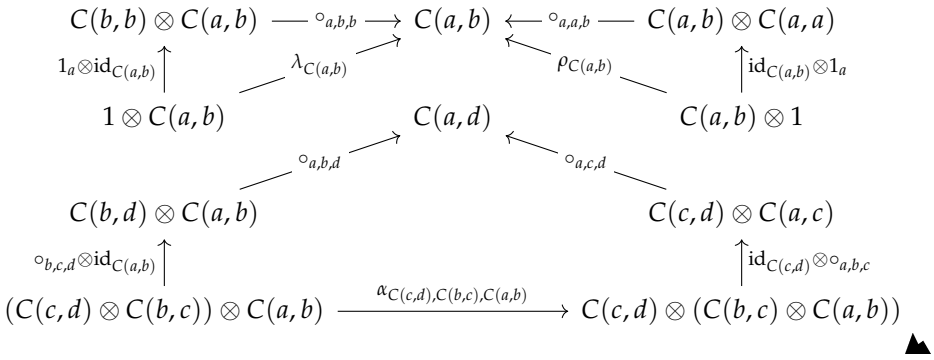


In the diagram, the triangle on the right can uniquely determine (called *lift to*) the one on the left. This definition is taken from [Lur21, §01T1]. ▲

Definition 7.9 (Enrichment)**.** A category C *enriched over* a monoidal (definition 4.24) category K is a category that replaces its hom sets with objects in K (called *hom objects* or *objects of morphisms*). This means we can have morphisms between the hom objects. Formally, C has the following structures:

(1) Objects and morphisms, just like other categories. Morphisms are objects in K, so we will denote objects in K as $C(a,b)$ for $a,b \in C$.
(2) For $a,b,c \in C$, there is a morphism $\circ_{a,b,c} \in K(C(a,b) \otimes C(b,c), C(a,c))$, called the *composition morphism*.
(3) For the tensor unit $1 \in K$ and $a \in C$, there is an *identity morphism* $1_a \in K(1, C(a,a))$.

Such that the following diagrams commute:



Demonstration 7.10. A category enriched in **Cat** is a strict 2-category (definition 3.23). This is a realization of remark 3.25.

7.1. **Common mistakes and counterexamples.**

Warning 7.11. For a category $\mathcal{C}$ and objects $A, B \in \mathcal{C}$, we take the full subcategory (definitions 3.31 and 3.32) of $\mathcal{C}$ by selecting objects $X \in \mathcal{C}$ such that both $\mathcal{C}(X,A)$ and $\mathcal{C}(X,B)$ are nonempty.

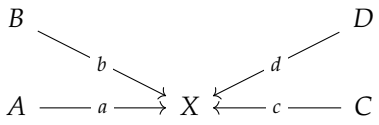
The terminal object in such full subcategory is *not* the product object $A \times B \in \mathcal{C}$. The reason is that the terminal object in such subcategory *does not commute* (although the morphisms exist) the diagram in definition 2.11.

Demonstration 7.12. In **Set**, for sufficiently large sets (say, size larger than 2) $A, B \in$ **Set**, the full subcategory of **Set** like in warning 7.11 is equivalent to **Set**, whose terminal object is the trivial set $\{\emptyset\}$. This is definitely not the product $A \times B$.

Remark 7.13. Similar to warning 7.11, pullbacks cannot be characterized as a terminal object in a subcategory either.

Warning 7.14. Coproduct types are *not* interpreted as pushouts (definition 7.1) which are coproducts in the undercategory (definition 7.2). Instead, they correspond to coproduct objects in the overcategory (definition 3.5).

Warning 7.15. We *cannot* turn isomorphisms of hom sets in the overcategory into the original category. In other words, given a category $\mathcal{C}$, objects $A, B, C, D, X \in \mathcal{C}$, and the following morphisms:



There is $\mathcal{C}_{/X}(a,b) \simeq \mathcal{C}_{/X}(c,d) \implies \mathcal{C}(A,B) \simeq \mathcal{C}(C,D)$ because $\mathcal{C}(A,B)$ may contain more morphisms than $\mathcal{C}_{/X}(a,b)$, and similarly for $\mathcal{C}(C,D)$ and $\mathcal{C}_{/X}(c,d)$. ☠

Demonstration 7.16. In definition 4.37, even $\mathcal{C}_{/\Gamma}(a,b) \simeq \mathcal{C}_{/\Gamma}(1_{\mathcal{C}_{/\Gamma}}, b^a)$, we cannot conclude $\mathcal{C}(B,A) \simeq \mathcal{C}(A \to_\Gamma B, \Gamma)$. ▲

Warning 7.17. Speaking of internalized definitions, we *cannot yet* define types by large elimination because there is not a type for all types (warning 4.3) (although we have a type for all propositions). ☠

Warning 7.18. For a category $\mathcal{C}$, its terminal object $1 \in \mathcal{C}$, an arbitrary object $A \in \mathcal{C}$, and a morphism $f \in \mathcal{C}(1,A)$, f is *not* the terminal object in the overcategory $\mathcal{C}_{/X}$.

Terminal objects in overcategories are identity morphisms. ☠

ACKNOWLEDGEMENT

We are grateful to Yuchen Wu, Niels van der Weide, Ende Jin, Anqur Lu, Jonathan Sterling, Parker Liu, and GitHub users @dramforever, @AliceLogos, and @Guest0x0 for their useful suggestions on improving this introduction. We thank @AliceLogos for carefully checking the definitions in the draft version of this introduction and Valery Isaev for sharing ideas on categorical semantics of dependent type theories.

The development of this introduction is based on the following artifacts:

- The LaTeX system, its TeXLive distribution[4], the BibTeX tool, and the mathpazo[5] font.
- The "quiver" tool[6], an interactive commutative-diagramming tool.
- "Detexify"[7], a tool that suggests LaTeX commands from a drawing.
- "BibTeX Tidy"[8], a smart formatter for BibTeX entries.
- The Emacs operating system and the AUCTeX[9] major-mode of it.
- The "ncatlab" website[10], the "kerodon" website [Lur21], the TeX Q&A website[11] and the Wikipedia.
- Several social networks connecting researchers together.

REFERENCES

[Ste20] Jonathan Sterling. *An OK Version of Type Theory*. 2020. URL: https://www.jonmsterling.com/pdfs/ok-type-theory.pdf.

[CA18] Jesper Cockx and Andreas Abel. "Elaborating Dependent (Co)Pattern Matching". In: *Proc. ACM Program. Lang.* 2.ICFP (July 2018). DOI: 10.1145/3236770.

[CFC59] Haskell B. Curry, Robert Feys, and William Craig. "Combinatory Logic, Volume I". In: *Philosophical Review* 68.4 (1959), pp. 548–550. DOI: 10.2307/2182503.

[4] https://www.tug.org/texlive

[5] https://www.ctan.org/pkg/mathpazo

[6] https://q.uiver.app

[7] https://detexify.kirelabs.org/classify.html

[8] https://flamingtempura.github.io/bibtex-tidy

[9] https://www.gnu.org/software/auctex

[10] https://ncatlab.org/nlab/show/HomePage

[11] https://tex.stackexchange.com

[de 72] N.G de Bruijn. "Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem". In: *Indagationes Mathematicae (Proceedings)* 75.5 (1972), pp. 381–392. ISSN: 1385-7258. DOI: 10.1016/1385-7258(72)90034-0. URL: https://www.sciencedirect.com/science/article/pii/1385725872900340.

[Sco] Luis Scoccola. *Models of Intensional Dependent Type Theory*. URL: http://luisscoccola.github.io/Luis%20Scoccola_files/dttmodels.pdf.

[Pit01] Andrew M. Pitts. "Categorical Logic". In: *Handbook of Logic in Computer Science: Volume 5: Logic and Algebraic Methods*. USA: Oxford University Press, Inc., 2001, pp. 39–123. ISBN: 0198537816.

[Jac93] Bart Jacobs. "Comprehension categories and the semantics of type dependency". In: *Theoretical Computer Science* 107.2 (1993), pp. 169–207. ISSN: 0304-3975. DOI: 10.1016/0304-3975(93)90169-T. URL: https://www.sciencedirect.com/science/article/pii/030439759390169T.

[Gro71] Alexander Grothendieck. "Categories Fibrees et Descente". In: *Revêtements Etales et Groupe Fondamental*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1971, pp. 145–194. ISBN: 978-3-540-36910-3.

[Cur90] P.-L. Curien. "Substitution up to isomorphism". en. In: *Diagrammes* 23 (1990), pp. 43–66. URL: http://www.numdam.org/item/DIA_1990__23__43_0/.

[CCD19] Simon Castellan, Pierre Clairambault, and Peter Dybjer. "Categories with Families: Unityped, Simply Typed, and Dependently Typed". In: *CoRR* abs/1904.00827 (2019). arXiv: 1904.00827.

[Car86] John Cartmell. "Generalised algebraic theories and contextual categories". In: *Annals of Pure and Applied Logic* 32 (1986), pp. 209–243. ISSN: 0168-0072. DOI: 10.1016/0168-0072(86)90053-9. URL: https://www.sciencedirect.com/science/article/pii/0168007286900539.

[Mar75] Per Martin-Löf. "An intuitionistic theory of types: predicative part". In: *Logic Colloquium '73, Proceedings of the Logic Colloquium*. Ed. by H.E. Rose and J.C. Shepherdson. Vol. 80. Studies in Logic and the Foundations of Mathematics. North-Holland, 1975, pp. 73–118.

[Coq85] Thierry Coquand. "Une théorie des constructions". PhD thesis. L'universite Paris VII, 1985.

[Luo90] Zhaohui Luo. "An Extended Calculus of Constructions". PhD thesis. University of Edinburgh, 1990.

[Luo94] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. USA: Oxford University Press, Inc., 1994. ISBN: 0198538359.

[Gir72] Jean-Yves Girard. "Interpretation fonctionelle et elimination des coupure dans l'arithmetic d'ordre superieur". PhD thesis. L'universite Paris VII, 1972.

[Hur95] Antonius J. C. Hurkens. "A simplification of Girard's paradox". In: *Typed Lambda Calculi and Applications*. Ed. by Mariangiola Dezani-Ciancaglini and Gordon Plotkin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 266–278. ISBN: 978-3-540-49178-1.

[See84] Robert AG Seely. "Locally cartesian closed categories and type theory". In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 95. 1. Cambridge University Press. 1984, pp. 33–48.

[Hof94] Martin Hofmann. "On the Interpretation of Type Theory in Locally Cartesian Closed Categories". In: *Selected Papers from the 8th International Workshop on Computer Science Logic*. CSL '94. Berlin, Heidelberg: Springer-Verlag, 1994, pp. 427–441. ISBN: 3540600175. DOI: 10.5555/647844.736579.

[Dyb96] Peter Dybjer. "Internal type theory". In: *Types for Proofs and Programs*. Ed. by Stefano Berardi and Mario Coppo. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 120–134. ISBN: 978-3-540-70722-6.

[Hof97] Martin Hofmann. "Syntax and semantics of dependent types". In: *Extensional Constructs in Intensional Type Theory*. London: Springer London, 1997, pp. 13–54. ISBN: 978-1-4471-0963-1. DOI: 10.1007/978-1-4471-0963-1_2.

[EH63] Beno Eckmann and P.J. Hilton. "Group-like structures in general categories II equalizers, limits, lengths". In: *Mathematische Annalen* 151.2 (1963), pp. 150–186. ISSN: 1432-1807. DOI: 10.1007/BF01344176.

[Lur21] Jacob Lurie. *Kerodon*. https://kerodon.net. 2021.

THE PENNSYLVANIA STATE UNIVERSITY
Email address: yqz5714@psu.edu
URL: https://personal.psu.edu/yqz5714