String and Box Diagrams for Topoi

William James Angus

A dissertation submitted in Michaelmas 2024 for the degree of

MSc Advanced Computer Science

at The University of Oxford

Supervised by Vincent Wang-Maścianica.

Abstract

String diagrams were developed to aid reasoning in certain categories, most notably monoidal categories. Extending basic string diagrams with copy maps allows reasoning in categories with finite products. Further adding functor boxes, allows reasoning in finitely complete categories, categories with sub-object classifiers, and cartesian closed categories. Combining these allows reasoning in arbitrary (elementary) topoi. This in turn allows for a proof of the Fundamental Theorem of Topos Theory purely using string diagrams, as well as developing a string diagrammatic account of the internal logic of a topos (which coincides with higher-order intuitionistic logic).

Contents

Chapter 0

Introduction

This thesis presents an introduction to topoi and their internal logic via the use of string diagrams and functor boxes.

String diagrams were introduced in [JS88], which demonstrates the equivalence between the morphisms in monoidal categories and diagrams existing in the plane – string diagrams. This has been extended to different kinds of categories; for a survey of such, see [Sel11]. It is common for these string diagrams to be used to aid reasoning in these categories, most notable is in the case of Quantum Computation being taught to High School students [D-CYP+23]. It is my hope that this thesis can contribute to this tradition with the addition of string diagrams for topoi. I will combine the pure string diagrammatic approach with functor boxes, as introduced in [Mel06].

A topos (or, in particular, an elementary topos, which is what this thesis deals with) is a category which is in some sense a generalisation of **Set**. That is, it allows for set-like reasoning within it. We shall see this in the form of higher-order intuitionistic logic, which is, at least in some senses, a slight weakening of typical set theory with classical first-order logic. Topoi were originally introduced by Grothendieck in the 1940s, for the purpose of studying sheaves on a space. However, nowadays there is a plethora of research into topoi, going in many directions (see e.g., the as of yet incomplete [Joh02]). Here, however, we are most concerned with its internal logic, through the lens of categorical logic.

Categorical logic is the field of interpreting logic within categories. Different kinds of categories have different kinds of logics. For example, cartesian closed categories have an internal logic equivalent to the typed lambda calculus. In the case of topoi, the internal logic is that of higher-order intuitionistic logic.

Categorical logic is used throughout computer science. For example, there are applications in the design of programming languages (such as Haskell or Lisp), compiler optimisation, and interactive theorem proving (such as in HOL or Isabelle).

The only pre-requisite if that of basic category theory: in particular, the notions of category, functor, natural transformation, and limit. Any basic course on category theory should cover these topics, but for a full introduction to the pre-requisite material see [Rie17] (first three chapters) or [Mac71] (up to, and including, chapter V). It would also be helpful for the reader to have encountered some type theory and logic before, as well as categorical string diagrams in any capacity.

The structure of this thesis is as follows.

- **Chapter 1** introduces monoidal categories and basic string diagrams and functor boxes.

- **Chapter 2** shows how we can reason string-diagrammatically, by extending the results of the previous chapter, in categories with finite products, finitely complete categories, cartesian closed categories, and categories with sub-object classifiers. Combining these gives a string-diagrammatic syntax for topoi.
- **Chapter 3** uses these string diagrams to prove the Fundamental Theorem of Topos Theory. This use of string diagrams, in places, provides a great simplification of the non-string-diagrammatic proofs of the Fundamental Theorem.
- **Chapter 4** introduces categorical logic using the previously developed string diagrams. Then we shall see a development of the key logical operations inside of a topos, using string diagrams, which yields a string-diagrammatic version of higher-order intuitionistic logic.

The main contributions of this thesis are as follows:

- In chapter 2, I introduce a new string-diagrammatic calculus (based on an old calculus which was not proved to be coherent) for left and right closed monoidal categories, and show its coherence.
- In chapter 2, I introduce a new string-diagrammatic calculus for categories with a sub-object classifier.
- The previous two results yield a new string-diagrammatic calculus for topoi.
- In chapter 3, I prove the Fundamental Theorem of Topos Theory by using this calculus, which yields new constructions of the dependent sum and dependent product functors.
- In chapter 4, I prove soundness for the internal type logic of a topos using string diagrams, and I introduce nice novel syntactic sugar for the logical operators, which yields a string-diagrammatic calculus for reasoning on higher-order intuitionistic logic.

Finally, before proceeding, here is a list of notation that I employ:

- $\mathrm{Ob}\,\mathcal{C}$ is the collection of objects of the category $\mathcal{C}$;
- id_A is the identity morphism on A;
- 1 is the terminal object of a category;
- $\pi_i^{A_1 \times A_2}$ is the projection map of $A_1 \times A_2$ onto A_i, as given by the binary product;
- $A \underset{f,g}{\times} B$ is the pullback of $f : A \to X$ and $g : B \to X$; and
- $\mathbb{E}_i^{f_1,f_2}$ is the projection morphism $A_1 \underset{f_1,f_2}{\times} A_2 \to A_i$ of the pullback.

Chapter 1

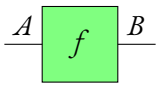
Basic String Diagrams and Functor Boxes

Contents

In this chapter, I will introduce the basic string diagrams and functor boxes, upon which the rest of the thesis is based, we will start with arbitrary string diagrams applicable to any category, then move into those specialised for monoidal categories; and finally I will introduce functor boxes.

1.1 String Diagrams

To begin, let's see some basic string diagrams. These are diagrams which allow us to graphically represent a unique morphism in a given category. The most basic form is given $f : A \to B$ in a category $\mathcal{C}$, we write

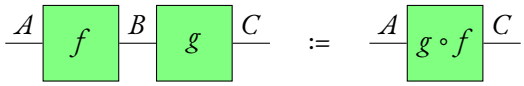


to denote f string diagrammatically.

We can represent composition as follows:

Definition 1.1.1: Graphical Composition

Let $\mathcal{C}$ be a category, and $f : A \to B$ and $g : B \to C$ in $\mathcal{C}$, then

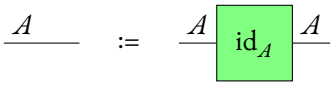


(1.1)

and then identities as just strings:

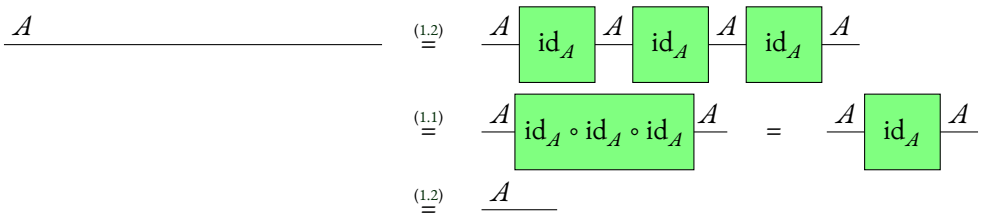
Definition 1.1.2: Graphical Identity

Let $\mathcal{C}$ be a category, and $A \in \mathrm{Ob}\,\mathcal{C}$, then

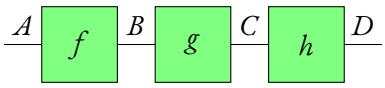


(1.2)

Given these definitions, we can see that the length of a "string" in a string diagram doesn't matter:



We can also see that associativity is baked in to string diagrams, consider the following morphism

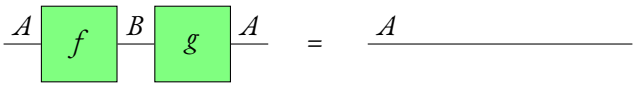


we have no way of telling whether this is the composite $(h \circ g) \circ f$ or $h \circ (g \circ f)$, which is good, because they are identical in any category.

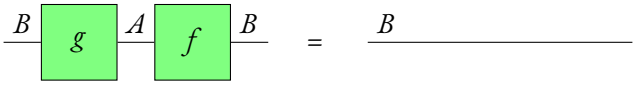
As it stands, these diagrams are not too interesting – we can enrich them by allowing vertical composition of wires and morphisms – which we shall see in the next section. However, first, we can string diagrammatically define the notions of isomorphism and monomorphism:

Definition 1.1.3: Isomorphism

Let $\mathcal{C}$ be a category with a morphism $f : A \to B$. Then we say that f is an *isomorphism* if there exists a morphism $g : B \to A$ in $\mathcal{C}$ such that

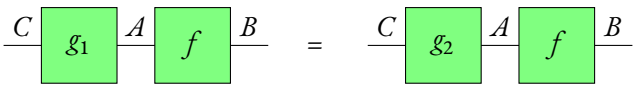


and

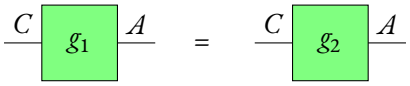


Definition 1.1.4: Monomorphism

Let $\mathcal{C}$ be a category with morphism $f : A \to B$, we say that f is *monic* or that it is a *monomorphism* if whenever



for some g_1 and g_2 in $\mathcal{C}$, then



1.2 Monoidal Categories

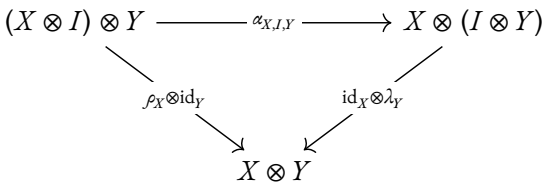
Almost all work done with these kinds of string diagrams take place in monoidal categories. This is because they allow us to greatly enrich the diagrams by allowing vertical composition, rather than the horizontal composition that we have been limited to thus far.

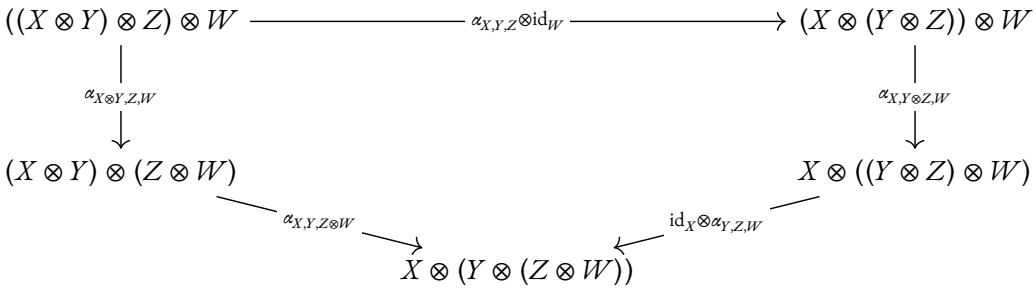
Definition 1.2.1: Monoidal Category

Let $\langle \mathcal{C}, \otimes, I, \alpha, \lambda, \rho \rangle$ be a sextuple consisting of

- a category $\mathcal{C}$;
- a functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, known as the *monoidal product*;
- a distinguished element $I \in \mathcal{C}$, known as the *monoidal unit*;
- a natural isomorphism α with components $\alpha_{X,Y,Z} : (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$, known as the *associator*;
- a natural isomorphism λ with components $\lambda_X : I \otimes X \to X$, known as the *left unitor*; and
- a natural isomorphism ρ with components $\rho_X : X \otimes I \to X$, known as the *right unitor*,

such that the following triangle and pentagon equations





commute for all objects X, Y, Z, W of $\mathcal{C}$, then we say that $\mathcal{C}$ (or more precisely, $\langle \mathcal{C}, \otimes, I, \alpha, \lambda, \rho \rangle$) is a *monoidal category*.

In the special case where α, λ, and ρ are identities, we call it a...

Definition 1.2.2: Strict Monoidal Category

A monoidal category $\langle \mathcal{C}, \otimes, I, \alpha, \lambda, \rho \rangle$ is a *strict monoidal category* if the natural transformations α, λ, and ρ are all the identity natural transformations.

These are the kinds of monoidal categories that we are interested in. I will assume that all monoidal categories are strict in this thesis. This may seem like a problematic approach, however, the following Theorem ensures that this is innocent

Theorem 1.2.3: [Mac63]

Every monoidal category is monoidally equivalent to a strict monoidal category.

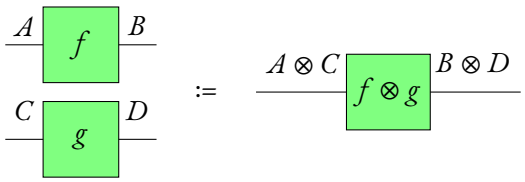
In other words, we may always replace any monoidal category with its strict equivalent, and work within that instead. That is what we do. The reason for this is that strict monoidal categories have a particularly nice string-diagrammatic syntax, which we shall see now.

1.2.1 String Diagrams for (Strict) Monoidal Categories

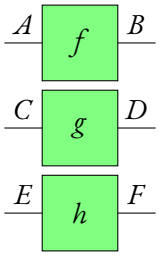
In string diagrams, we represent the monoidal product as stacking diagrams vertically. That is,

Definition 1.2.4

et $\mathcal{C}$ be a (strict) monoidal category with morphisms $f : A \to B$ and $g : C \to D$, we define



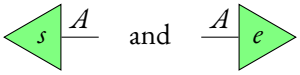
Again, this has associativity baked in (which is why string diagrams naturally work with strict monoidal categories over their non-strict counterparts):



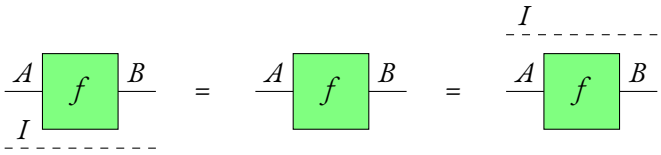
we cannot tell whether this is $(f \otimes g) \otimes h$ or $f \otimes (g \otimes h)$.

We draw the morphism id_I, the identity on the monoidal unit as an empty diagram:

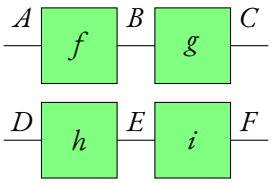
which gives rise to the notion of states and effects, morphisms from I and into I respectively:



Representing id_I as the empty diagram encodes the unital strictness as follows (where the dashed lines mean that we do not usually draw them):



In any monoidal category, we have the "interchange law": $(g \circ f) \otimes (i \otimes h) = (g \otimes i) \circ (f \otimes h)$. This is immediately obvious string-diagrammatically, as can be seen by the following diagram which represents both sides of the equation:



This is one way in which string diagrams make things much simpler compared to the "1-dimensional" language of pure text.

This syntax is sound and complete for reasoning in monoidal categories, so long as we equate diagrams up to "planar isotopy", i.e., two diagrams, drawn on the plane, within a boundary rectangle, with all input and output strings touching the boundary rectangle, are equal if and only if it is possible to transform, continuously, one into the other by continuously moving around morphisms inside the boundary rectangle, disallowing any crossing of strings or boxes, and disallowing any strings attached to the boundary rectangle from becoming unattached. Then,

Theorem 1.2.5: [JS91, Theorem 1.2]

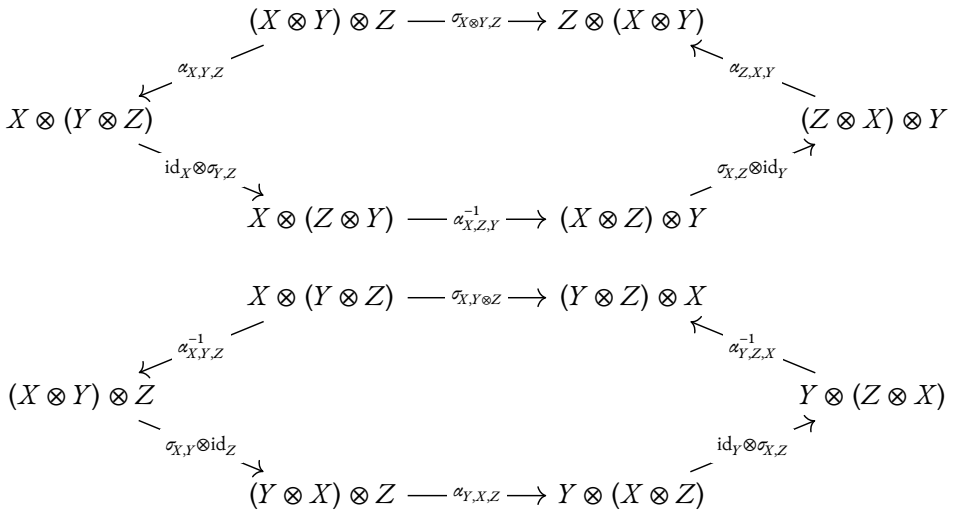
Two morphisms are equal in a monoidal category if and only if the two string diagrammatic representation of these morphisms are equal up to planar isotopy.

1.2.2 Symmetric Monoidal Categories

Given Theorem 1.2.7, it may be interesting to consider diagrams in which strings are allowed to cross over each other (at least sometimes). This gives rise to the notion of a symmetric monoidal category:

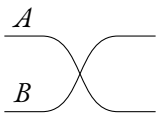
Definition 1.2.6: Symmetric Monoidal Category

A monoidal category $\mathcal{C}$ is a *symmetric monoidal category* if it has a braiding natural isomorphism σ with components $\sigma_{A,B} : A \otimes B \to B \otimes A$, which is self-inverse: $\sigma_{B,A} \circ \sigma_{A,B} = \mathrm{id}_{A\otimes B}$, and satisfies the hexagon equations

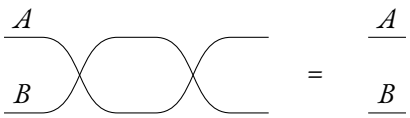


for all objects X, Y, Z of $\mathcal{C}$.

We represent the braiding $\sigma_{A,B}$ in a symmetric monoidal category with the diagram

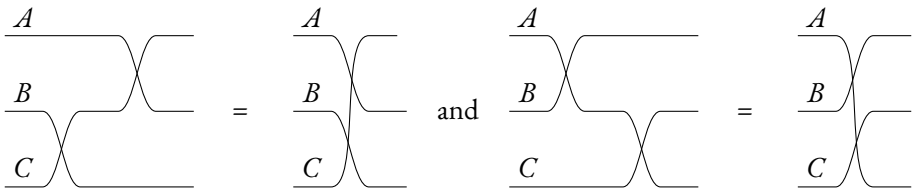


The self-inverse property ensures that

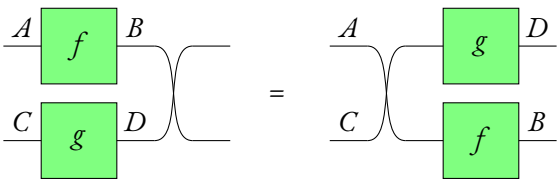


(1.3)

And the hexagon equations ensure that



The natural transformation property ensures that:



(1.4)

Soundness and completeness for this graphical calculus is then weakened to just being isotopic; i.e.,

Theorem 1.2.7: [JS91, Theorem 2.3]

Two morphisms are equal in a symmetric monoidal category if and only if the two string diagrammatic representation of these morphisms are equal up to isotopy (that is, they are isotopic, or, equivalently, isomorphic).

Next, I'll introduce functor boxes.

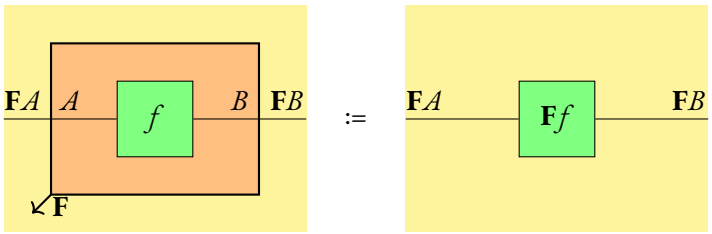
1.3 Functor Boxes

Functor boxes allow us to reason string diagrammatically with functors. There are two notions of functor boxes: inside-out functor boxes, and outside-in functor boxes. We shall see the former first – these are the original notion of functor boxes, as introduced in [Mel06].

1.3.1 Inside-out Functor Boxes

Definition 1.3.1: Functor Box

Let $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ be a functor. An *(inside-out) functor box* (representing $\mathbf{F}$) is a diagram of the kind on the left, which is defined to be equal to a diagram on the right.



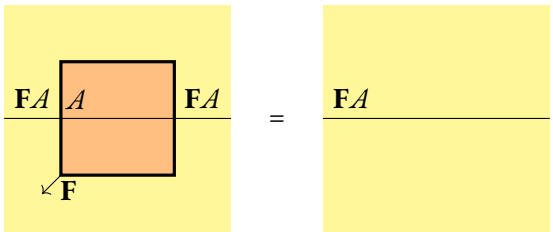
(1.5)

Remark 1.3.2. Note that in Equation 1.5, I start using colours. Yellow and orange demonstrate that the ambient category is $\mathcal{D}$ and $\mathcal{C}$, respectively. I will often use the notation $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ to demonstrate which colours denote which ambient category.

We can now see that these behave just like functors, in that they preserve composition, and identities.

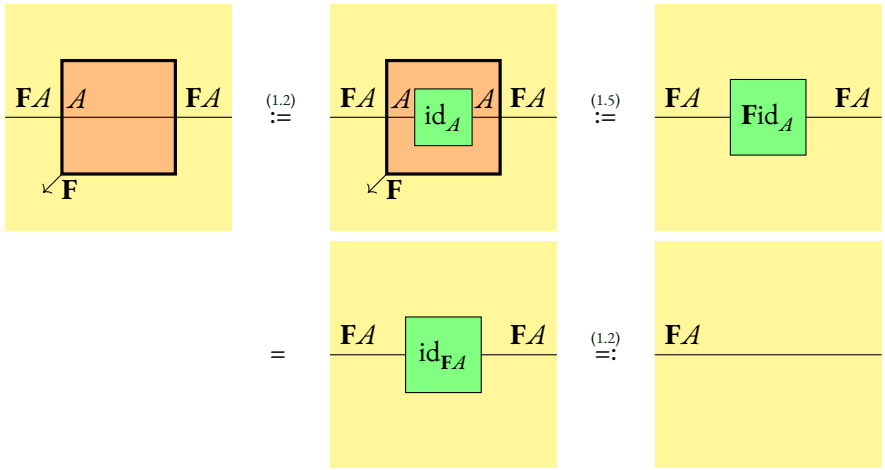
Lemma 1.3.3

Inside-out functor boxes preserve identities. That is, given $\mathbf{F} : \mathcal{C} \to \mathcal{D}$,



holds.

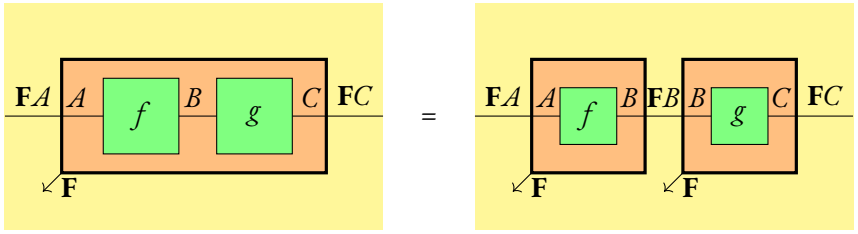
Proof:



■

Lemma 1.3.4

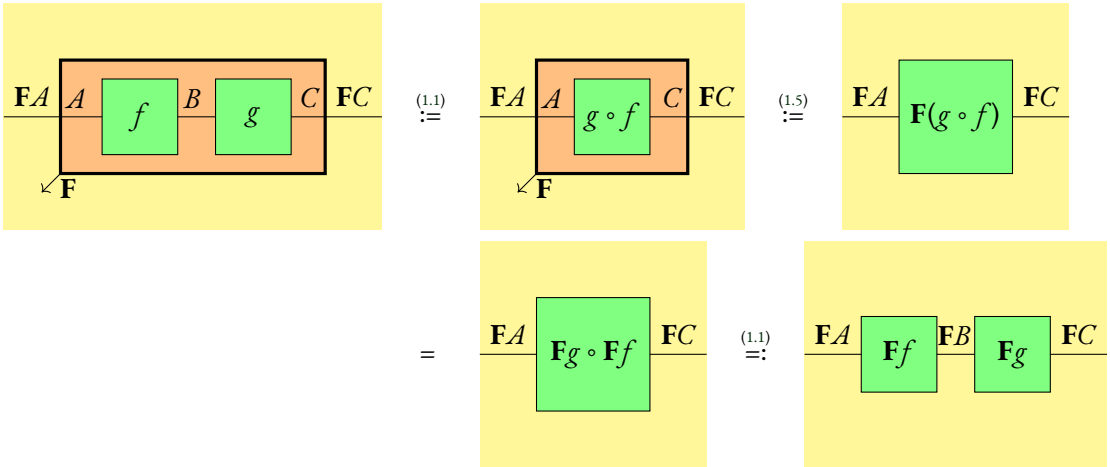
Inside-out functor boxes preserve composition. That is, given $\mathbf{F} : \mathcal{C} \to \mathcal{D}$,

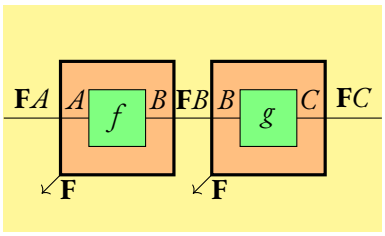


(1.6)

holds.

Proof:



$\overset{(1.5)}{=:}$ 

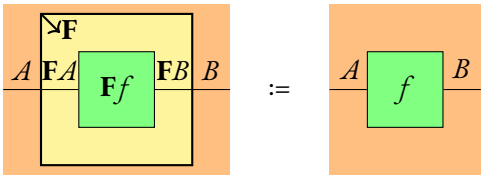
■

1.3.2 Outside-In Functor Boxes

There is an alternative convention for functor boxes, which makes sense when the functor in question is faithful. These functor boxes go in the opposite direction to the functor:

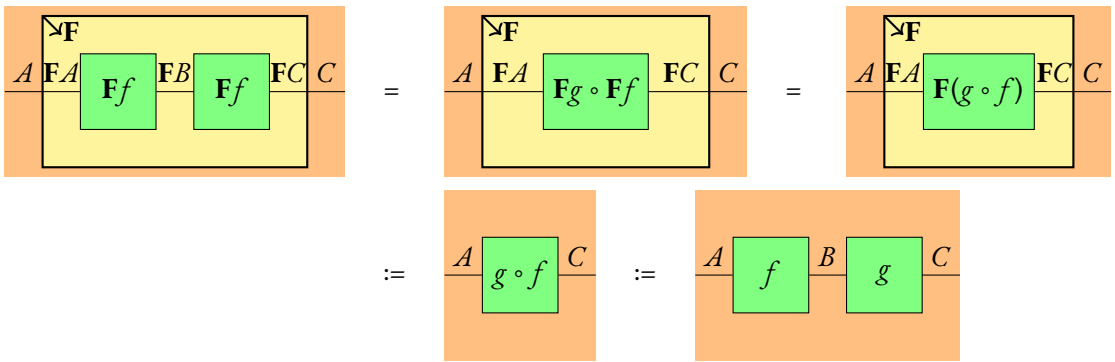
Definition 1.3.5: Outside-In Functor Boxes

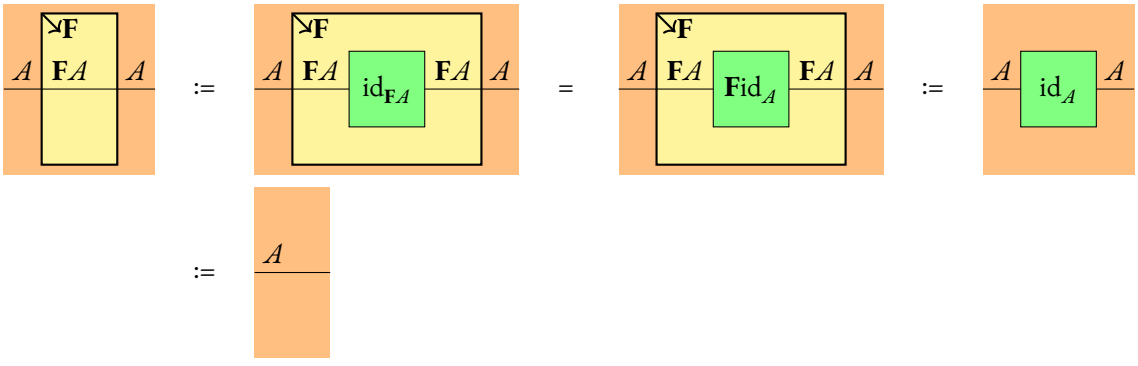
Let $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ be a faithful functor. Then we define,



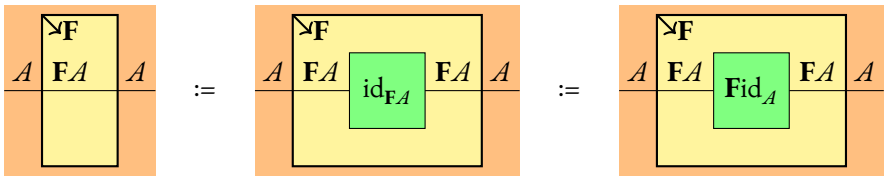
this makes sense as long as everything inside the outside-in functor box is indeed in the codomain of $\mathbf{F}$.

These functor boxes also behave like functors, preserving composition and identities...





Moreover, these are the diagrammatic outer-inverse of a normal functor box:

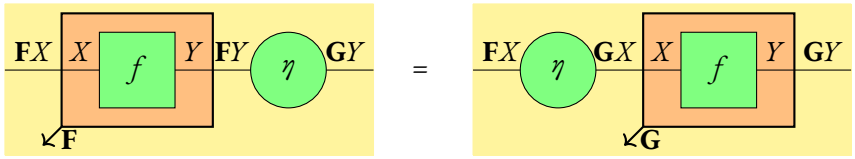


1.4 Natural Transformations

Using functor boxes, we can use string-diagrams to represent natural transformations, isomorphisms between categories, as well as adjunctions between functors. That is what we shall see now.

Definition 1.4.1: Graphical Natural Transformations

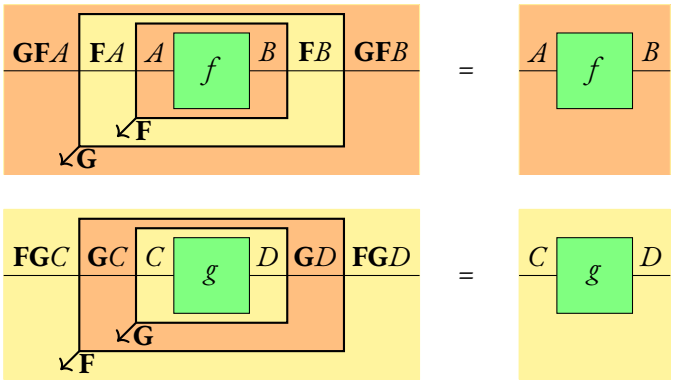
Let $\mathbf{F}, \mathbf{G} : \mathcal{C} \to \mathcal{D}$ be functors. Then $\eta : F \Rightarrow G$ is a natural transformation if for all $f : X \to Y$ in $\mathcal{C}$,



noting that we do not write the components.

Definition 1.4.2: Graphical Isomorphism of Categories

Let $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ and $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ be functors. We say that they are isomorphisms of categories if for all $f : A \to B$ in $\mathcal{C}$ and $g : C \to D$ in $\mathcal{D}$,

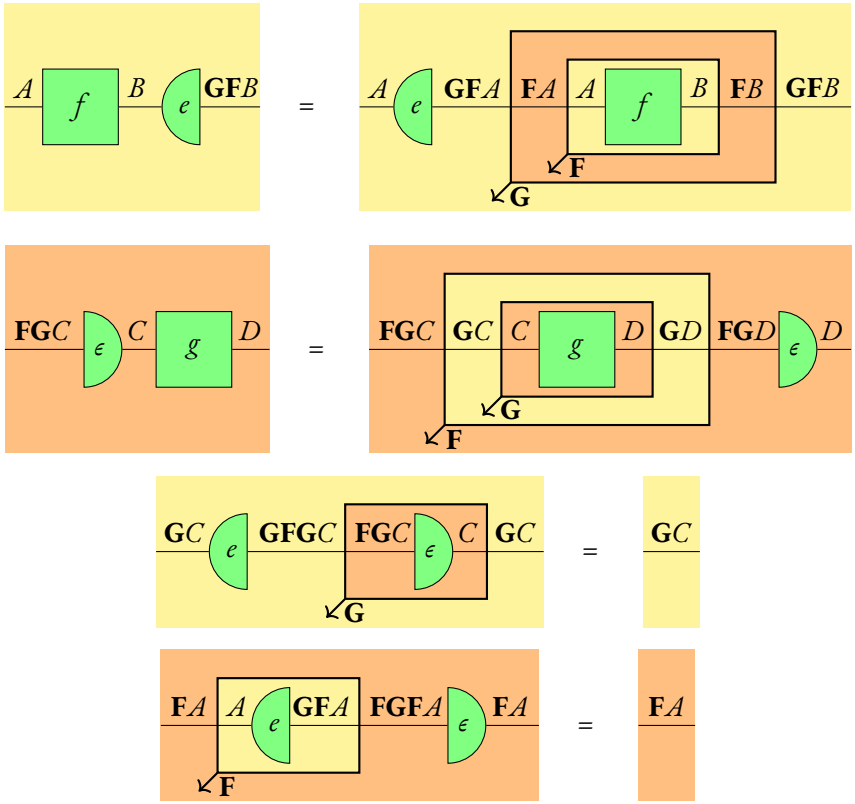


For the following, I am using the semi-circle notation for units and co-units as found in [GZ23].

Definition 1.4.3: Graphical Adjunction

Let $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ and $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ be functors. We say that $F \dashv G$, or that F is *left adjoint* to G if there exist $e : \mathrm{id}_{\mathcal{C}} \Rightarrow \mathbf{GF}$ (known as the *unit*) and $\eta : \mathbf{FG} \Rightarrow \mathrm{id}_{\mathcal{D}}$ (known as the *co-unit*) such that

for all $f : A \to B$ in $\mathcal{C}$ and $g : C \to D$ in $\mathcal{D}$,



Chapter 2

Topoi

Contents

In this chapter, we will see how we can use string diagrams to reason in arbitrary topoi. To do this, we will look at how we can reason graphically about each of the component parts: finite limits, cartesian closedness, and sub-object classifiers. To begin, we will look at finite limits.

2.1 Finite Limits

In this section, I will show how we can reason graphically in categories with finite limits. To do this, I will first show how we can reason categories with finite products, by invoking Fox's Theorem. Then, we shall see that we can extend this, by using functor boxes that represent forgetful functors from slice categories into their base categories, in order to reason in a category with all finite limits.

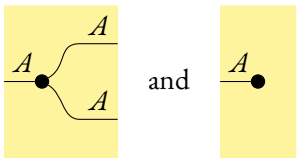
2.1.1 Products

We can extend a Theorem, due to Fox [Fox76], as follows

Theorem 2.1.1: Graphical Fox

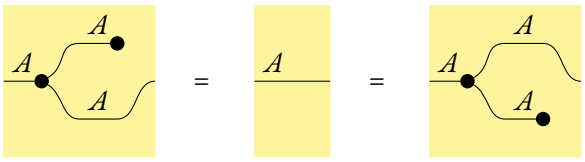
A symmetric monoidal category $\mathcal{C}$ is a cartesian category[0] if and only if for each $A \in \mathrm{Ob}\,\mathcal{C}$, there

exist morphisms (known as the *copy morphisms* and *deleting morphisms* respectively)

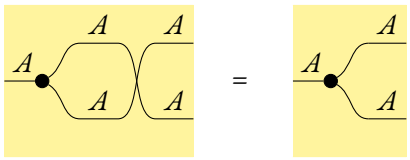


such that

- for each $A \in \mathrm{Ob}\,\mathcal{C}$,

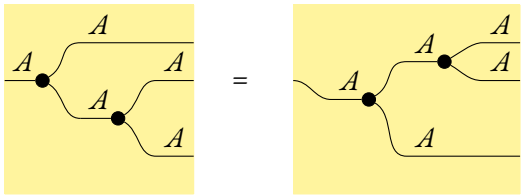


(2.1)



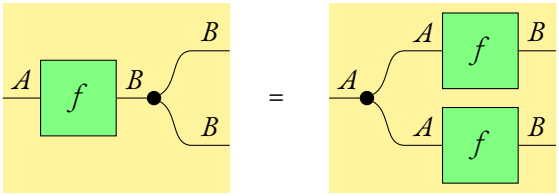
(2.2)

and



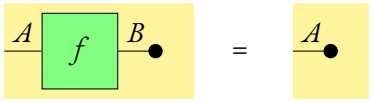
(2.3)

- for each $f : A \to B$ in $\mathcal{C}$,



(2.4)

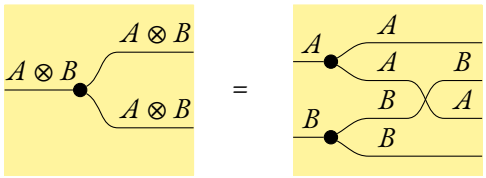
and



(2.5)

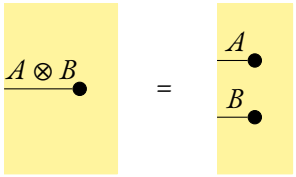
and

- for each pair of objects $\{A, B\} \subseteq \mathrm{Ob}\,\mathcal{C}$,



(2.6)

and



(2.7)

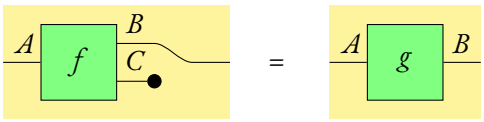
Proof: see [Rom24] for various outlines and different versions of this proof. ∎

Hence, we can use these copying and deleting morphisms to reason using string diagrams in categories with finite products, as we can use finite products to induce a monoidal structure on the category (moreover, we may assume, that the products are strict due to Theorem 1.2.3).

An example of this can now be seen in proving, graphically, the universal property of products.

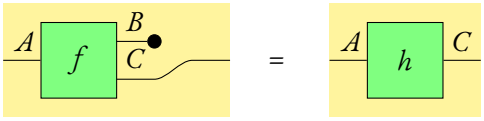
Proposition 2.1.2: Binary Products Universal Property, Graphically

Suppose that $\mathcal{C}$ has finite products and that



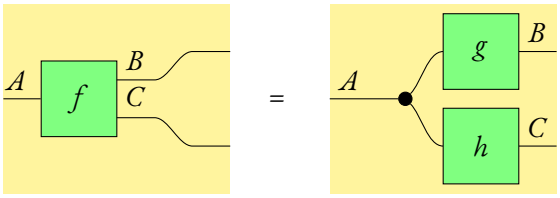
(2.8)

and also that

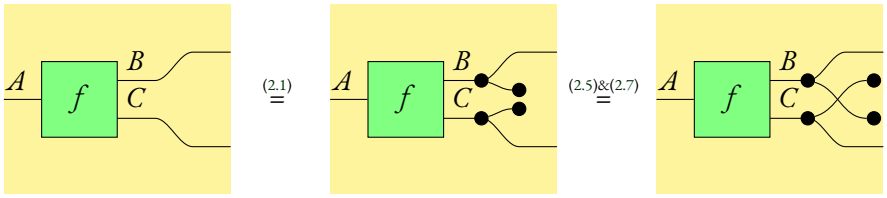


(2.9)

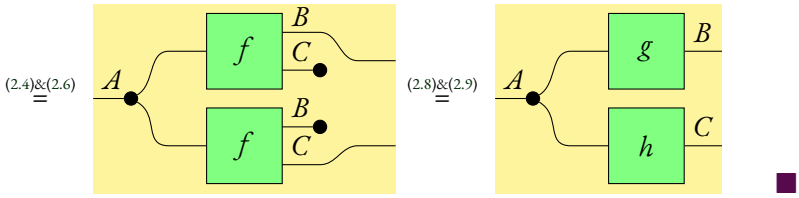
then,



Proof:



[0] That is to say the monoidal structure is given by the product functor and terminal object.



■

2.1.2 Pullbacks and Slice Categories

Let's now extend the string diagrams of the last subsection in order to reason about categories with all finite products.

First, recall (see, e.g., [nlab:fcc])

Theorem 2.1.3

A category $\mathcal{C}$ has all finite limits (i.e., is finitely complete) if and only if $\mathcal{C}$ has pullbacks and a terminal object.

this gives us the obvious corollary

Corollary 2.1.4

A category $\mathcal{C}$ has all finite limits if and only if $\mathcal{C}$ has pullbacks and finite products.

So, we just need to come up with a graphical calculus for categories with pullbacks. In order to do that, we will use slice categories.

Definition 2.1.5: Slice Category

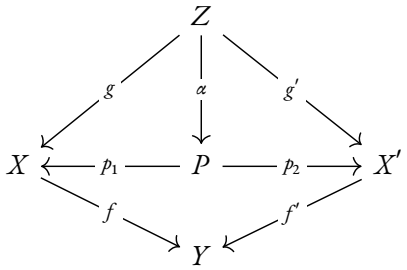
Let $\mathcal{C}$ be a category and $A \in \operatorname{Ob}\mathcal{C}$ an object. Then we define a category $\mathcal{C}/A$ called the *slice of* $\mathcal{C}$ *over* A as follows:

- objects: an object is a morphism $f : X \to A$ in $\mathcal{C}$;
- morphisms: a morphism $g : f_1 \to f_2$, where $f_1 : X_1 \to A$ and $f_2 : X_2 \to A$ in $\mathcal{C}$, is a morphism $g : X_1 \to X_2$ in $\mathcal{C}$ such that $f_2 \circ g = f_1$ in $\mathcal{C}$;
- composition and identities are the same as in $\mathcal{C}$.

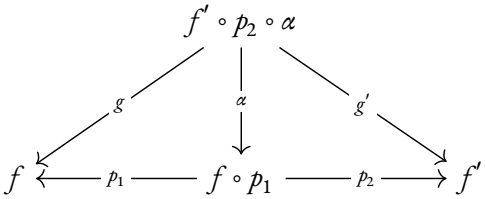
We call $\mathcal{C}$ the *base category*.

We care about slice categories because a category $\mathcal{C}$ has pullbacks if and only if every slice category has finite products. So, if there is a canonical functor from each slice category into the base category, we can use a functor box to represent this situation. That is what I will now prove.

Lemma 2.1.6



commutes in $\mathcal{C}$ if and only if



commutes in $\mathcal{C}/Y$.

Proof: ($\Longrightarrow$) First, note that each object in the bottom diagram is indeed an object of $\mathcal{C}/Y$ because they each are morphisms in $\mathcal{C}$ with domain Y. Moreover, each morphism in the second diagram is typed correctly:

- $f \circ g = f' \circ g' = f' \circ p_2 \circ \alpha$, showing g is well-typed;
- $f \circ p_1 \circ \alpha = f' \circ p_2 \circ \alpha$, showing α is well-typed;
- $f' \circ g' = f' \circ p_2 \circ \alpha$, showing g' is well-typed;
- $f \circ p_1 = f \circ p_1$, showing p_1 is well-typed; and
- $f' \circ p_2 = f \circ p_1$, showing p_2 is well-typed.

Hence, everything in the second diagram is typed correctly. It then obviously commutes as the first diagram does.

($\Longleftarrow$) taking Z, X, and X' to be the appropriate domains, we see that the first diagram is correctly typed. The second diagram shows us immediately that the large top triangle commutes in the first diagram, so all that needs to be verified is that $f \circ p_1 = f' \circ p_2$. This holds as $p_2 : f \circ p_1 \to f'$ in $\mathcal{C}/Y$. ■

Corollary 2.1.7

$\mathcal{C}$ has pullbacks if and only if for all $A \in \mathrm{Ob}\,\mathcal{C}$, $\mathcal{C}/A$ has finite products.

Proof: ($\Longrightarrow$) Suppose that $\mathcal{C}$ has pullbacks. Then consider some $\mathcal{C}/A$ and morphisms (of $\mathcal{C}$, objects of $\mathcal{C}/A$) $f : X \to A$ and $g : Y \to A$. As $\mathcal{C}$ has pullbacks, the pullback of f and g exists, but any such unique α satisfying the universal property of the pullback of f and g will, by Lemma 2.1.6, automatically satisfy the universal property of the product of f and g – showing that the binary product of f and g does indeed exist in $\mathcal{C}/A$.

For $\mathcal{C}/A$ to have finite products, it just remains to verify that it has a terminal object. The terminal

object is id_A. To see this, note that given $f : X \to A$, there is only one map $g : X \to A$ such that $\mathrm{id}_A \circ g = f$, namely f.

($\Longleftarrow$) Suppose that for all $A \in \mathrm{Ob}\,\mathcal{C}$, $\mathcal{C}/A$ has finite products. Then consider $f : X \to A$ and $g : Y \to A$ in $\mathcal{C}$. The binary product of these exists in $\mathcal{C}/A$, and any α satisfying the universal property of this product must also satisfy the universal property of the pullback of f and g in $\mathcal{C}$, by Lemma 2.1.6. ∎

Now we need some kind of canonical functor from each slice category to the base category in order to use functor boxes to characterise finite completeness. Luckily there is such a functor.

Definition 2.1.8: Slice Forgetful Functor

Let $\mathcal{C}$ be a category with $A \in \mathrm{Ob}\,\mathcal{C}$. Then we can define the forgetful functor $U_A : \mathcal{C}/A \to \mathcal{C}$ as follows:

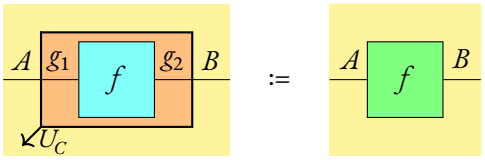
- on objects: $f : X \to A$, as an object in $\mathcal{C}/A$ is mapped to X, an object in $\mathcal{C}$;
- on morphisms: $g : f_1 \to f_2$, as a morphism in $\mathcal{C}/A$ (with $f_1 : X \to A$ and $f_2 : Y \to A$ being objects in $\mathcal{C}/A$) is mapped to $g : X \to Y$, a morphism in $\mathcal{C}$.

Moreover, this functor is faithful and so we can use outside-in functor boxes to represent it.

We also use blue boxes, instead of green to represent the morphisms in the slice category. I.e.,

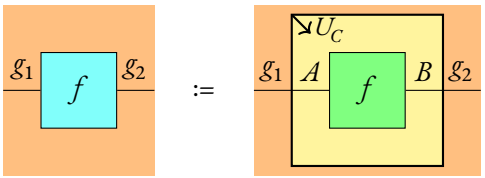
Definition 2.1.9: Blue Morphism Boxes

Let $\mathcal{C}$ be a finitely complete category, with slices $\mathcal{C}/C$, then for each $f : A \to B$ in $\mathcal{C}$, we define



(2.10)

and similarly,



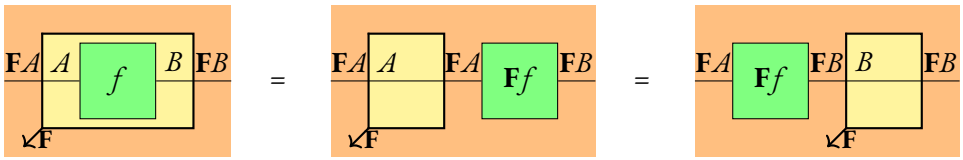
(2.11)

Remark 2.1.10. One may worry that when we assume that a finitely complete category is strict with respect to its product structure, that it also follows that the products in the slice category may also be assumed to be strict. Luckily, however, we can see that the products in the slice categories are always strict, irrespective of the properties of the base category, as they are defined in terms of morphisms, which are associative and have units anyway.

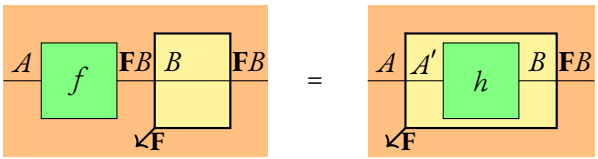
Another nice property of this forgetful functor can be seen next.

2.1.3 Discrete Fibrations

In general, it is always possible to expel morphisms from inside-out functor boxes. This can be seen in the following equality:



However, is is not possible, in general, for inside-out functor boxes to eat morphisms. That is, there may be no A' and h such that the following equality holds.



One of the reasons for this is that the functor may not be surjective (i.e., full and surjective on objects); there may not be any morphism in the domain category to map onto the morphism in the codomain. We can generalise surjectivity of functors into a kind of directed surjectivity. This generalisation is known as the property of being a discrete (op-)fibration. For more on fibrations, see [LR20].

Definition 2.1.11: Discrete Fibration

A functor $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ is a *discrete fibration* if for each $C \in \mathcal{C}$, $D \in \mathcal{D}$, and $g : D \to \mathbf{F}C$ in $\mathcal{D}$, there is a unique morphism $h : C' \to C$ in $\mathcal{C}$ such that $\mathbf{F}h = g$.

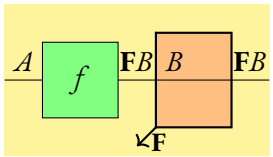
Definition 2.1.12: Discrete Op-fibration

A functor $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ is a *discrete op-fibration* if for each $C \in \mathcal{C}$, $D \in \mathcal{D}$ and $g : \mathbf{F}C \to D$ in $\mathcal{D}$, there exists a unique morphism $h : C \to C'$ in $\mathcal{C}$ such that $\mathbf{F}h = g$.

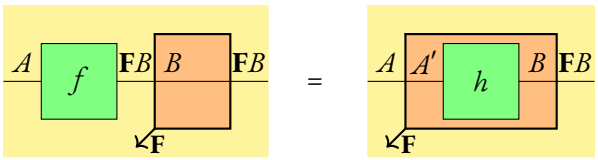
Graphically, this corresponds to the fact that functor boxes can eat morphisms from the left or right, depending on whether the functor is a discrete fibration or a discrete op-fibration.

Theorem 2.1.13: Graphical Discrete Fibrations

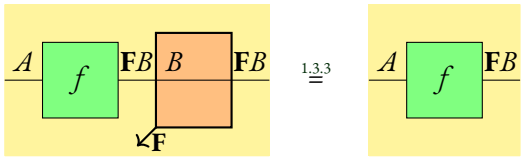
A functor $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ is a discrete fibration if and only if its inside-out functor box can eat, uniquely, morphisms on the left. That is, whenever



is a morphism, then there exists a unique h such that

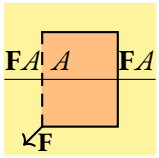


Proof: this is immediate from the fact that



■

When a functor is a discrete fibration, I draw its box with a porous left-side, to denote that morphisms may freely pass over this boundary, like

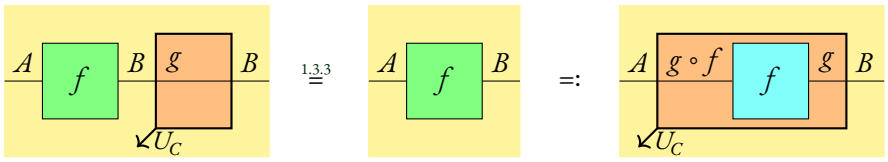


similarly, when a functor is a discrete op-fibration, we may draw its box with a porous right-side.

Proposition 2.1.14

The forgetful functors from slice categories are discrete fibrations.

Proof: consider a category $\mathcal{C}$ with slice $\mathcal{C}/C$; whenever a morphism like the left hand side exists, the following equality holds:



as $g \circ f = g \circ f$. Uniqueness follows by the fact that the functor is faithful. ■

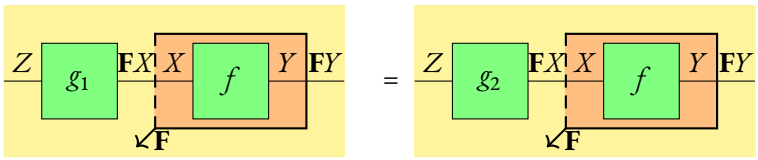
So, from now on, I will draw the inside-out functor boxes for the forgetful functors from slice categories with a porous left side.

One interesting property of faithful discrete fibrations is that they preserve and reflect monomorphisms; we will use this later to prove the Fundamental Theorem of Topos Theory.

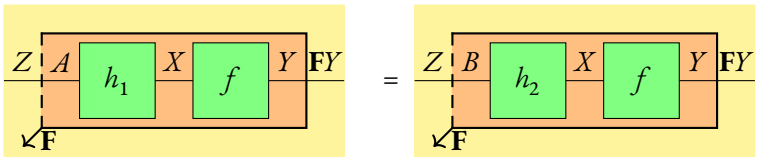
Proposition 2.1.15: Faithful Discrete Fibrations Preserve/Reflect Monomorphisms

Let $\mathbf{F} : \mathcal{C} \to \mathcal{D}$ be an faithful discrete fibration. Then $f : X \to Y$ is monic in $\mathcal{C}$ if and only if $\mathbf{F}f$ is monic in $\mathcal{D}$.

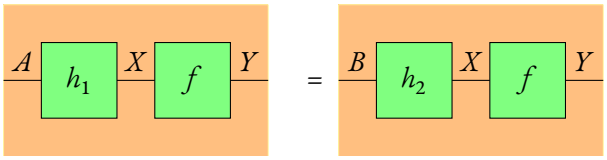
Proof: the fact that **F** reflects monomorphisms follows from the fact that it is faithful. So, I will just show that faithful discrete fibrations preserve monomorphisms. Using colouring $\mathcal{C}$ and $\mathcal{D}$, suppose that f is monic and



then, as **F** is a discrete fibration, there exist (unique) h_1 and h_2 such that,



where $\mathbf{F}h_1 = g_1$ and $\mathbf{F}h_2 = g_2$, and as **F** is faithful,



(noting then that we must have that $A = B$), which implies that $h_1 = h_2$, as f is monic. But then $g_1 = \mathbf{F}h_1 = \mathbf{F}h_2 = g_2$; i.e., $\mathbf{F}f$ is monic. ■

So, in particular, each $U_A : \mathcal{C}/A \to \mathcal{C}$ preserves and reflects monomorphisms.

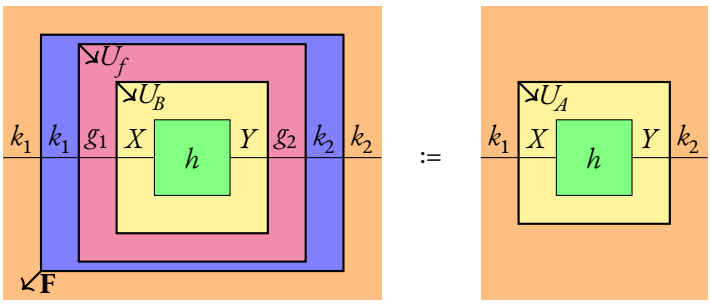
2.1.4 Slices of Slices are just Slices of the Base Category

Finally, we can see that we never need to consider slices of slices, by the following proposition.

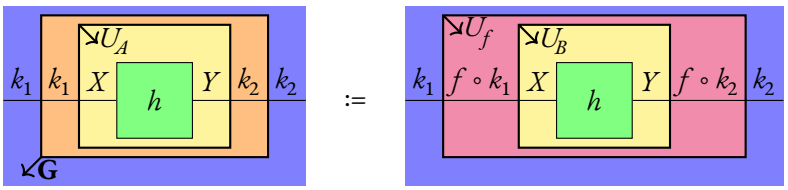
Proposition 2.1.16

Let $f : A \to B$ in $\mathcal{C}$, then $(\mathcal{C}/B)/f$ is isomorphic to $\mathcal{C}/A$.

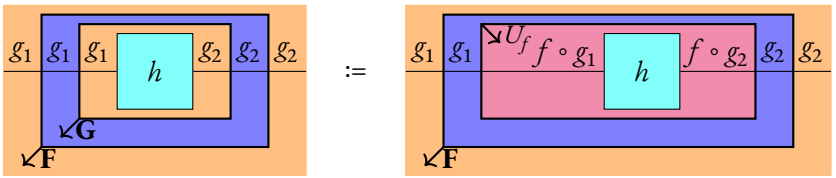
Proof: define a functor $\mathbf{F} : (\mathcal{C}/B)/f \to \mathcal{C}/A$ as follows (where we have colourings $\mathcal{C}$ and $\mathcal{C}/B$):



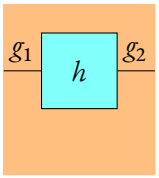
and its supposed inverse $\mathbf{G} : \mathcal{C}/A \to (\mathcal{C}/B)/f$ as follows:



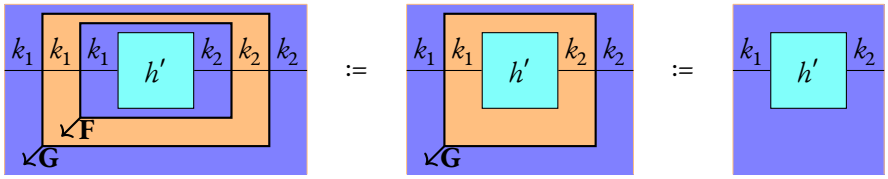
Now we see that



:=



and also that



Consequently, **F** and **G** induce an isomorphism between $(\mathcal{C}/B)/f$ and $\mathcal{C}/A$. ■

This concludes the evaluation of graphically representing categories with finite products.

2.2 Closed Categories

Given a monoidal category $\mathcal{C}$, we say it is (left or right) closed when there is a canonical object C for each pair of objects A and B such that C somehow represents the collection of morphisms from A to B. This is particularly useful when we want our category to look like **Set**, where each collection of morphisms is itself a set, as we do in the case of Topoi. For more information about closure see [EM66]. Formally, we define left and right closure as follows.

Definition 2.2.1: Left-Closed Monoidal Category

Let $\langle \mathcal{C}, \otimes, I, \alpha, \lambda, \rho \rangle$ be a monoidal category. We say that it is *left-closed* if there exists, for each $A \in \operatorname{Ob} \mathcal{C}$ a functor $A \multimap \cdot : \mathcal{C} \to \mathcal{C}$ such that it is right adjoint to the functor $A \otimes \cdot : \mathcal{C} \to \mathcal{C}$.

Remark 2.2.2. Similarly, we say that a monoidal category is *right-closed* if there is a functor $\cdot \multimap A : \mathcal{C} \to \mathcal{C}$ such that it is right adjoint to the functor $\cdot \otimes A : \mathcal{C} \to \mathcal{C}$.

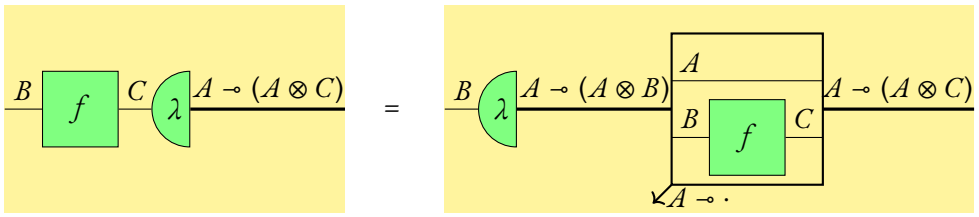
Remark 2.2.3. Any monoidal category that is both left-closed and right-closed is said to be *bi-closed*.

2.2.1 With Functor Boxes

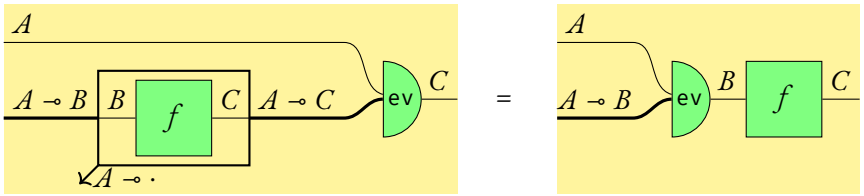
Graphically, using functor boxes, we can represent left-closure as follows.

Definition 2.2.4: Left-Closure Using Functor Boxes

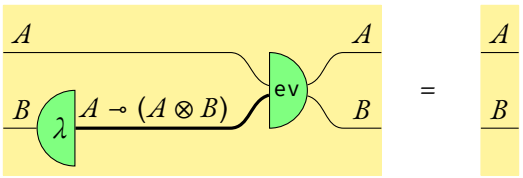
A monoidal category $\mathcal{C}$ is left-closed if and only if for every $A \in \mathrm{Ob}\,\mathcal{C}$ and $f : C \to B$, there exist natural transformations λ and ev, and functors $A \multimap \cdot$ such that the following hold (thick wires just indicate types involving $\multimap$ and have no actual affect on diagrams):



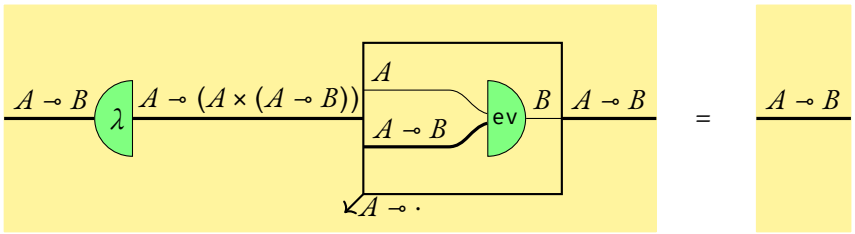
(2.12)



(2.13)



(2.14)



(2.15)

Right-closure can be represented in the obviously similar way.

2.2.2 Bastard Cups/Caps and Clasps

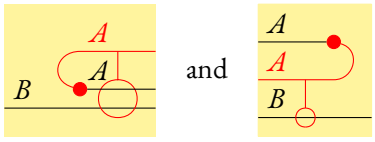
In [BS10], a notation for left-closed (or alternative, right-closed) monoidal categories is introduced. This notation is called "bubble and clasp notation". I think that this notation is unnecessarily cluttered [GZ23, p. 71], and so have slightly altered it to remove the "bubbles"[1]. Moreover, the original notation has not been proved to be coherent [Wij14, p. 27]. But using the machinery of functor boxes, we can easily prove coherence for the altered notation here. And I strongly conjecture (but do not prove) that the original bubble and clasp notation is therefore also coherent.

Let's now see this altered notation, which I call "bastard cap/cup and clasp notation"[2].

[1] Also strings going in different directions have been replaced with colour!
[2] These are named after the "bastard spiders" of [CK17], which similarly connect wires of different kinds.

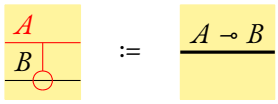
Definition 2.2.5: Downward Bastard Cups and Caps

Let $\mathcal{C}$ be a monoidal category. We say it admits *downward bastard cups and caps* if there exist, for each $\{A, B\} \subseteq \mathrm{Ob}\,\mathcal{C}$, morphisms

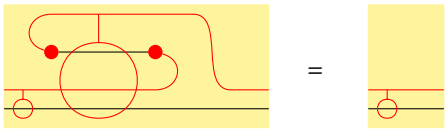


(2.16)

where

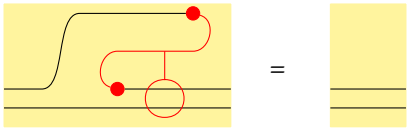


(so the "clasps" are just typing judgements – and so strings and morphisms may freely move in and out of them, as long as the typing is still correct) and these morphisms also satisfying the yanking equations:



(2.17)

and

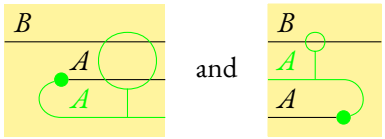


(2.18)

And similarly...

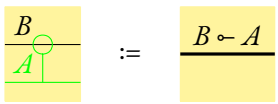
Definition 2.2.6: Upward Bastard Cups and Caps

Let $\mathcal{C}$ be a monoidal category. We say it admits *upward bastard cups and caps* if there exist, for each $\{A, B\} \subseteq \mathrm{Ob}\,\mathcal{C}$, morphisms

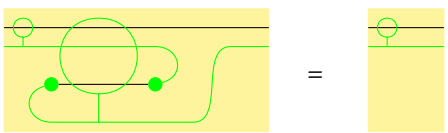


(2.19)

where

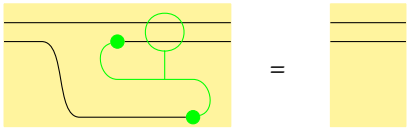


and these morphisms also satisfying the yanking equations:



(2.20)

and



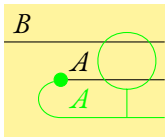
(2.21)

Remark 2.2.7. One thinks of these red (or green) strings as being "in the opposite category", and so they send information from right to left, rather than left to right. The black strings are treated as normal strings of that type, so morphisms can be applied to them (and morphisms may slide through the "clasps" freely, as though they were not there, like in a normal monoidal category).

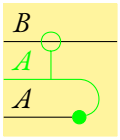
Theorem 2.2.8: Coherence of Bastard Cups and Caps

A monoidal category $\mathcal{C}$ has upward bastard cups and caps if and only if it is left-closed. Similarly, it has downward bastard cups and caps if and only if it is right-closed.

Proof: I will only consider left-closure. Defining



to be the unit and



to be the co-unit of the adjunction immediately yields the equivalence between the functor box and upward bastard cup/cap notation.

The effect of the functor box is to take a morphism and surround it with clasps. This immediately yields the naturality conditions as we may pull morphisms freely though claps. Similarly, the true adjointness conditions are precisely the yanking equations. ∎

2.2.3 Symmetric Monoidal Closed Categories

In a symmetric category, we can easily see the following proposition (see e.g., [nlab:cmc]).

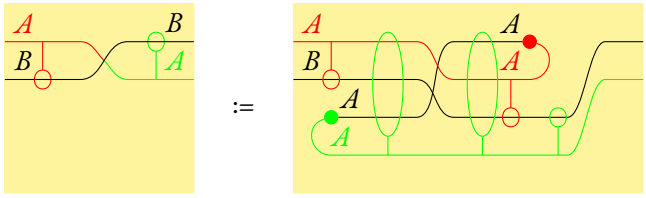
Proposition 2.2.9

A category symmetric monoidal $\mathcal{C}$ is bi-closed if and only if it is either left-closed or right-closed.

Moreover, in any bi-closed symmetric monoidal category, we can define a (natural) isomorphism between the two kinds of closure. Graphically, this looks very nice, resembling a swap morphism.

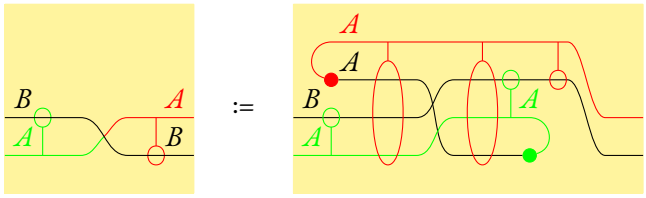
Definition 2.2.10

Let $\mathcal{C}$ be a biclosed symmetric monoidal category, with objects A and B. Then we define



(2.22)

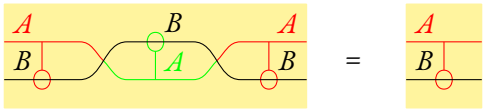
and



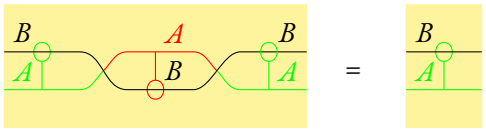
(2.23)

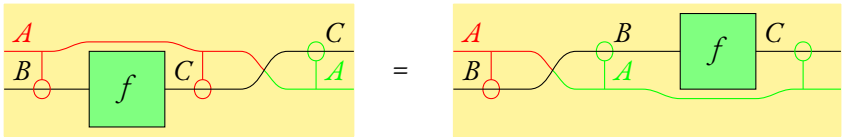
Theorem 2.2.11

Let $\mathcal{C}$ be a biclosed symmetric monoidal category, with object A and morphism $f : B \to C$. Then the following hold

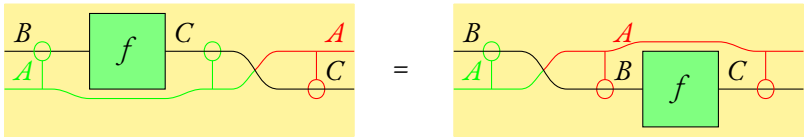


(2.24)



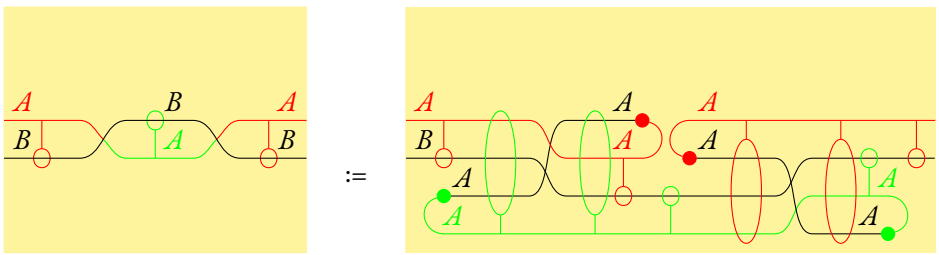


(2.25)

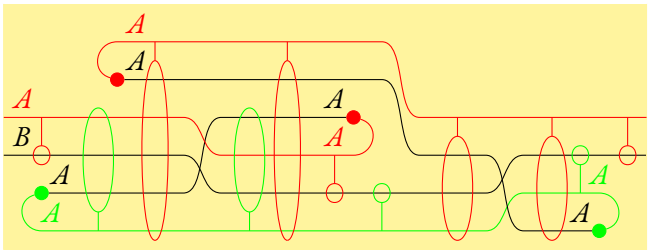


demonstrating that these swapping morphisms are in fact natural isomorphisms.

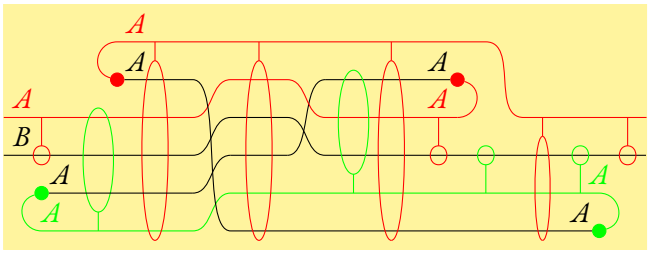
Proof: I will show Equation 2.24 and Equation 2.25; the other two are similar (in fact you can just horizontally flip the proofs and swap the red and green strings).



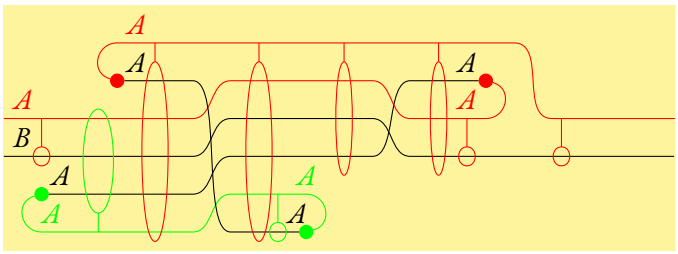
=



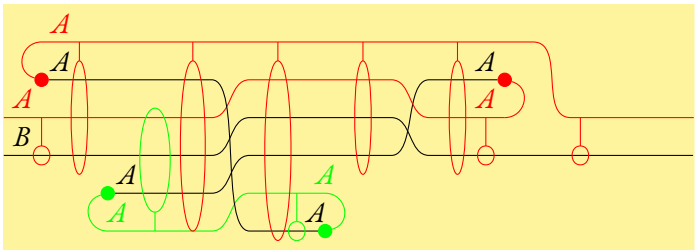
(1.4)
=



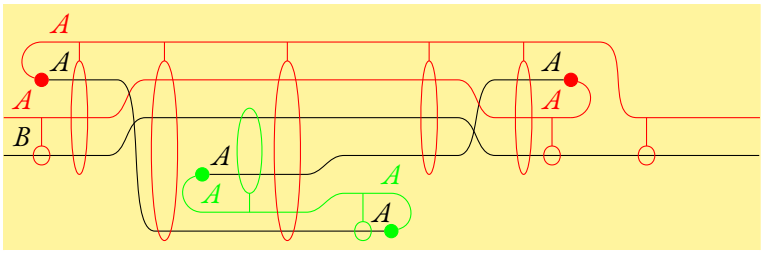
=



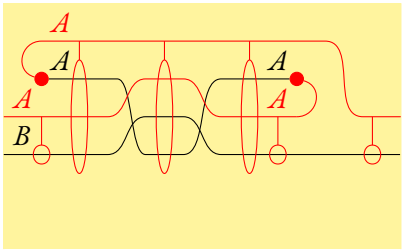
=



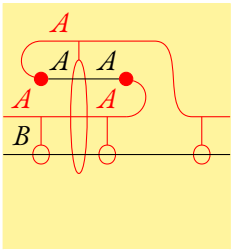
(1.4)
=



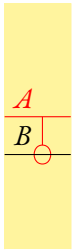
(2.21)
=

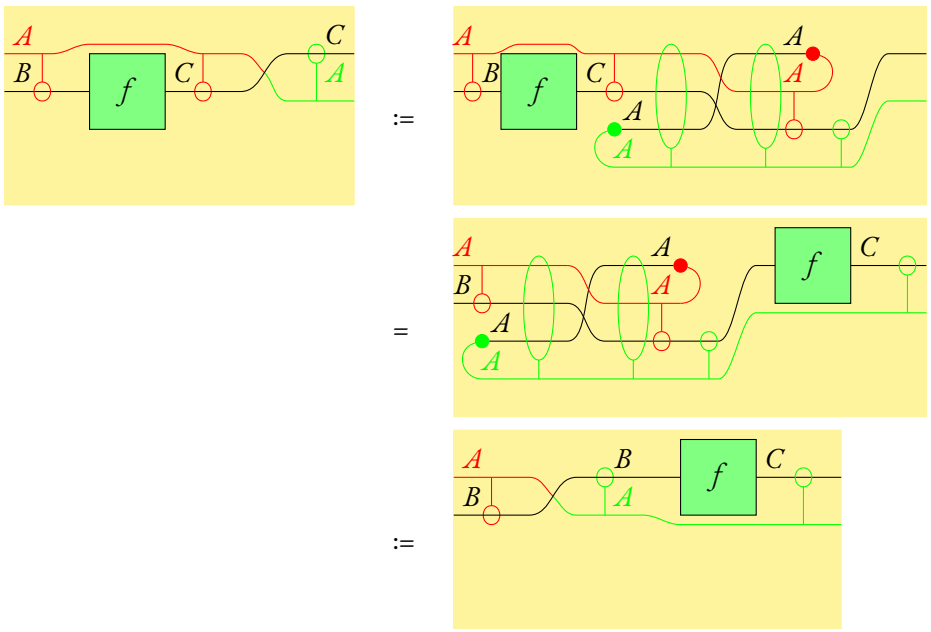


(1.3)
=



(2.17)
=





This means that when reasoning in a symmetric monoidal category it doesn't matter which kind of closure we wish to work with as we can always convert one into another via this isomorphism.

Remark 2.2.12. In the special case where the monoidal structure is given by the product structure of the category (i.e., it is a cartesian category), we say that the category is *cartesian closed*, and we usually denote the (left/right)-closed (up to author preference) as $[-]^A$ instead of $A \multimap -$ or $- \multimap A$.

This concludes the analysis of left and right monoidal closure graphically.

2.3 Sub-object Classifiers

Since a topos is in some sense a generalisation of **Set**, the sub-object classifier condition allows "subset-like" reasoning. In particular, the object Ω acts like the set $\{0, 1\}$ in **Set**, and $\mathtt{t}$ acts like 1 in this set. Subset/set-like reasoning then follows from the idea that if we can define a morphism that acts like the characteristic function of set, then we can determine which objects are "members" of others. That is how the sub-object classifier works. Formally,

Definition 2.3.1: Sub-object Classifier

Let $\mathcal{C}$ be a category with a terminal object 1. A *sub-object classifier* of $\mathcal{C}$ is a pair $\langle \Omega, \mathtt{t} \rangle$, where

- Ω is in $\mathrm{Ob}\,\mathcal{C}$; and
- $\mathtt{t} : 1 \to \Omega$ is a morphism in $\mathcal{C}$,

such that for every monic $m : A \to X$ in $\mathcal{C}$, there exists a unique morphism $\chi_m : X \to \Omega$ such that

$$\begin{array}{ccc} A & \xrightarrow{!} & 1 \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \mathrm{t}} \\ X & \xrightarrow{\chi_m} & \Omega \end{array}$$

commutes, and is a pullback square.

Remark 2.3.2. We call χ_m the *characteristic morphism* of m.

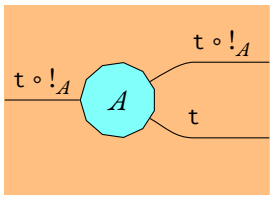
I am going to give graphical conditions for sub-object classifiers in arbitrary categories (with a terminal object). As such the slices in general will not have finite products. However, I will use diagrams to represent a partial monoidal structure induced by those products that do exist. Some care will be needed to reason in these kinds of categories. In particular, we can only form the monoidal product $f \otimes g$ if both the monoidal product of the domain and codomain are known to exist. And as such, the vertical composition of two strings is an existence claim that the monoidal product of the two objects represented by the strings exist.

In order to give the graphical conditions for sub-object classifiers in full generality, we will need the existence of certain morphisms. These morphisms will be used here – later we shall see that they automatically exist in any category with finite products, and thus will be discharged after giving the graphical conditions for sub-object classifiers. Here they are:

Definition 2.3.3

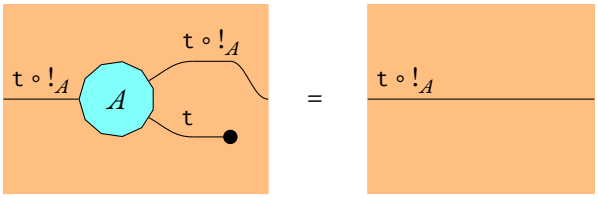
Special morphism's properties.

For every $A \in \mathrm{Ob}\,\mathcal{C}$,



(2.26)

exists in $\mathcal{C}/\Omega$. Moreover, the following equality holds:



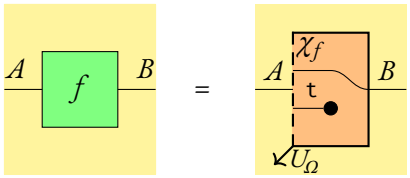
(2.27)

Now we can see the full graphical conditions for categories with sub-object classifiers.

Theorem 2.3.4: Graphical Sub-object Classifiers

A category $\mathcal{C}$, with a terminal object 1, has a sub-object classifier $\langle \Omega, \mathrm{t} \rangle$ if and only if

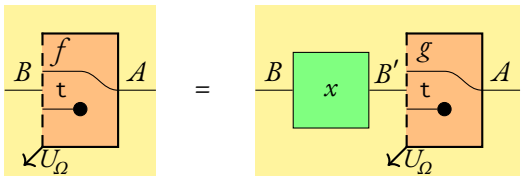
- $\mathcal{C}$ has "special morphisms" in $\mathcal{C}/\Omega$;
- if $f : A \to B$ is monic in $\mathcal{C}$, then there exists $\chi_f : B \to \Omega$ in $\mathcal{C}$ such that



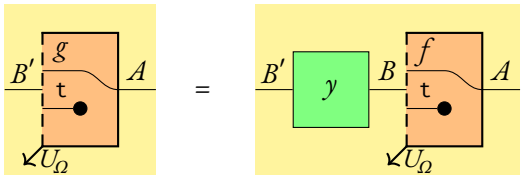
(2.28)

holds; and

- for every f and g in $\mathcal{C}$, if there are maps x and y such that

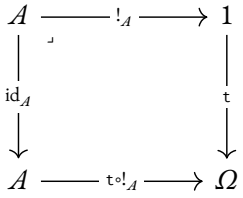


and



then $f = g$.

Proof: ($\Longrightarrow$) For the first condition, note that the following is a pullback square for each A in Ob $\mathcal{C}$:



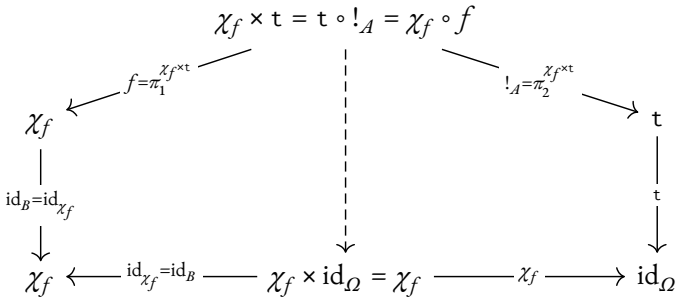
which implies that the product $\mathtt{t} \circ !_A \times \mathtt{t}$ exists in $\mathcal{C}/\Omega$. Then we can define the "special morphism" on A to be $\langle \mathrm{id}_{\mathtt{t}\circ!_A}, !_A \rangle : \mathtt{t} \circ !_A \to \mathtt{t} \circ !_A \times \mathtt{t}$, demonstrating its existence. Then the required property can be demonstrated by the following:

$$(\mathrm{id}_{\mathtt{t}\circ!_A} \times \mathtt{t}) \circ \langle \mathrm{id}_{\mathtt{t}\circ!_A}, !_A \rangle = \langle \mathrm{id}_{\mathtt{t}\circ!_A} \circ \mathrm{id}_{\mathtt{t}\circ!_A}, \mathtt{t} \circ !_A \rangle = \mathrm{id}_{\mathtt{t}\circ!_A} \times \mathrm{id}_{\Omega} = \mathrm{id}_{\mathtt{t}\circ!_A}$$

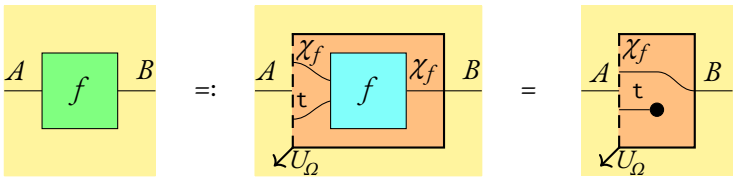
demonstrating the required property. So, the "special morphisms" exist.

For the second condition, suppose $f : A \to B$ is monic. Then there exists a unique $\chi_f : B \to \Omega$ such

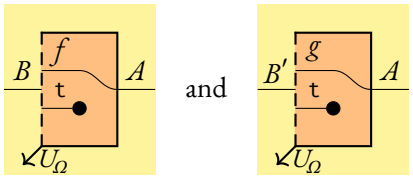
that $\chi_f \circ f = \mathtt{t} \circ !_A$. Moreover, f makes the following diagram commute (in $\mathcal{C}/\Omega$):



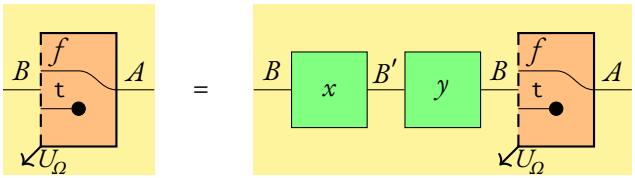
and by the universal property of products, we obtain that $f = \mathrm{id}_{\chi_f} \times \mathtt{t}$ in $\mathcal{C}/\Omega$, and so,



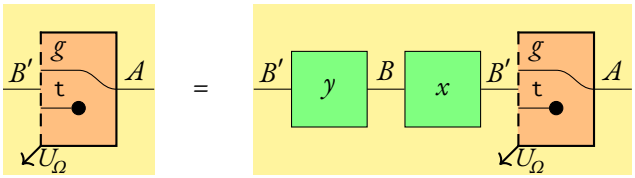
For the third condition, note that x and y are isomorphisms. This is because



are both monic (because $\mathtt{t}$ is monic), and so as

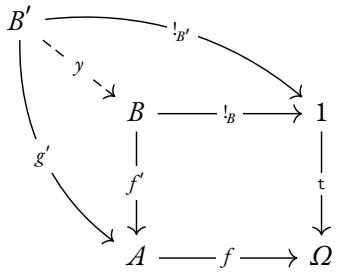


and



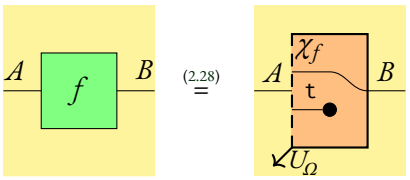
we must have that $x \circ y$ and $y \circ x$ are the respective identities.

So, if we define f' to be the morphism classified by f and g' by g, we must have that



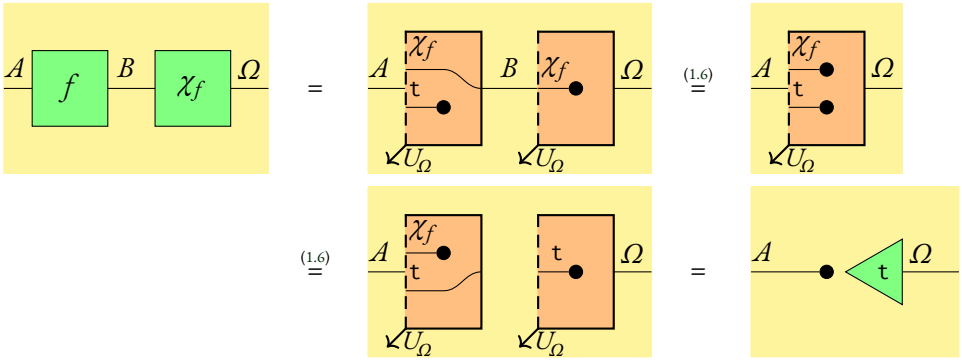
commutes, and as y is an isomorphism, and the inner square is a pullback square, the outer square must also be a pullback square; and therefore f must be the classifying morphism for g', which means that $f = g$, as the classifying morphisms are unique.

($\Longleftarrow$) Suppose $f : A \to B$ is monic. Then, there exists $\chi_f : B \to \Omega$ in $\mathcal{C}$ such that

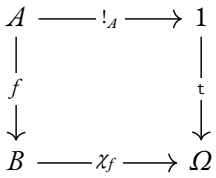


(2.29)

and so

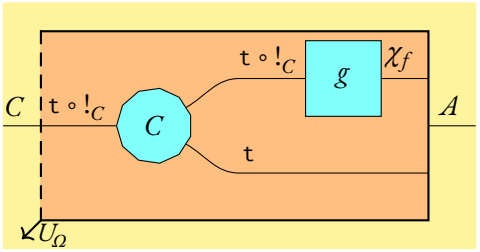


and to see that the square



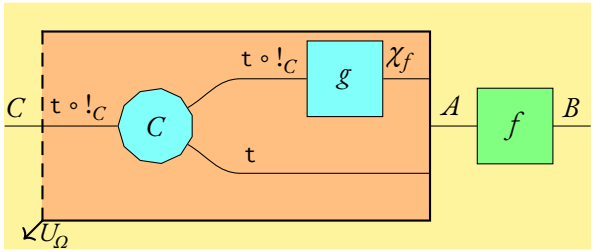
is a pullback, consider a morphism $g : C \to B$ in $\mathcal{C}$, and suppose that $\chi_f \circ g = \mathtt{t} \circ !_C$. Then, for this square to be a pullback, we just require the existence of a $u : C \to A$ such that $f \circ u = g$.[2]

Consider the following:

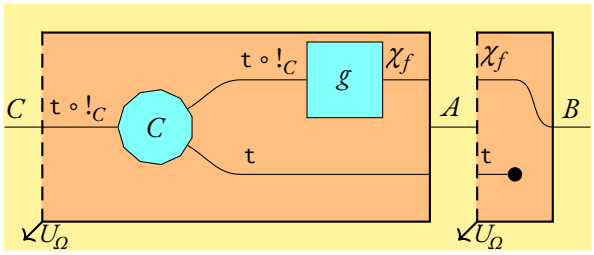


I will show that this is the required unique u.

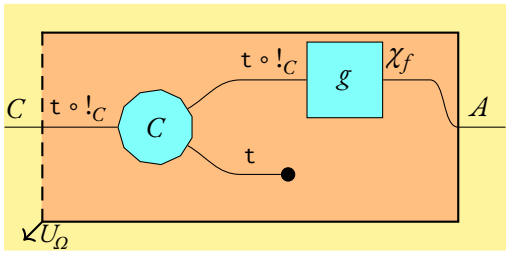
To see that it has the desired property, consider:



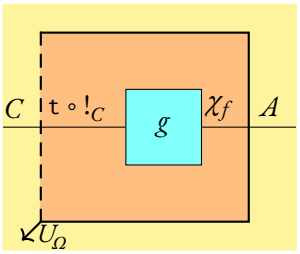
$\overset{(2.29)}{=}$



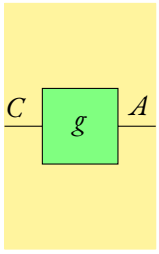
$\overset{(1.6)}{=}$



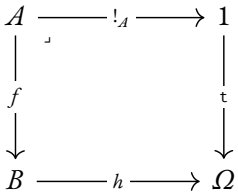
$\overset{(2.27)}{=}$



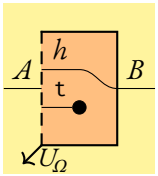
$=$



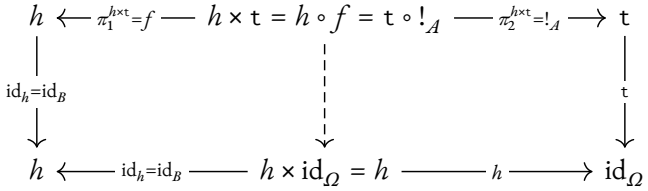
Finally to see that χ_f is the unique morphism with the above properties, suppose that



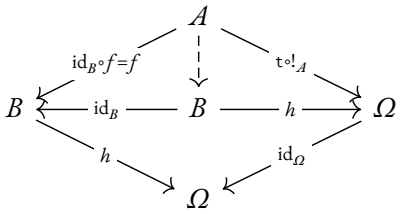
is a pullback square. Then,



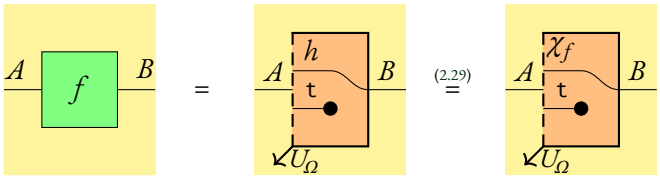
is the unique map making



commute in $\mathcal{C}/\Omega$, which is by Lemma 2.1.6, the unique map making



commute in $\mathcal{C}$. Clearly f makes this diagram commute and so,



which implies that $h = \chi_f$, by the third condition, – in other words, χ_f is the unique morphism with these properties. ∎

This concludes the graphical analysis of sub-object classifiers. Now we can represent topoi graphically.

[2]Note that $!_A \circ u = !_C$ follows trivially for any $u : C \to A$. Similarly, any u such that $f \circ u = g$ must be unique, as if $f \circ u' = g$, then as f is monic, $u = u'$.

2.4 Topoi

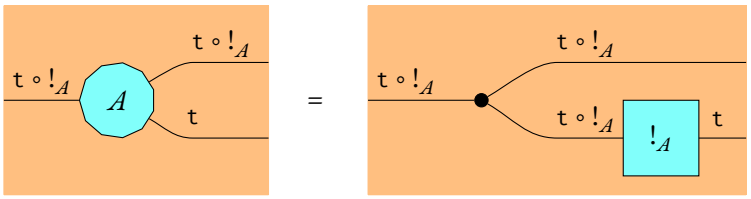
Definition 2.4.1: Topos

A *topos* is a cartesian closed category with finite limits and a sub-object classifier.

Before I give the exact graphical calculus for topoi, I will first prove that we never need to worry about "special morphisms" in the case of topoi.

Proposition 2.4.2

A category $\mathcal{C}$ (with a terminal object 1, and a morphism $\mathtt{t} : 1 \to \Omega$) has the product $A \times A$ for each $A \in \mathrm{Ob}\,\mathcal{C}$, if and only if $\mathcal{C}$ has "special morphisms" and



(2.30)

holds in $\mathcal{C}/\Omega$.

Proof: in order for the equality Equation 2.30 to hold, we need the products $\mathtt{t} \circ !_A \times \mathtt{t}$ and $\mathtt{t} \circ !_A \times \mathtt{t} \circ !_A$ to exist in $\mathcal{C}/\Omega$. The former always exists, as the pullback of $\mathtt{t} \circ !_A$ and $\mathtt{t}$ always exists by the following pullback square:

$$\begin{array}{ccc} A & \xrightarrow{\mathrm{id}_A} & A \\ {\scriptstyle !_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathtt{t}\circ !_A} \\ 1 & \xrightarrow{\mathtt{t}} & \Omega \end{array}$$

and similarly, the product $\mathtt{t} \circ !_A \times \mathtt{t} \circ !_A$ exists in $\mathcal{C}$ if and only if the pullback of $\mathtt{t} \circ !_A$ with itself exists. I.e., the following is a pullback square:

$$\begin{array}{ccc} P & \xrightarrow{\mathfrak{p}_2} & A \\ {\scriptstyle \mathfrak{p}_1}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathtt{t}\circ !_A} \\ A & \xrightarrow{\mathtt{t}\circ !_A} & \Omega \end{array}$$

and it is not hard to see that P together with $\mathfrak{p}_1$ and $\mathfrak{p}_2$ must be the product of A and A in $\mathcal{C}$. ∎

Hence, we may conclude,

Theorem 2.4.3: Topoi, Graphically

A category $\mathcal{C}$ is a topos if and only if its string diagrams admit:

- copy and deleting maps, as given by Fox's Theorem;
- products in each of the slice category forgetful functor boxes;
- upward/downward bastard caps/cups; and
- the latter two conditions of Theorem 2.3.4 are satisfied.

Chapter 3

The Fundamental Theorem

Contents

We shall now see the Fundamental Theorem of Topos Theory [Fre72, p. 24]. It is in two parts. The first part, which we shall see in the next section, says that in a topos, each slice category is also a topos. The second part says that there is a canonical functor, called the "pullback functor", between each $\mathcal{T}/B$ and $\mathcal{T}/A$ given a morphism $f : A \to B$ in the base category $\mathcal{T}$, which is also a topos, and that this functor has left and right adjoints.

The proof will be done entirely using string diagrams, and in places, which I shall highlight, will be much simpler than the non-graphical alternatives.

3.1 Slices of Topoi are Topoi

For the first part of the Fundamental Theorem of Topos Theory, to show that each slice category is a topos, I will show, in turn, that each slice category of a topos has finite limits, a sub-object classifier, and exponentials.

3.1.1 Finite Limits

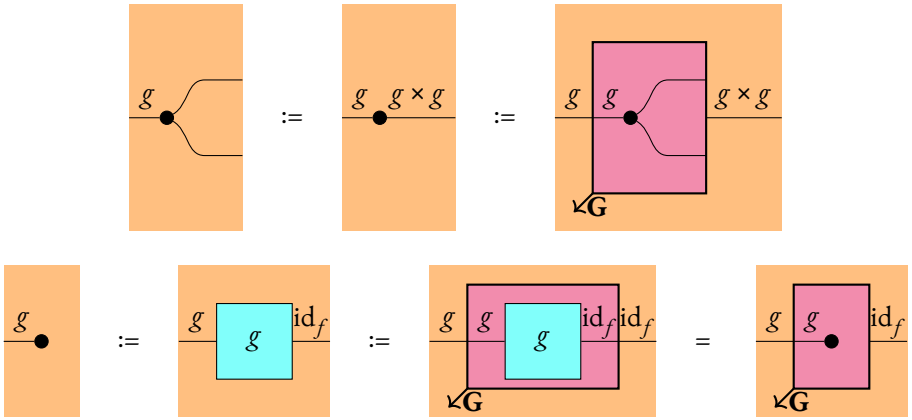
This method of proving that the slice of each topos is finitely complete is unique.

By Proposition 2.1.16, we can see that

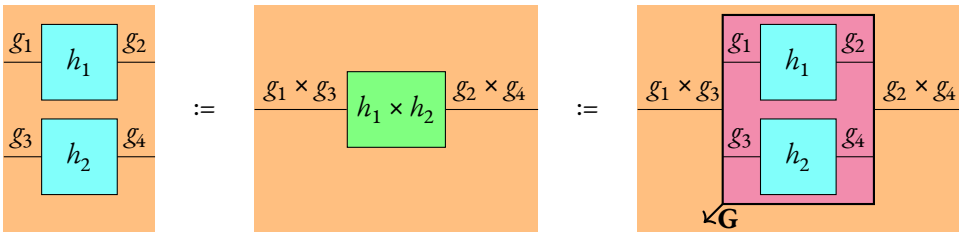
Lemma 3.1.1: Slices of Topoi are Finitely Complete

When $\mathcal{C}$ is a topos, every slice category $\mathcal{C}/A$ of $\mathcal{C}$ if finitely complete.

Proof: consider a slice category $\mathcal{C}/B$ of $\mathcal{C}$. We need to show that its slice categories all have finite products, by Corollary 2.1.7. But if we consider a slice category $(\mathcal{C}/B)/f$ of $\mathcal{C}/B$ with $f : A \to B$ in $\mathcal{C}$, then we see that $\mathcal{C}/A \cong (\mathcal{C}/B)/f$ by Proposition 2.1.16. But we already know that $\mathcal{C}/A$ has all finite products, as $\mathcal{C}$ is a topos, so $(\mathcal{C}/B)/f$ must have all finite products too. Graphically, this amounts to showing that, with $\mathbf{G} : \mathcal{C}/A \to (\mathcal{C}/B)/f$, from Proposition 2.1.16, that defining



and



tells us that eqs. (2.1) to (2.7) are satisfied, as per Fox's Theorem (Theorem 2.1.1). This is easy to see, and follows by simple compositionality of functor boxes. ∎

Now we can examine sub-object classifiers in slice categories.

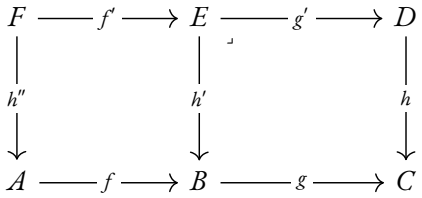
3.1.2 Sub-object Classifier

In order to prove that each slice of a topos has a sub-object classifier, we need a way to "colour change" the forgetful functors from slice category's boxes; so that's what we will prove first. This method is a unique, to my knowledge, way of proving that each slice of a topos has a sub-object classifier, which arises from staying within string-diagrammatic constraints.

Recall

Lemma 3.1.2: Pullback Lemma

Let $\mathcal{C}$ be a category. Then suppose the following commutes, and the right-hand square is a pullback

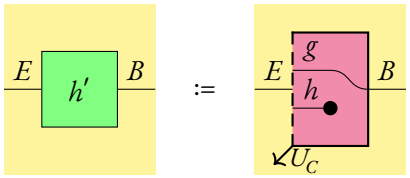


then the left-hand square is a pullback if and only if the outer rectangle is.

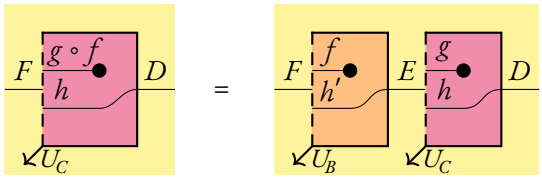
Proof: see [pwik:pl] or [Gol84, p. 67]. ∎

Proposition 3.1.3: Pullback Lemma, Graphically

Define

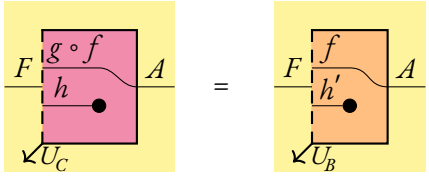


then if either the pullback of f and h', or the pullback of $g \circ f$ and h exist in $\mathcal{C}$, we have



(3.1)

and



(3.2)

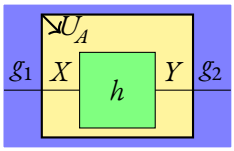
Proof: using Lemma 3.1.2, the existence of h' ensures that the pullback of g and h exists. Moreover, if either the pullback of f and h' or $g \circ f$ and h exist, then both exist, by Lemma 3.1.2. Hence, we see that $\mathbb{D}_2^{g,h}$ is the left-hand side of Equation 3.1, and $g' \circ f'$ is the right-hand side, demonstrating the equality. Similarly, we see that $\mathbb{D}_1^{g,h}$ is the left-hand side of Equation 3.2, and h'' is the right-hand side,

again, demonstrating the equality. ■

Lemma 3.1.4: Slices of Topoi have Sub-object Classifiers

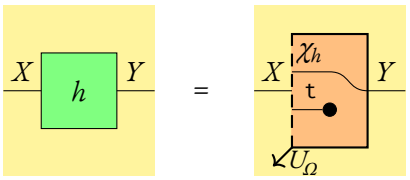
Each slice $\mathcal{C}/A$ of a topos $\mathcal{C}$ has a sub-object classifier.

Proof: consider the slice $\mathcal{C}/A$ of $\mathcal{C}$ and a monomorphism (in $\mathcal{C}/A$)



I will show that the sub-object classifier in $\mathcal{C}/A$ is $\pi_2^{\Omega\times A}$, and the truth morphism is $\mathtt{t}\times \mathrm{id}_A$ (which we know exists by previous subsection).

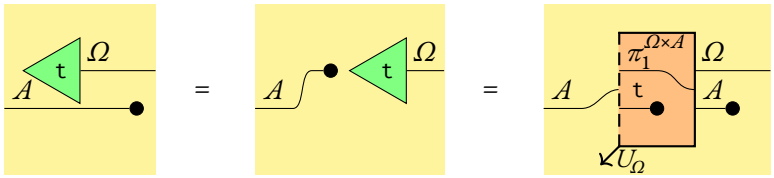
By Proposition 2.1.15, as U_A is a discrete fibration, $h : X \to Y$ must be monic in $\mathcal{C}$. Therefore, there must be a classifying morphism χ_h such that



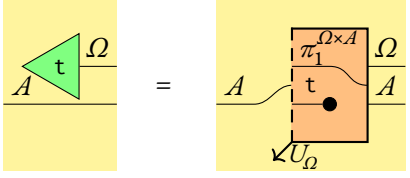
with colouring $\mathcal{C}/\Omega$.

Then, we can use the Graphical Pullback Lemma to show that h can be colour-changed from being a morphism in the slice over $\mathcal{C}/\Omega$ to the slice over $\mathcal{C}/(\Omega\times A)$, which then tells us that $\mathcal{C}/A$ has a sub-object classifier via the isomorphism $\mathcal{C}/(\Omega\times A) \cong (\mathcal{C}/A)/\pi_2^{\Omega\times A}$.

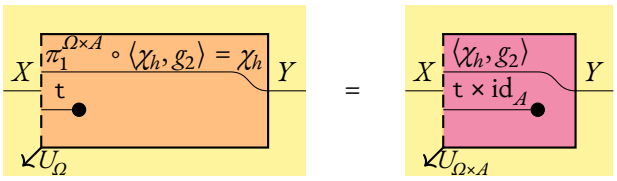
In order to the Graphical Pullback Lemma, first note that



by the definition of what it means to classify a morphism, and then as the projection map is monic, we conclude that

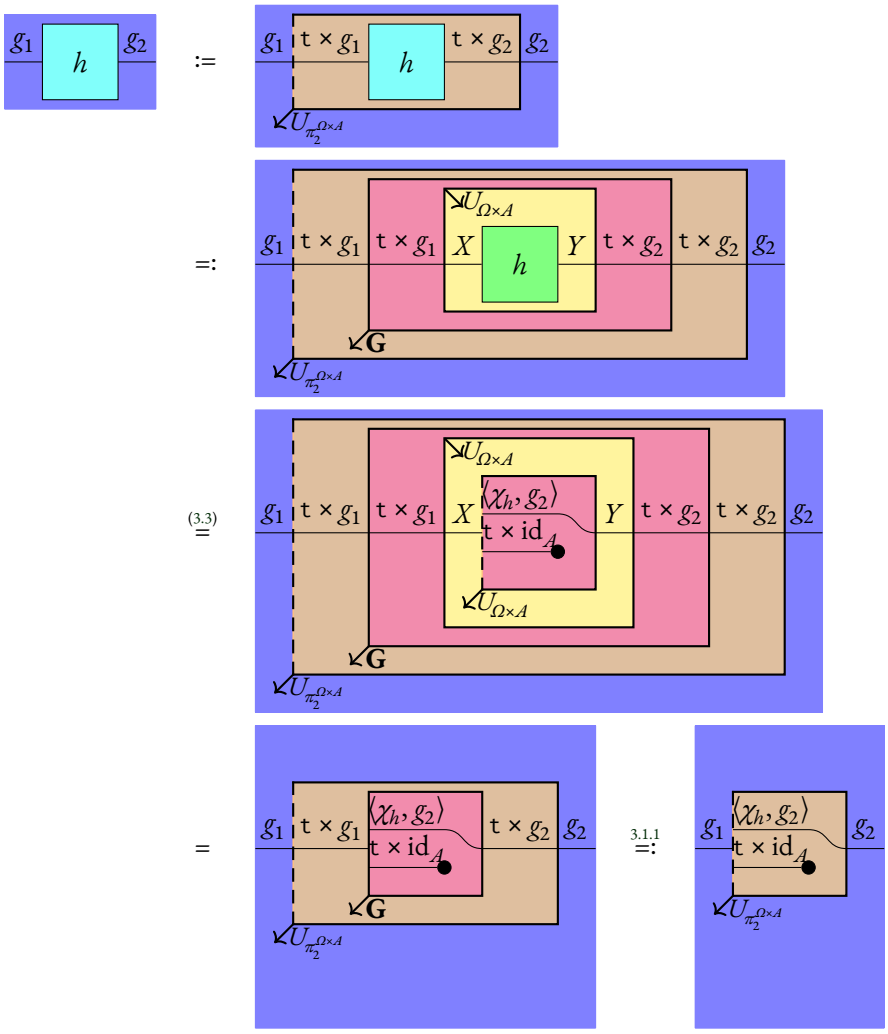


and so, by Proposition 3.1.3 The Graphical Pullback Lemma,



(3.3)

Hence, via the isomorphism $\mathbf{G} : \mathcal{C}/(\Omega \times A) \cong (\mathcal{C}/A)/\pi_2^{\Omega\times A}$ given in Proposition 2.1.16, $\mathcal{C}/A$ must have a sub-object classifier, $\pi_2^{\Omega\times A}$, with truth morphism $\mathtt{t} \times \mathrm{id}_A$. Explicitly, we can see this as follows



■

3.1.3 Exponentials

Lemma 3.1.5

Let $\mathcal{C}$ be a topos, then each slice category $\mathcal{C}/A$ has exponentials.

Proof: omitted. ■

The proof is omitted here because the use of string diagrams does not add anything. It is perfectly possible

to encode, for example, the proof in [MM94], in string diagrams. But what ends up happening is still a diagram chase after all.

This is certainly a place for future work. I conjecture that if string diagrams are to aid the proof in any way, then a more explicit construction such as [Joh14, Theorem 1.42] would perhaps be more useful. But, this proof would currently go too far afield, using partial morphisms.

With this Lemma, we can conclude

Theorem 3.1.6: The Fundamental Theorem of Topos Theory, Part I

Let $\mathcal{C}$ be a topos, then each slice $\mathcal{C}/A$ is also a topos.

Next, we will prove part II.

3.2 Pullback Functor has Left and Right Adjoints

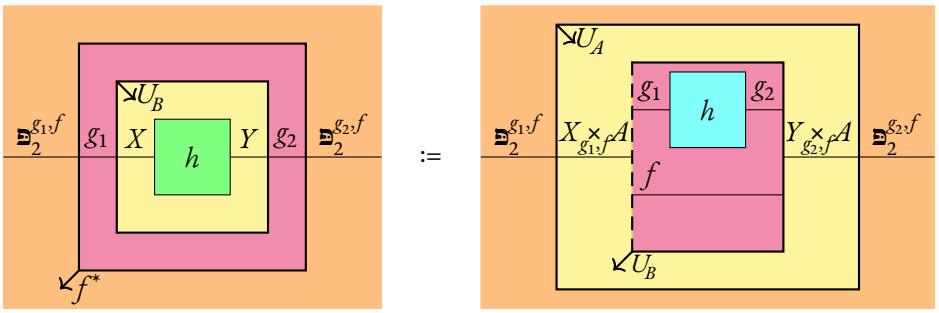
In this section, we will see the second part of the Fundamental Theorem. It asserts that a certain functor, the pullback functor, introduced next, has both left and right adjoints. The constructions of these two functors, are, in my opinion, much simpler than those in the literature, owing to the string diagrammatic descriptions. This is particularly true in the case of the right adjoint – the dependent product functor. I strongly recommend the reader compares the constructions here with those in [MM94, p. 193, Theorem 2].

3.2.1 Pullback Functor

This is the functor which given $f : A \to B$, intuitively takes an object g_1 of a slice category over B to the pullback with f.

Definition 3.2.1: Pullback Functor

Let $f : A \to B$ in a topos $\mathcal{C}$, then this induces a functor $f^* : \mathcal{C}/B \to \mathcal{C}/A$.

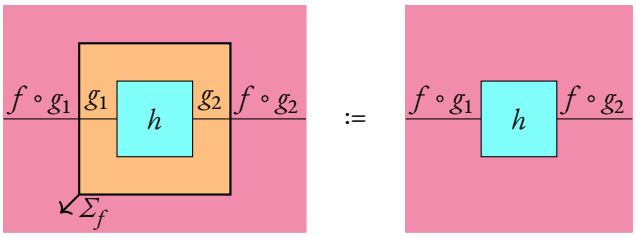


3.2.2 Dependent Sum Functor

First, we shall see the left adjoint, which is known as the "dependent sum" functor.

Definition 3.2.2: Dependent Sum Functor

Given $f : A \to B$ in a topos $\mathcal{C}$, we have $\Sigma_f : \mathcal{C}/A \to \mathcal{C}/B$.

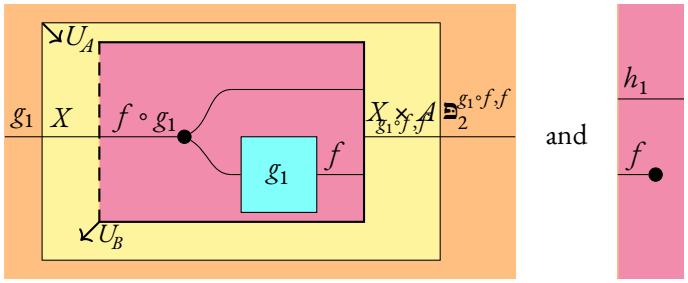


As we can see, this intuitively just lifts a morphism from $\mathcal{C}/A$ to $\mathcal{C}/B$.

Lemma 3.2.3

Let $\mathcal{C}$ be a topos with morphism $f : A \to B$. Then $\Sigma_f \dashv f^*$.

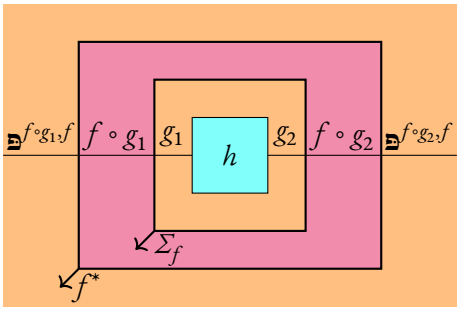
Proof: fix a topos $\mathcal{C}$, $f : A \to B$ in $\mathcal{C}$, and slice categories $\mathcal{C}/A$ and $\mathcal{C}/B$. Then the claim is that (given $g_1 : X \to A$ and $h_1 : Y \to B$)



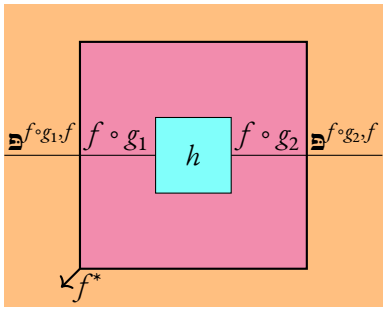
and

are the unit and counit of the adjunction, respectively.

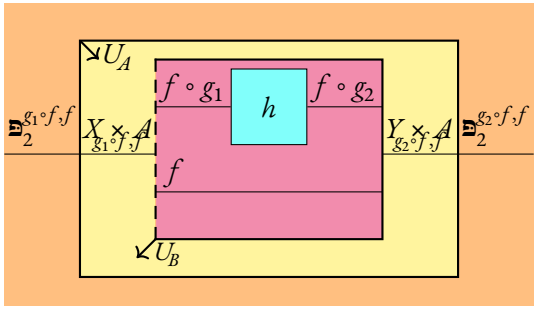
To see this, first note that



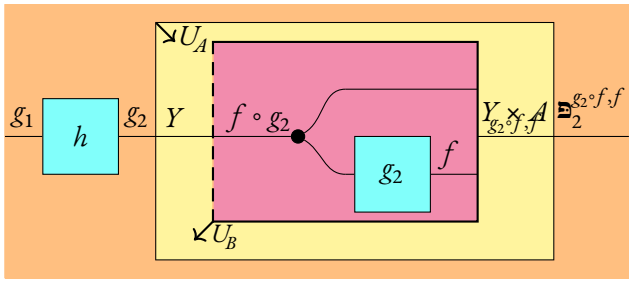
:=



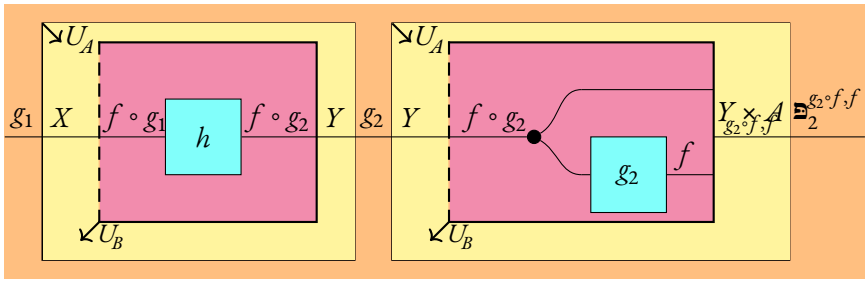
:=



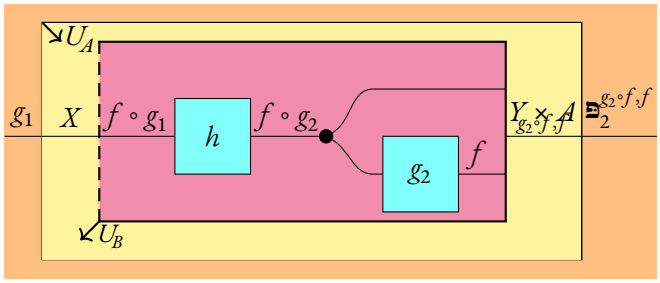
so we can see



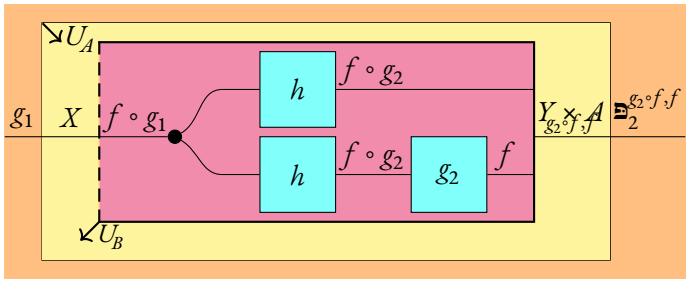
$=:$



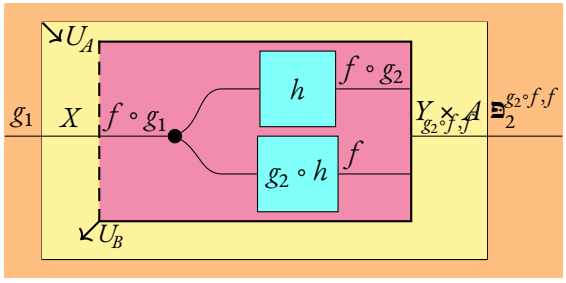
$\overset{1.3.4}{=}$



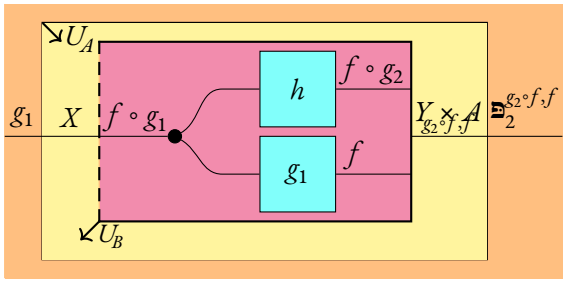
$\overset{(2.4)}{=}$



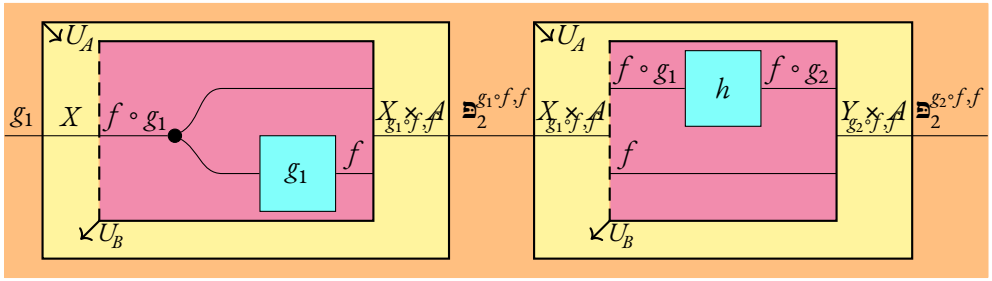
$\overset{1.3.4}{=}$



$=$

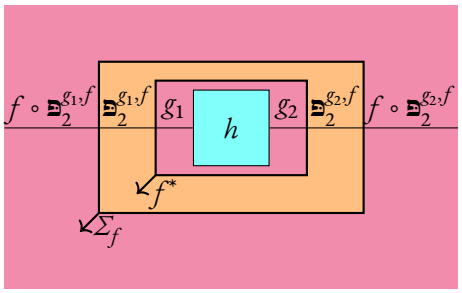


$\overset{1,3,4}{=}$

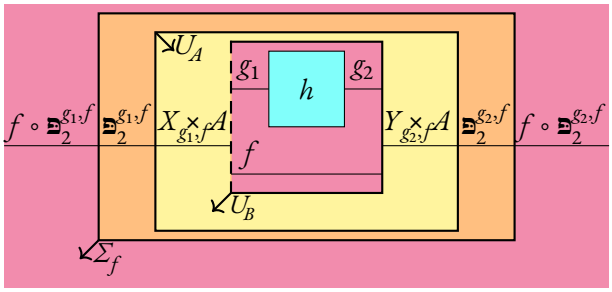


Establishing the first required identity for the adjunction.

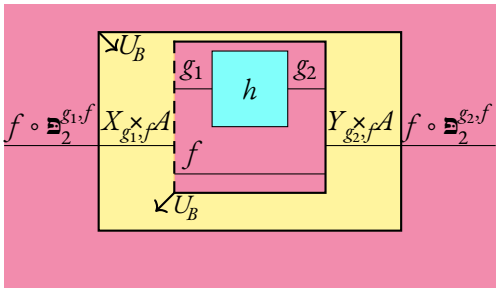
For the second, note that



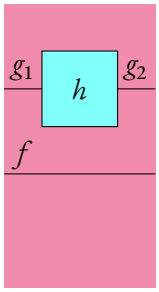
$:=$



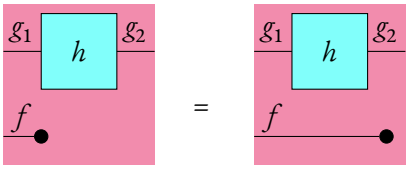
$:=$



$=$

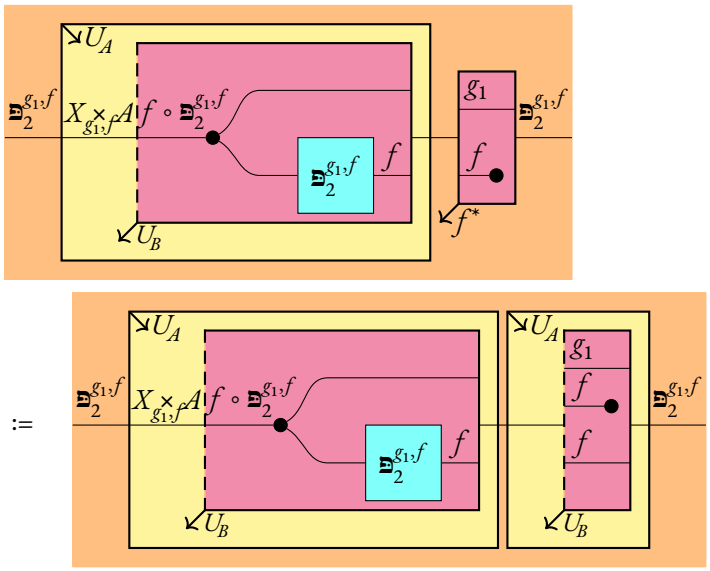


and then we have

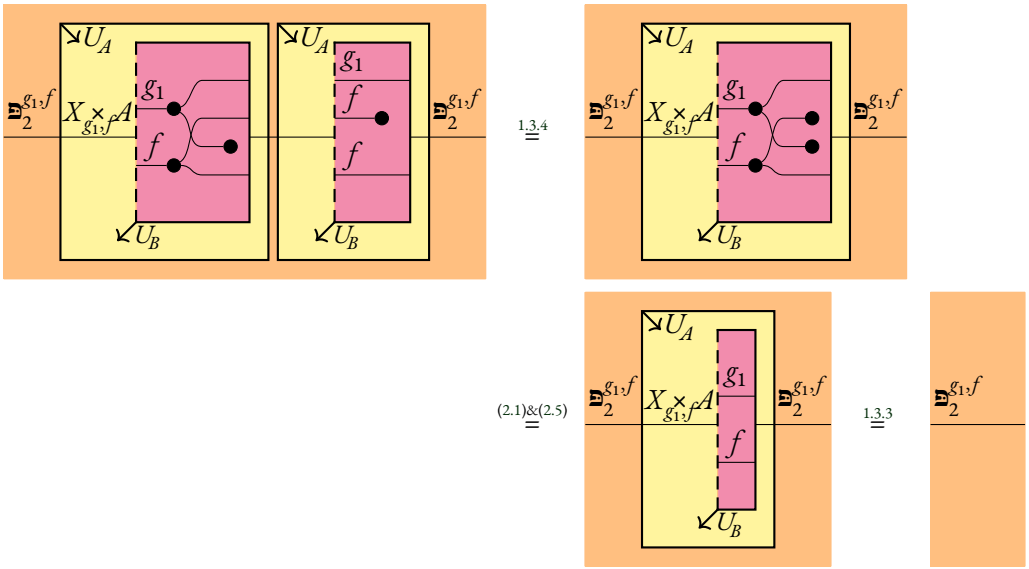


demonstrating the second required equality for the adjunction.

For the third, we have

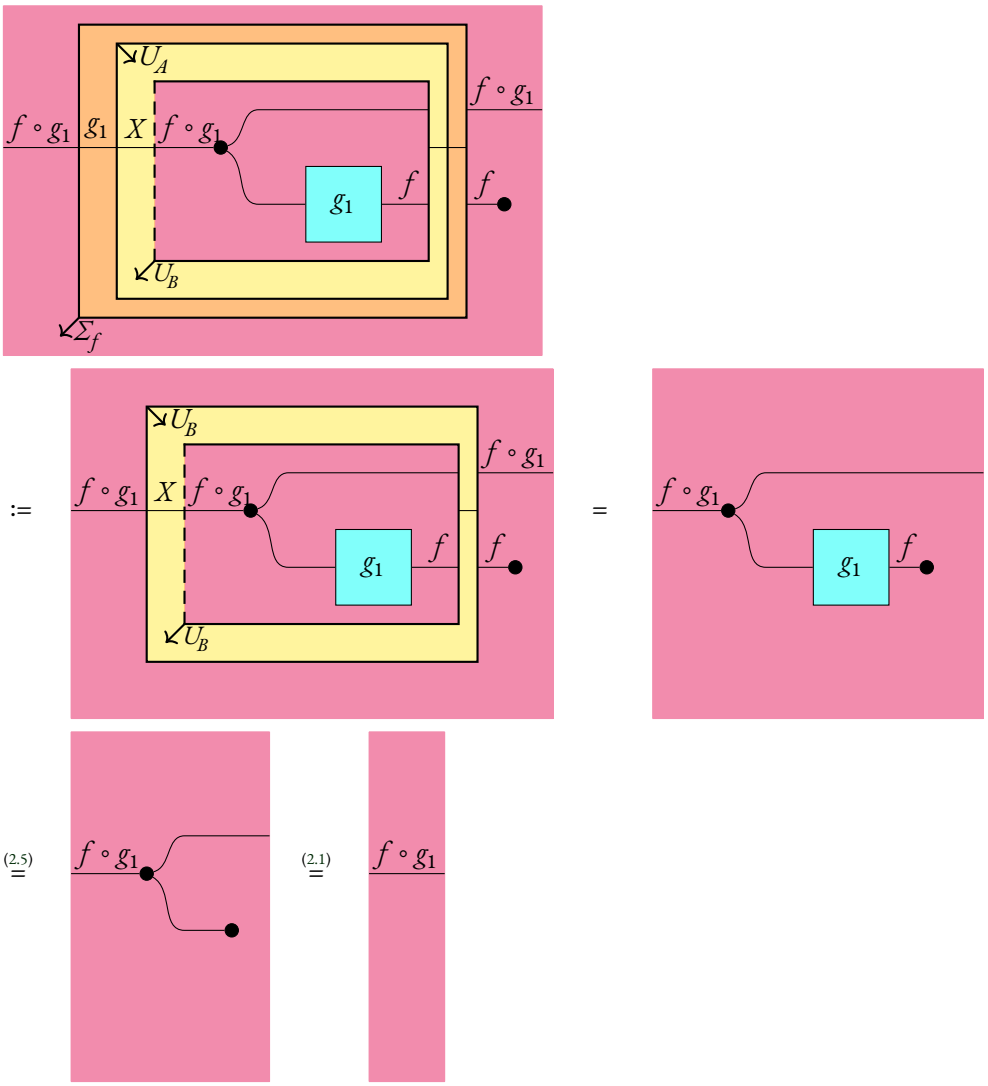


which, by expanding the definition of $\exists_2^{g_1,f}$, is equal to



which demonstrates the third required equality.

Finally,



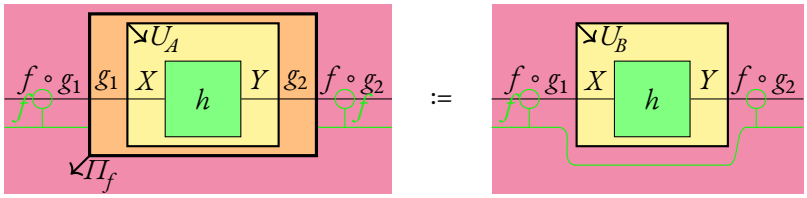
demonstrating the last required equality. ∎

3.2.3 Dependent Product Functor

Now we can see the right adjoint, the "dependent product" functor.

Definition 3.2.4: Dependent Product Functor

Given $f : A \to B$ in a topos $\mathcal{C}$, we have $\Pi f : \mathcal{C}/A \to \mathcal{C}/B$.

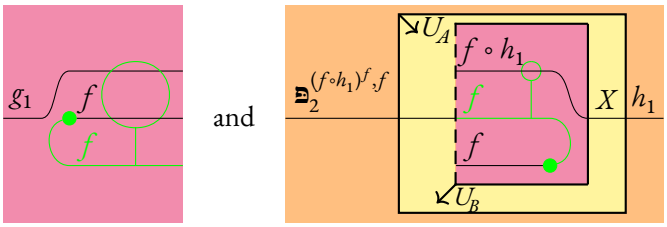


Again, this functor intuitively just lifts each morphism in $\mathcal{C}/A$ to one in $\mathcal{C}/B$.

Lemma 3.2.5

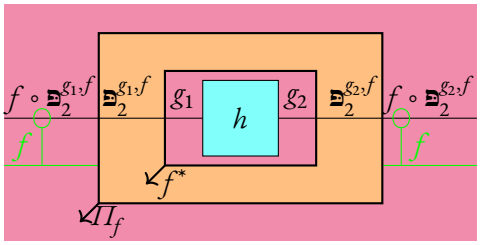
Let $\mathcal{C}$ be a topos with morphism $f : A \to B$. Then $f^* \dashv \Pi_f$.

Proof: fix a topos $\mathcal{C}$, $f : A \to B$ in $\mathcal{C}$, and slice categories $\mathcal{C}/A$ and $\mathcal{C}/B$. Then the claim is that (given $g_1 : Y \to B$ and $h_1 : X \to A$)

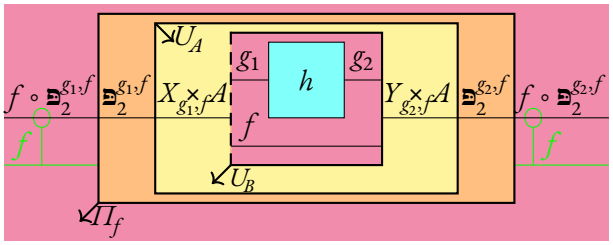


are the unit and counit of the adjunction, respectively.

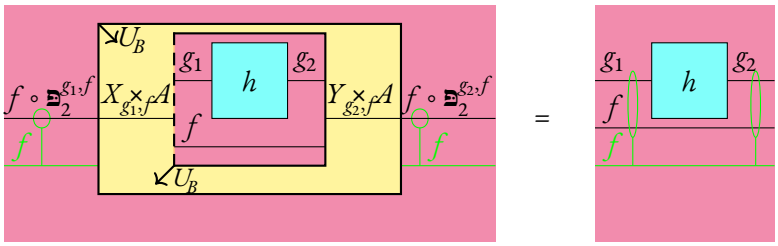
To see this, first note that



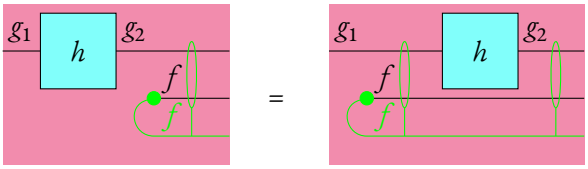
$:=$



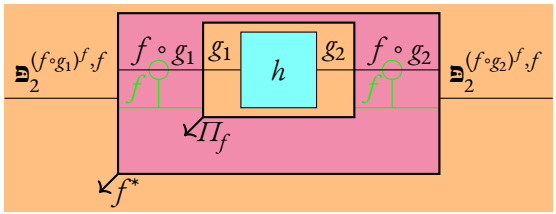
$:=$



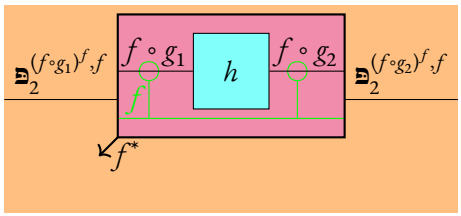
and then the first required equality to demonstrate the adjunction (showing that the unit is natural transformation) is immediately obvious:



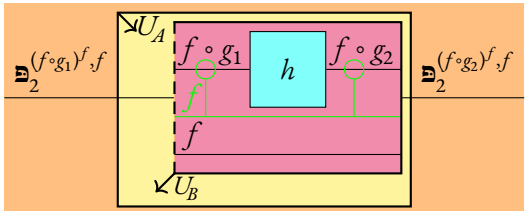
For the second, first note that



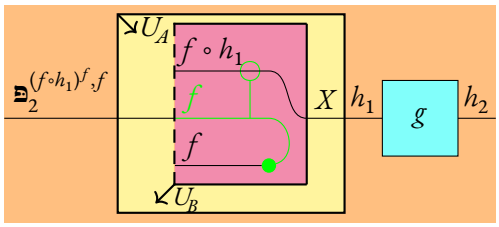
$:=$



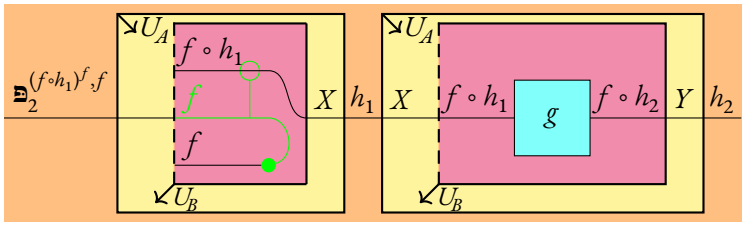
$:=$



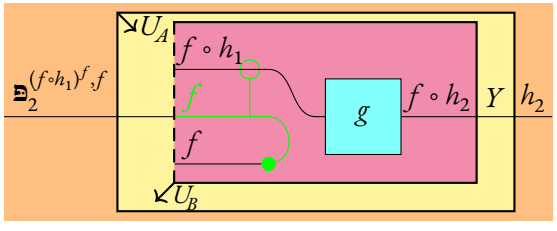
then we have



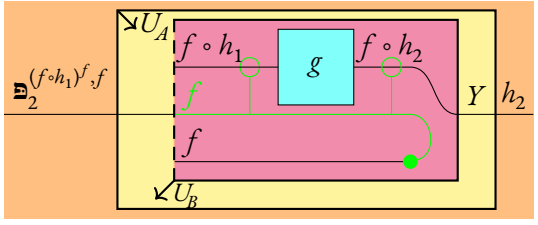
$=$



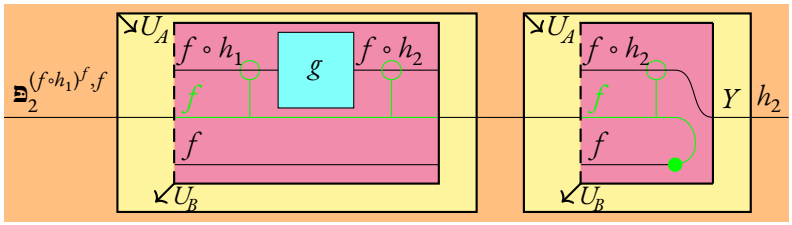
$\overset{(1.6)}{=}$



$=$

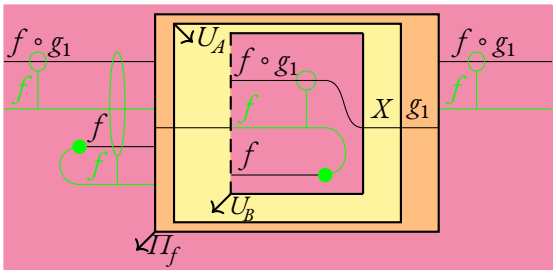


$\overset{(1.6)}{=}$

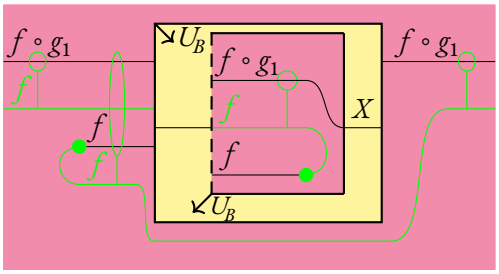


demonstrating the second required equality.

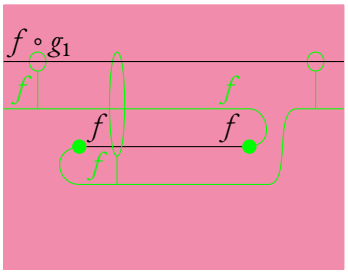
For the third,



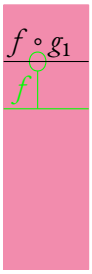
$:=$



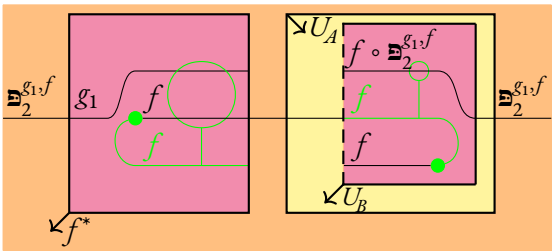
$=$



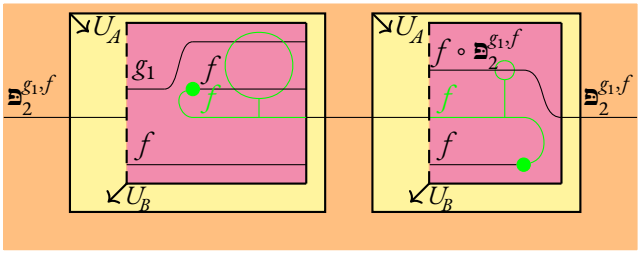
$\overset{(2.20)}{=}$



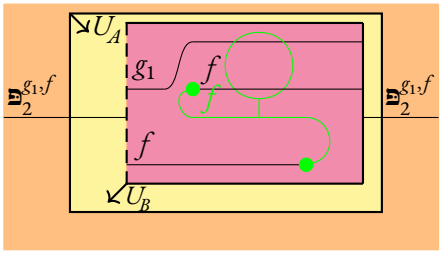
And, finally, for the fourth,



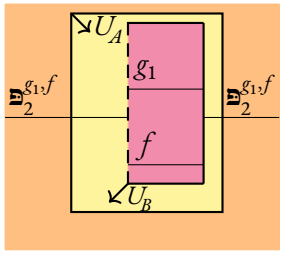
:=



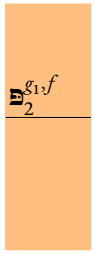
$\overset{(1.6)}{=}$



$\overset{(2.21)}{=}$



$\overset{1.3.3}{=}$



completing the proof. ∎

So, we can now conclude the full Fundamental Theorem of Topos Theory

Theorem 3.2.6: The Fundamental Theorem of Topos Theory

If $\mathcal{C}$ is a topos, then each slice $\mathcal{C}/A$ is also a topos, and there exists a sequence

$$\Sigma_f \dashv f^* \dashv \Pi_f$$

of adjunctions for each $f : A \to B$ in $\mathcal{C}$.

Chapter 4

Categorical Logic

Contents

In any category we can define a type theory, and in particular, in a topos, we can define a logic over that type theory. We will see what this general type theory looks like in cartesian-closed categories (hint: the typed lambda-calculus), and describe the internal logic over the type theory in topoi. Once we've introduced this, we will introduce some nice syntactic sugar into the diagrams, to have a final diagram set, which corresponds to reasoning in higher-order intuitionistic logic. Here, a predicate is any morphism with codomain Ω.

The "higher-order" part of the logic means that we can quantify over variables of every type – including propositions, which is in contrast to first-order logic, where only quantification over objects, and not formulæ/propositions, is allowed.

This section is largely based on [Str].

4.1 Internal Type Theory

Given $f : B \to A$ in a cartesian closed category (and so in particular a topos), we may interpret this as a term $f(x)$ of type A, where x is a free variable of type B. Symbolically, $x : B \vdash f(x) : A$, we write for the *judgement* "given x is of type B, $f(x)$ is of type A". Extending this idea, in any cartesian closed category, we may interpret a typed lambda calculus. We shall see this construction now.

4.1.1 Type Semantics

Fix $\mathbb{B}$ a collection of *base types*. Then we have $\mathbb{T}_{\mathbb{B}}$, the collection of types, generated by the following rules:

- every base type is a type;
- if A and B are types, then so is $A \to B$;
- 1 is a type;
- if A and B are types, then so is $A \times B$; and
- nothing else is a type.

A *context* is an expression of the form $x_1 : A_1, \ldots, x_n : A_n$, where n is a natural number, each A_i is a type, and the variables x_i are pairwise distinct.

To interpret a type theory in a closed cartesian category $\mathcal{C}$, we simply fix an assignment $[\![\cdot]\!] : \mathbb{B} \to \mathrm{Ob}\,\mathcal{C}$ on the base types of the theory.

This is then extended naturally as follows into an assignment from all types into the objects of $\mathcal{C}$:

- $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$;
- $[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$; and
- $[\![1]\!] := 1$, the terminal object in $\mathcal{C}$.

Moreover, we interpret a context as follows:

$$[\![x_1 : A_1, \ldots, x_n : A_n]\!] := [\![A_1]\!] \times \cdots \times [\![A_n]\!].$$

We write $\Gamma \vdash t : A$ to indicate that t is a *term* of type A in context Γ. We interpret this judgement as being true if and only if there exists a morphism $[\![\Gamma \vdash t : A]\!] : [\![\Gamma]\!] \to [\![A]\!]$ in $\mathcal{C}$.

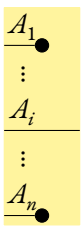
Now we can see the calculus on these types in a given context, which corresponds to the existence of certain morphisms in any cartesian closed category. This tells us how we can construct new terms in any cartesian closed category – i.e., term formation rules.

4.1.2 Term Formation Rules

The rule

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\ (\mathrm{Var})$$

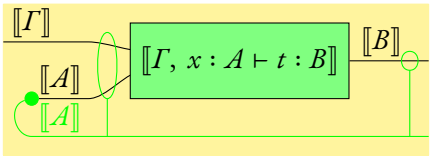
is interpreted as constructing the morphism



The rule

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\lambda x : A.t) : A \to B}\ (\lambda)$$

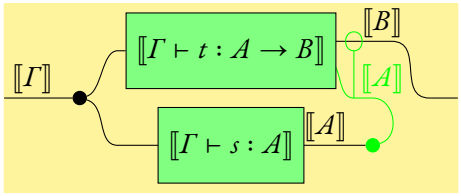
is interpreted as constructing the morphism



The rule

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash t(s) : B}\ (\text{App})$$

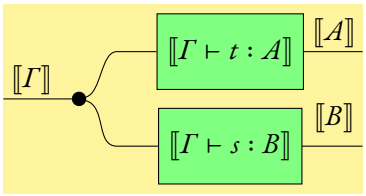
is interpreted as constructing the morphism



The rule

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle t, s \rangle : A \times B}\ (\text{Pair})$$

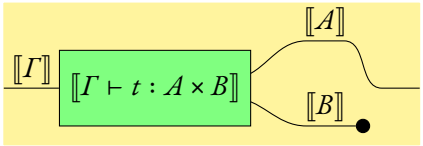
is interpreted as constructing the morphism



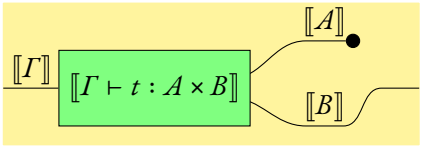
The pair of rules

$$\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_1(t) : A}\ (\text{Proj}_1) \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_2(t) : B}\ (\text{Proj}_2)$$

are interpreted as constructing the morphisms



and

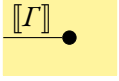


respectively.

Finally, the rule

$$\frac{}{\Gamma \vdash * : 1}\ (\text{Unit})$$

is interpreted as constructing the morphism



It can be seen, e.g., in [Str], or graphically in [GZ23] that this is sound and complete with respect to the typed lambda calculus, with η equivalence.

4.2 Logical Connectives in a Topos

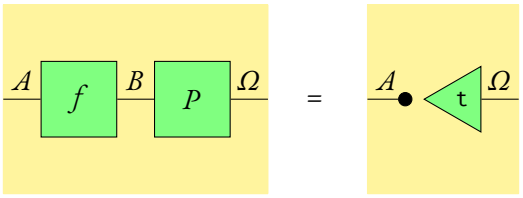
From here on, fix a topos $\mathcal{C}$.

Remark 4.2.1. I will henceforth write $\mathtt{t}_A$ for $\mathtt{t} \circ !_A$.

The following Lemma tells us precisely when a predicate holds of a morphism.

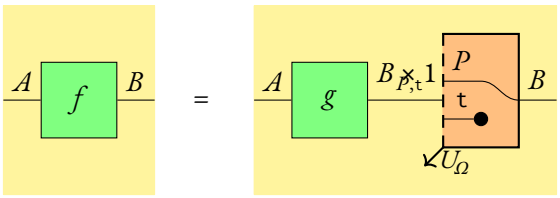
Lemma 4.2.2

This is in any finitely complete category with sub-object classifiers.



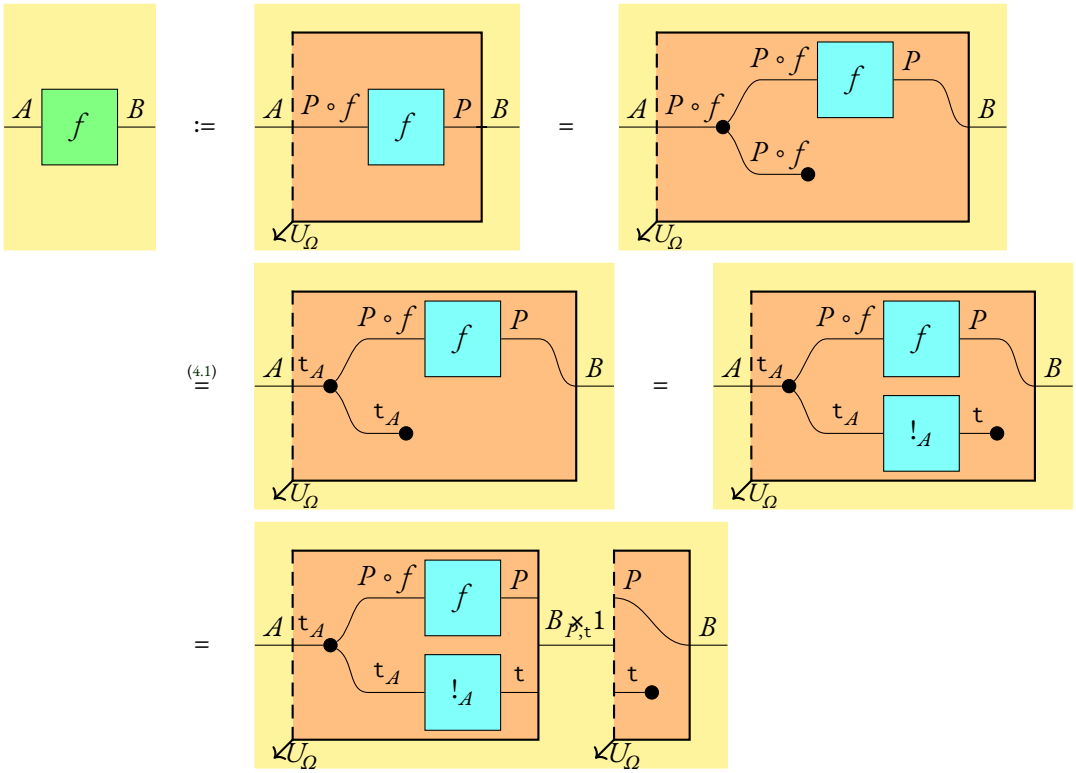
(4.1)

if and only if there exists a $g : A \to B \times_{P,\mathtt{t}} 1$ such that

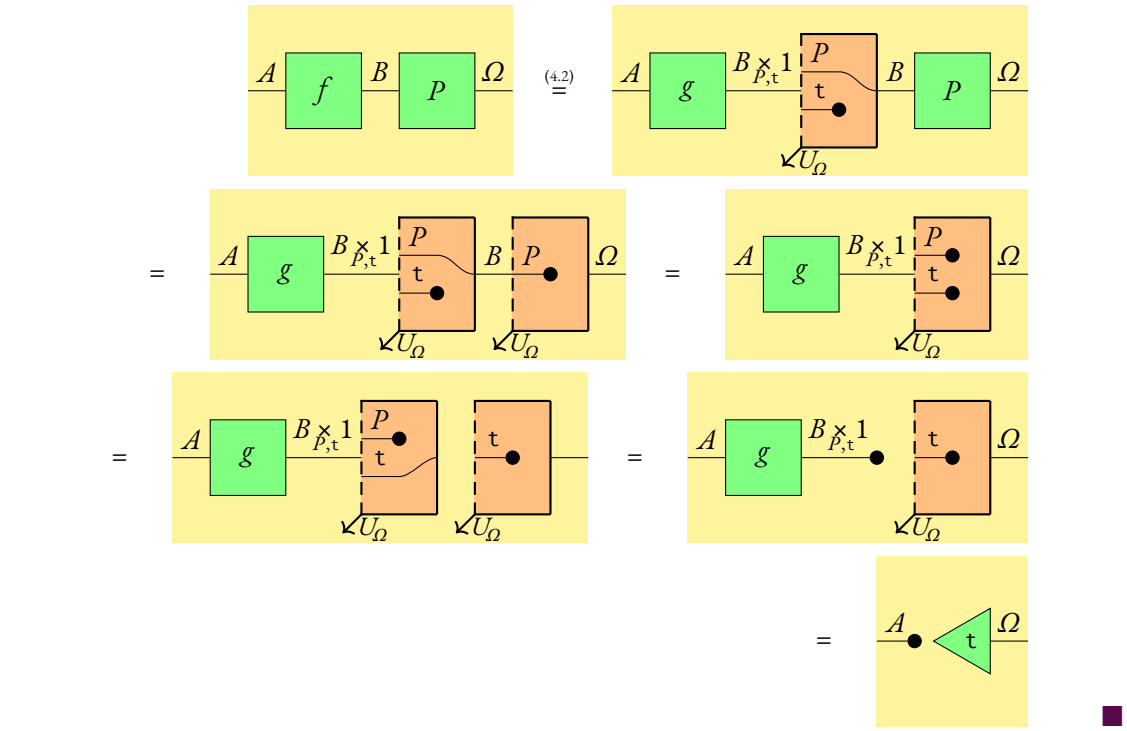


(4.2)

Proof: ($\Longrightarrow$)



($\Longleftarrow$)



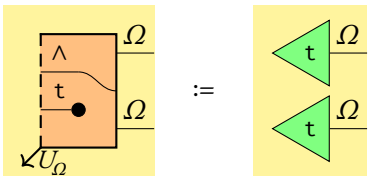
■

Using this, we can introduce the familiar logical operations.

4.2.1 Conjunction

First up, is conjunction. As the right-hand side of the below equation is (trivially) monic, we can define a morphism uniquely as its subobject classifier, this is $\wedge$.

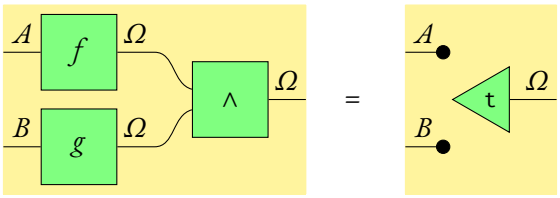
Definition 4.2.3



(4.3)

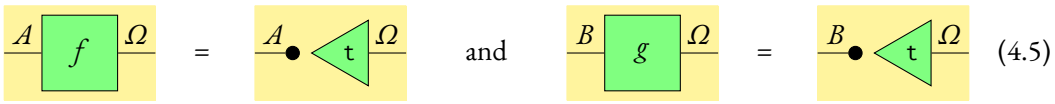
Its truth conditions are expressed as expected.

Proposition 4.2.4



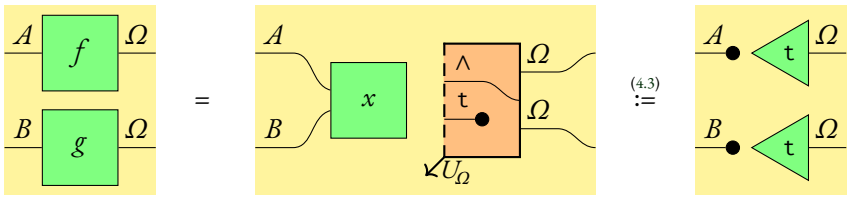
(4.4)

if and only if



(4.5)

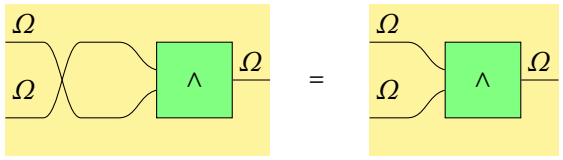
Proof: by Lemma 4.2.2, Equation 4.4 holds if and only if there exists a morphism x such that



which in turn holds if and only if Equation 4.5 holds. ∎

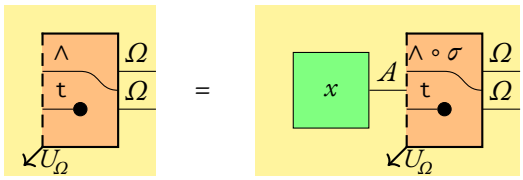
We can now prove some (nice) properties of $\wedge$. We can show it is a commutative monoid with respect to the (strict) product structure on each topos.

Lemma 4.2.5

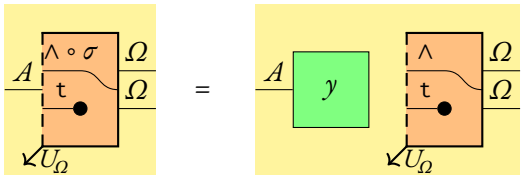


(4.6)

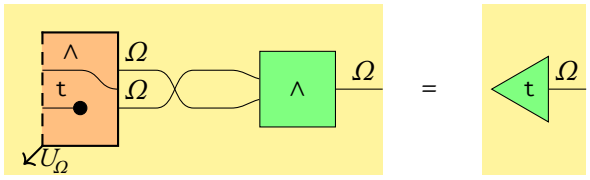
Proof: Equation 4.6 holds if and only if there exists maps x and y such that



and

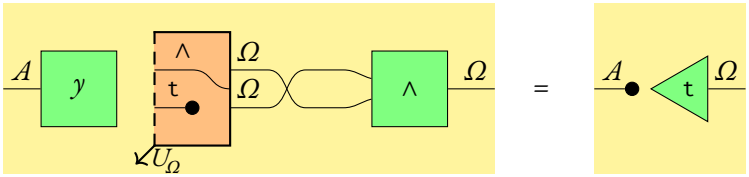


and, by Lemma 4.2.2, these hold if and only if



(4.7)

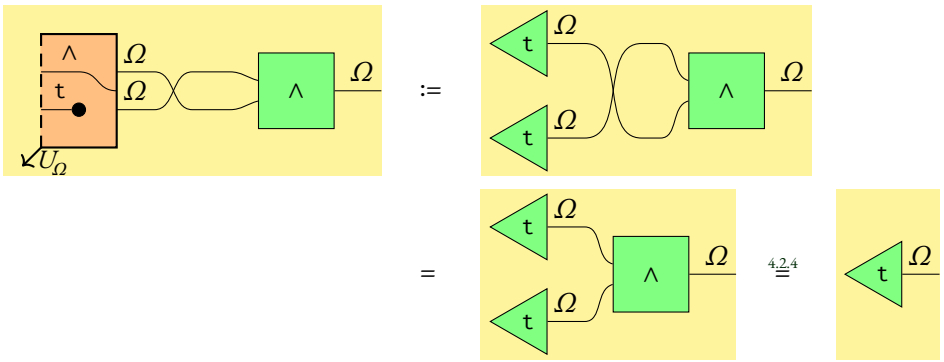
and



(4.8)

hold.

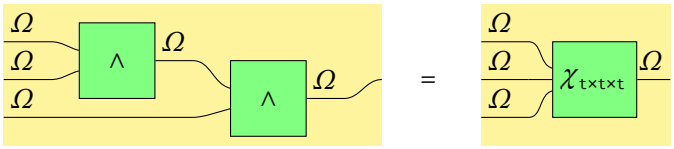
But as



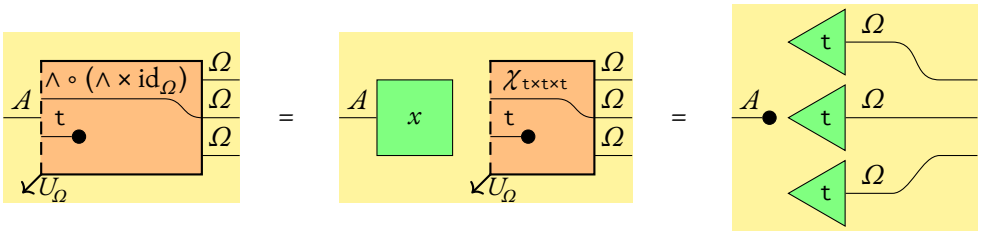
both Equation 4.7 and Equation 4.8 must hold. ■

Now I will show that ∧ is associative, which is done first by seeing the following.

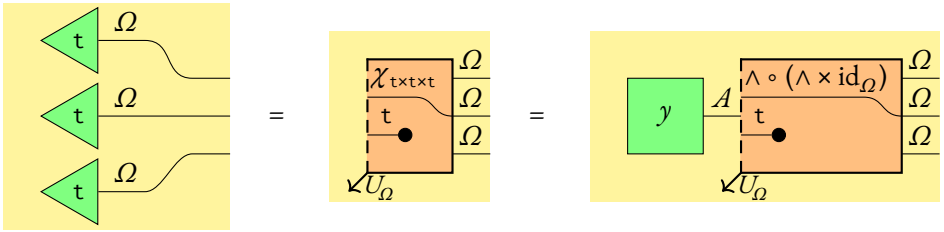
Lemma 4.2.6



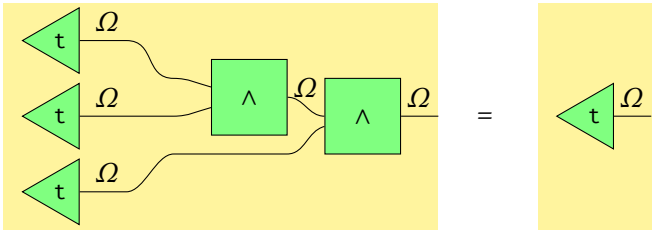
Proof: this is true if and only if there exist x and y such that



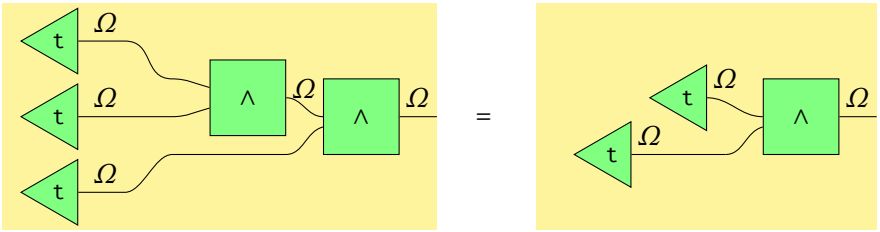
and

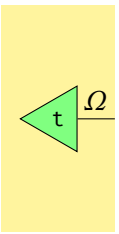


By Lemma 4.2.2, these both hold if and only if.



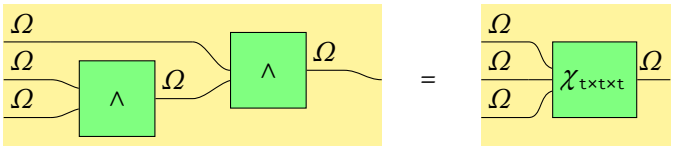
which we see because





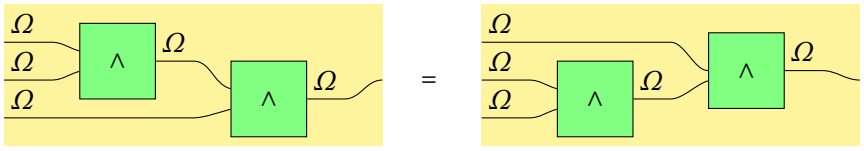
Which completes the proof. ■

The same reasoning shows us that



and so we conclude that

Proposition 4.2.7



Theorem 4.2.8: ∧ is a Monoid

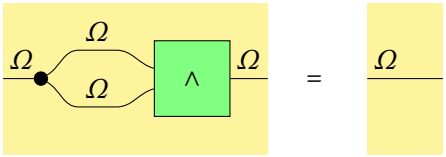
∧ is a symmetric monoid with monoidal unit t (with respect to the product structure).

Proof: the only thing left to prove is that t is the monoidal unit, which is similar to the above. ■

Finally, we can see how this interacts with the comonoidal structure on a topos induced by the product

structure.

Proposition 4.2.9



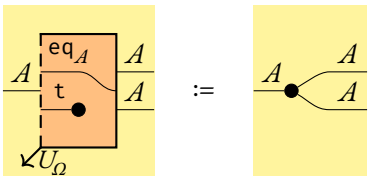
Proof: similar to everything else above. ∎

Remark 4.2.10. Given the results of this subsection, I will denote conjunction in a category as a small triangle, like the copy maps, but with only 1 output. These triangles can take in as many inputs as needed, and it will be unambiguous what is meant – the pairwise conjoining of two of the inputs, where the order does not matter. To reflect that `t` is the unit of the adjunction, I will draw it as the small triangle with no inputs, and a single output. The triangle with exactly one input and exactly one output is just the identity on Ω.

4.2.2 Implication

We will define implication in terms of conjunction and logical equality. So first we will see how to define logical equality.

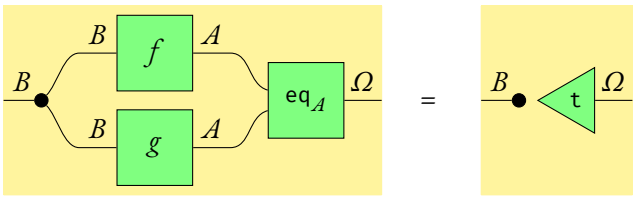
Definition 4.2.11



(4.9)

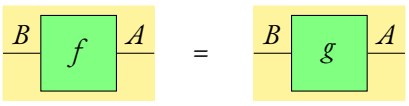
Remark 4.2.12. I write '$\Leftrightarrow$' for $\mathtt{eq}_\Omega$.

Proposition 4.2.13



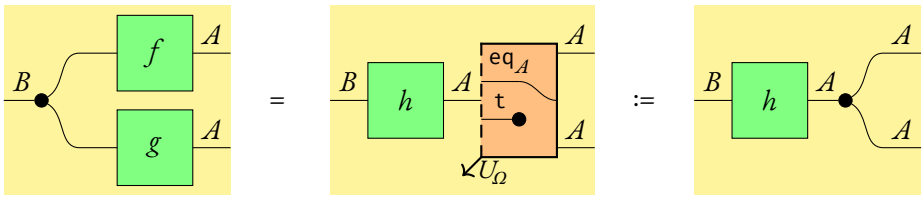
(4.10)

if and only if



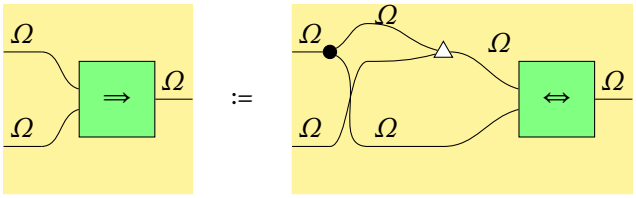
(4.11)

Proof: by Lemma 4.2.2, Equation 4.10 holds if and only if there exists an h such that



which holds, via the universal property for the product if and only if Equation 4.11 holds. ■

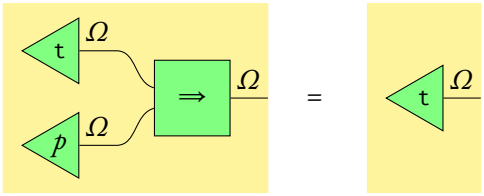
Definition 4.2.14



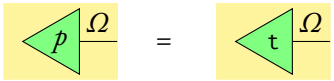
(4.12)

where I am using the white triangle to represent conjunction, as discussed earlier.

Theorem 4.2.15: Graphical Modus Ponens



implies that

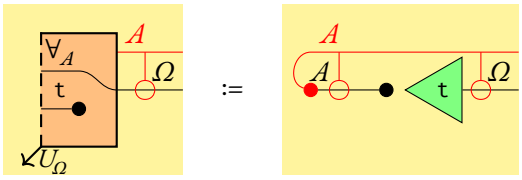


Proof: immediate from the definitions of $\wedge$ and $\Leftrightarrow$. ■

4.2.3 Universal Quantification

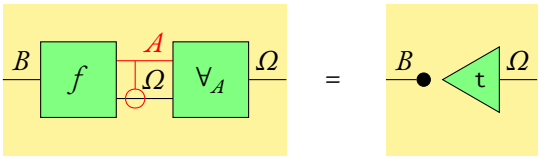
We have a different universal quantifier for every type in $\mathcal{C}$.

Definition 4.2.16: Universal Quantification

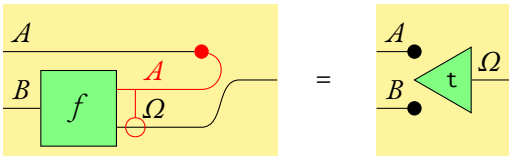


Then it has truth conditions as follows.

Proposition 4.2.17

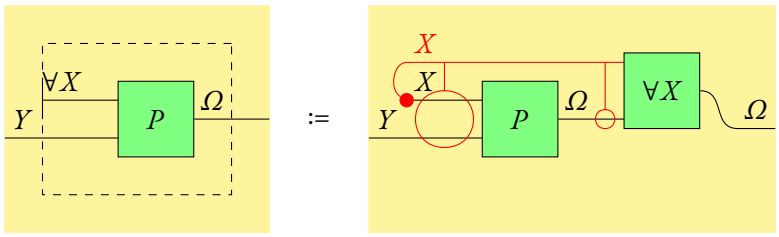


if and only if

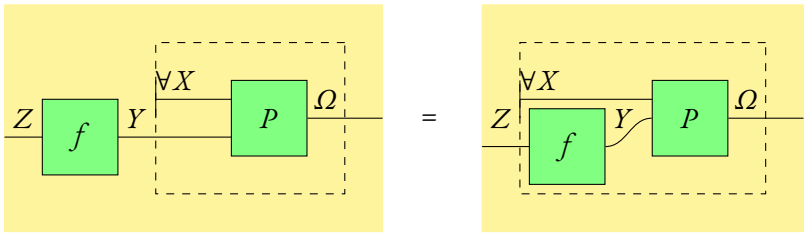


Proof: this is similar to the conditions for ∧ and eq. ■

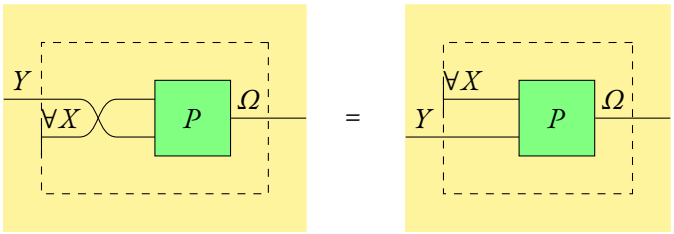
Finally, we can introduce some more notation, which allows us to abstract away with universal quantifiers, and treat them just like states. So, we define



This is coherent in the following sense, we see, trivially that



And also trivially that



With just these three logical operations, we can define every other logical operation in higher-order intuitionistic logic. The following definitions are what we need.

$$\mathsf{f} := \forall(\phi : \Omega).\phi$$

$$\phi \vee \psi := \forall(\chi : \Omega).((\phi \Rightarrow \chi) \wedge (\psi \Rightarrow \chi)) \Rightarrow \chi$$

$$\exists(x : A).\phi(x) := \forall(\psi : \Omega).(\forall(x : A).\phi(x) \Rightarrow \psi) \Rightarrow \psi$$

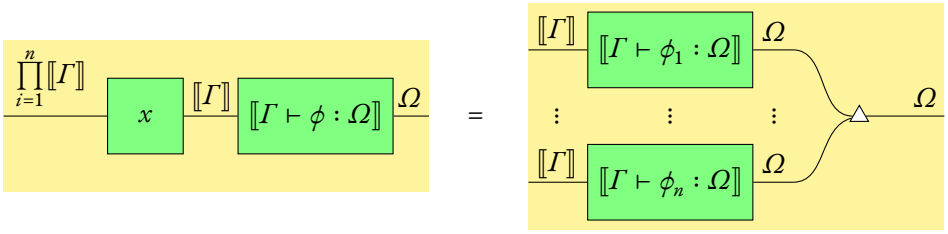
$$\neg\phi := \phi \Rightarrow \mathtt{f}$$

4.3 The Internal Logic of a Topos

We will now extend the type theory of section 4.1. Terms of type Ω in a context, are called the propositions of the context, and are special. In contrasts to the judgements of type theory, we now have sequents. If Γ is a context and $\Gamma \vdash \phi_i : \Omega$ for $i = 1, \dots, n$, and $\Gamma \vdash \phi : \Omega$, then we write

$$\Gamma \mid \phi_1, \dots, \phi_n \vdash \phi$$

if there exists an $x : \prod_{i=1}^{n} [\![\Gamma]\!] \to [\![\Gamma]\!]$ such that



and I will write Φ as a shorthand for $\phi_1, \dots, \phi_n$.

We will add some new term-forming rules (specific to Ω) which allow us to use the logical connectives that we saw in the previous section.

$$\frac{\Gamma \vdash \phi : \Omega \qquad \Gamma \vdash \psi : \Omega}{\Gamma \vdash \phi \wedge \psi : \Omega}\ (\wedge)$$

$$\frac{}{\Gamma \vdash \mathtt{t} : \Omega}\ (\mathtt{t})$$

$$\frac{\Gamma \vdash \phi : \Omega \qquad \Gamma \vdash \psi : \Omega}{\Gamma \vdash \phi \Rightarrow \psi : \Omega}\ (\Rightarrow)$$

$$\frac{\Gamma \vdash \phi : A \to \Omega}{\Gamma \vdash \forall(x : A).\phi(x) : \Omega}\ (\forall)$$

These correspond to the obvious morphisms in the topos.

With these, we can start to define the rules of the logic over the type theory. The first collection of rules tell us the valid structural rules of the logic, which are not of interest to us (although substructural logics are definitely of interest in general: see for example [Abe24] and [Res99]). As such, I will only give two examples of which morphisms these correspond to.

$$\frac{\sigma : \Delta \to \Gamma \qquad \Gamma \mid \Phi \vdash \phi}{\Delta \mid \Phi[\sigma] \vdash \phi[\sigma]}\ (\text{Subst})$$

$$\frac{\Gamma \mid \Phi \vdash \psi \qquad \Gamma \vdash \phi : \Omega}{\Gamma \mid \Phi, \phi \vdash \psi}\ (\text{Weak})$$

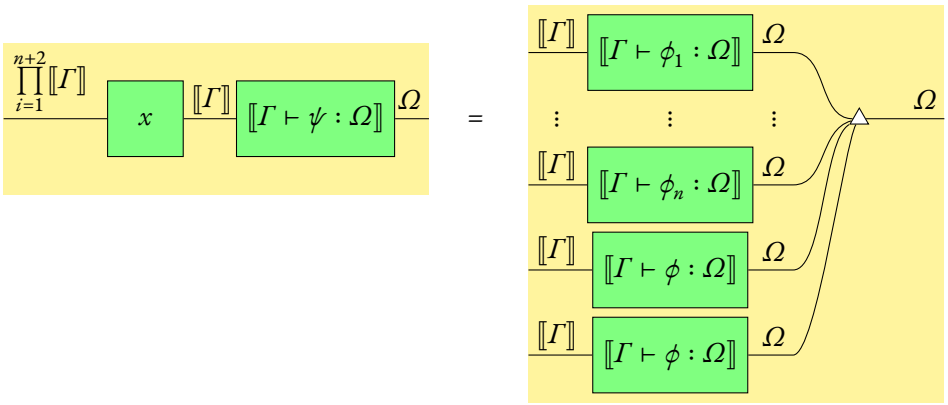
$$\frac{\Gamma \mid \Phi,\ \phi,\ \phi \vdash \psi}{\Gamma \mid \Phi,\ \phi \vdash \psi}\ (\text{Contr})$$

$$\frac{\Gamma \mid \Phi_1,\ \phi_1,\ \phi_2,\ \Phi_2 \vdash \psi}{\Gamma \mid \Phi_1,\ \phi_2,\ \phi_1,\ \Phi_2 \vdash \psi}\ (\text{Perm})$$

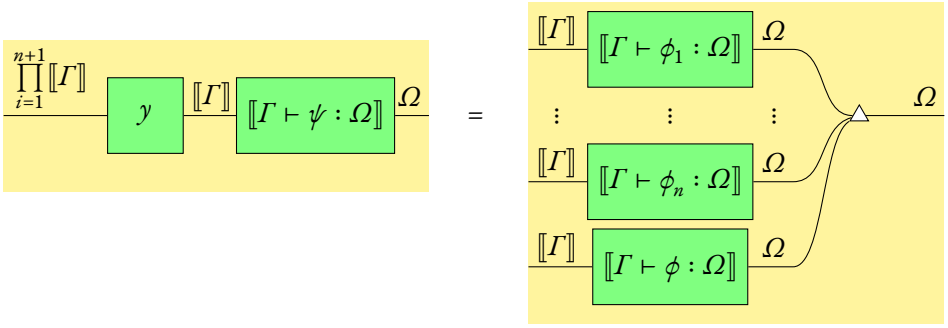
$$\frac{\Gamma \vdash \phi : \Omega}{\Gamma \mid \phi \vdash \phi}\ (\text{Ax})$$

$$\frac{\Gamma \mid \Phi \vdash \phi \qquad \Gamma \mid \Phi,\ \phi \vdash \psi}{\Gamma \mid \Phi \vdash \psi}\ (\text{Cut})$$

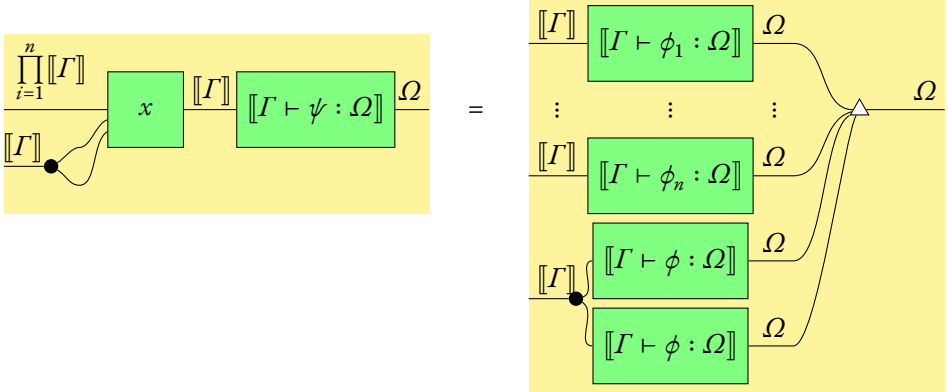
Let's see which morphisms the (Contr) and (Ax) rules correspond to. The (Contr) rule says that given that there exists an x such that



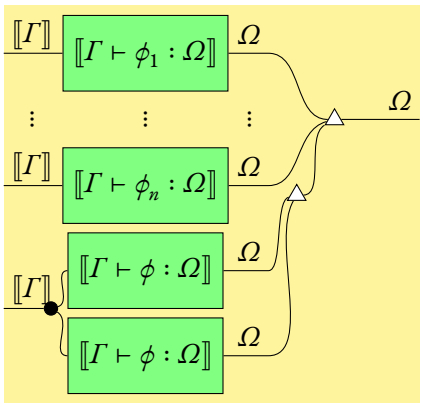
there must be a y such that



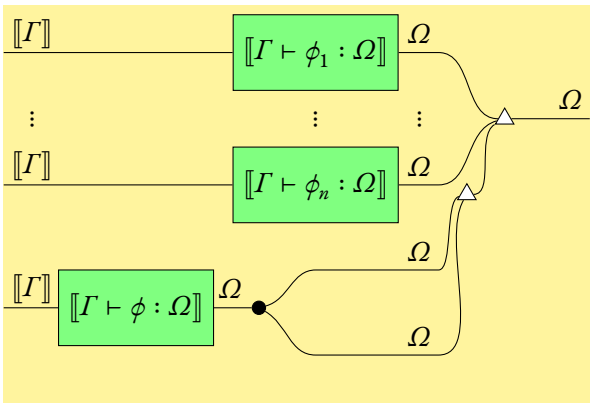
We see this is the case by



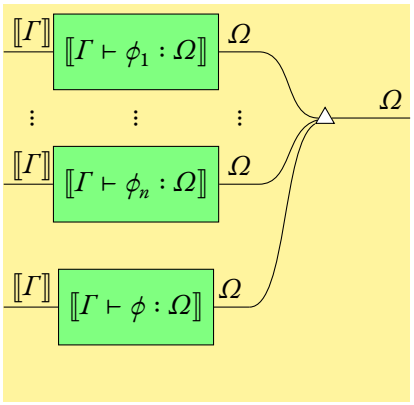
$=$



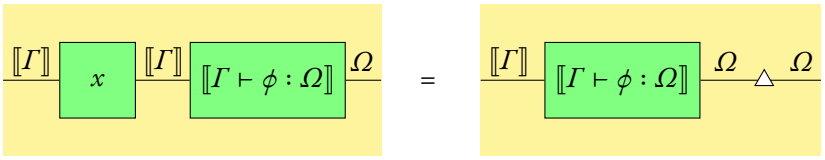
$=$



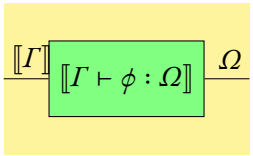
$\overset{4.2.9}{=}$



The Ax rule says given a morphism $[\![\Gamma \vdash \phi : \Omega]\!] : [\![\Gamma]\!] \to \Omega$, there exists an x such that



$=$



this is trivially satisfied by the identity on $[\![\Gamma]\!]$.

Now we can see the rules for the logical operations, and the morphisms they correspond to in each topos.

$$\frac{}{\Gamma \mid \Phi \vdash \mathtt{t}}\,(\mathtt{t}\mathrm{I})$$

$$\frac{\Gamma \mid \Phi \vdash \phi \qquad \Gamma \mid \Phi \vdash \psi}{\Gamma \mid \Phi \vdash \phi \wedge \psi}\,(\wedge\mathrm{I})$$

$$\frac{\Gamma \mid \Phi \vdash \phi \wedge \psi}{\Gamma \mid \Phi \vdash \phi}\,(\wedge\mathrm{E1})$$

$$\frac{\Gamma \mid \Phi \vdash \phi \wedge \psi}{\Gamma \mid \Phi \vdash \psi}\,(\wedge\mathrm{E2})$$

$$\frac{\Gamma \mid \Phi, \phi \vdash \psi}{\Gamma \mid \Phi \vdash \phi \Rightarrow \psi}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \mid \Phi \vdash \phi \Rightarrow \psi \qquad \Gamma \mid \Phi \vdash \phi}{\Gamma \mid \Phi \vdash \psi}\,(\Rightarrow\mathrm{E})$$

$$\frac{\Gamma, x : A \mid \Phi \vdash \phi(x)}{\Gamma \mid \Phi \vdash \forall(x : A).\phi(x)}\,(\forall\mathrm{I})$$

$$\frac{\Gamma \mid \Phi \vdash \forall(x : A).\phi(x)}{\Gamma, x : A \mid \Phi \vdash \phi(x)}\,(\forall\mathrm{E})$$

These morphisms can be easily inferred from the truth conditions given in the previous subsection. Hence, we see that the internal logic of a topos is sound with respect to these rules. In fact, however, it is not complete, there are three rules missing: the axiom of unique choice, and extensionality of both functions and predicates. Completeness is proved by showing that these rules form a topos themselves. For a detailed proof, see [Str, Chapter 13].

Chapter 5

Conclusion

In conclusion, we have seen a string-diagrammatic presentation of topoi. I have shown how this can be used to do topos theory purely via string diagrams – at least the parts that deal with categorical logic, and the fundamental theorem. In places, particularly in the case of the dependent product functor, we saw a significant simplification over the classical constructions.

It is my hope that these diagrams can be fine-tuned and used further to simplify topos theory and reasoning inside of topoi. Paraphrasing [Joh02], I hope that this can be the beginning of another sketch of the elephant that is topos theory.

Perhaps there is also use to be found in the string-diagrammatic syntax for higher-order intuitionistic logic presented in the final section. At least, it should be possible to define an entire proof assistant, which allows for everything to be proved using string-diagrams, although just as it is cumbersome to do *everything* with pure set theory, I suspect it would be cumbersome to limit *everything* to being done within string diagrams. So, again, the usefulness of this, outside of working within a topos directly, is unclear.

I have quite a few comments on where this work could be extended.

- I would like to see a purely diagrammatic characterisation of each $U_A : \mathcal{C}/A \to \mathcal{C}$, so that we never have to consider the actual action of the function, but can stay purely within string diagrams – this may not be possible, although I conjecture that everything can be proved with the graphical pullback lemma;
- I would like to see a good, simple, string diagrammatic account of the construction of exponentials in each slice of a topos – as remarked before, I think that if this is possible, then it would be most likely to be possible with the construction given in [Joh14];
- I would like to see the construction of coproducts in a topos done with string diagrams;
- I would like to see what a boolean topos looks like string diagrammatically, and perhaps, more general, any boolean category;
- I would like to see if this string-diagrammatic presentation yields any improvements in understanding in more areas of topos theory – and in particular in a Grothendieck topos, rather than in the elementary topoi considered here; and finally
- I would like to see whether the string-diagrammatic syntax for higher-order intuitionistic logic

could aid in finding a normal form for higher-order intuitionistic logic formulæ – as no such normal form exists.

References

[Abe24] C. B. Aberlé. *Foundations of Substructural Dependent Type Theory*. 2024. arXiv: 2401.15258 [cs.LO]. URL: https://arxiv.org/abs/2401.15258.

[BS10] J. Baez, and M. Stay. 'Physics, Topology, Logic and Computation: A Rosetta Stone'. In: *New Structures for Physics*. Edited by: B. Coecke. Springer Berlin Heidelberg, 2010, pp. 95–172. ISBN: 9783642128219. DOI: 10.1007/978-3-642-12821-9_2. URL: http://dx.doi.org/10.1007/978-3-642-12821-9_2.

[CK17] B. Coecke, and A. Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[D-CYP+23] S. Dündar-Coecke, L. Yeh, C. Puca, S. M.-L. Pfaendler, M. H. Waseem, T. Cervoni, A. Kissinger, S. Gogioso, and B. Coecke. 'Quantum Picturalism: Learning Quantum Theory in High School'. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, September 2023. DOI: 10.1109/qce57702.2023.20321. URL: http://dx.doi.org/10.1109/QCE57702.2023.20321.

[EM66] S. Eilenberg, and G. M. Kelly. 'Closed Categories'. In: *Proceedings of the Conference on Categorical Algebra*. Edited by: S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl. Springer Berlin Heidelberg, Berlin, Heidelberg, 1966, pp. 421–562. ISBN: 978-3-642-99902-4.

[Fox76] T. Fox. 'Coalgebras and cartesian categories'. In: *Communications in Algebra* 4 (7) (1976), pp. 665–667. DOI: 10.1080/00927877608822127.

[Fre72] P. Freyd. 'Aspects of topoi'. In: *Bulletin of the Australian Mathematical Society* 7 (1) (1972), pp. 1–76. DOI: 10.1017/S0004972700044828.

[Gol84] R. Goldblatt. *Topoi. the categorical analysis of logic*. Revised Edition. Studies in logic and the foundations of mathematics. Elsevier Science, 1984.

[GZ23] D. Ghica, and F. Zanasi. *String Diagrams for λ-calculi and Functional Computation*. 2023. arXiv: 2305.18945 [cs.LO]. URL: https://arxiv.org/abs/2305.18945.

[Joh02] P. T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*. Clarendon Press, Oxford, England, 2002.

[Joh14] P. Johnstone. *Topos Theory*. Dover Books on Mathematics. Dover Publications, 2014. ISBN: 9780486493367.

[JS88] A. Joyal, and R. Street. *Planar diagrams and tensor algebra*. 1988.

[JS91] A. Joyal, and R. Street. 'The geometry of tensor calculus, I'. In: *Advances in Mathematics* 88 (1) (1991), pp. 55–112. ISSN: 0001-8708. DOI: https://doi.org/10.1016/0001-8708(91)90003-P. URL: https://www.sciencedirect.com/science/article/pii/000187089190003P.

[LR20] F. Loregian, and E. Riehl. 'Categorical notions of fibration'. In: *Expositiones Mathematicae* 38 (4) (2020), pp. 496–514. ISSN: 0723-0869. DOI: https://doi.org/10.1016/j.exmath.2019.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0723086918300872.

[Mac63] S. Mac Lane. 'Natural Associativity and Commutativity'. In: *Rice Institute Pamphlet - Rice University Studies* 49 (1963), pp. 28–46.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*. 2nd edn. Graduate Texts in Mathematics. Springer New York, New York, 1971.

[Mel06] P.-A. Melliès. 'Functorial Boxes in String Diagrams'. In: *Computer Science Logic*. Edited by: Z. Ésik. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–30. ISBN: 978-3-540-45459-5.

[MM94] S. Mac Lane, and I. Moerdijk. *Sheaves in Geometry and Logic. A First Introduction to Topos Theory*. Universitext. Springer New York, New York, 1994.

[nlab:cmc] nLab authors. *closed monoidal category*. Revision 56. August 2024. URL: %5Curl%7Bhttps://ncatlab.org/nlab/show/closed+monoidal+category%7D.

[nlab:fcc] nLab authors. *finitely complete category*. Revision 26. August 2024. URL: https://ncatlab.org/nlab/show/finitely+complete+category.

[pwik:pl] Proof Wiki authors. *Pullback Lemma*. Change ID 690525. April 2024. URL: https://proofwiki.org/wiki/Pullback_Lemma.

[Res99] G. Restall. *An Introduction to Substructural Logics*. Routledge, New York, 1999.

[Rie17] E. Riehl. *Category theory in context*. Aurora: Dover modern math originals. Dover Publications, 2017. ISBN: 978-0-486-82080-4.

[Rom24] M. Román. *instances of Fox's theorem*. July 2024. URL: https://mroman42.github.io/notes/pieces/Foxs-theorem/.

[Sel11] P. Selinger. 'A Survey of Graphical Languages for Monoidal Categories'. In: *New Structures for Physics*. Edited by: B. Coecke. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 289–355. ISBN: 978-3-642-12821-9. DOI: 10.1007/978-3-642-12821-9_4. URL: https://doi.org/10.1007/978-3-642-12821-9_4.

[Str] T. Streicher. *Introduction to Category Theory and Categorical Logic*. URL: https://www2.mathematik.tu-darmstadt.de/~streicher/CTCL.pdf.

[Wij14] G. Wijnholds. 'Categorical Foundations for Extended Compositional Distributional Models of Meaning'. MSc Thesis. Universiteit van Amsterdam, 2014.