

Mac Lane's Comparison Theorem for the (co)Kleisli construction in Coq

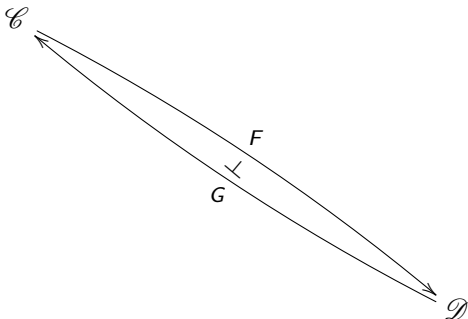
Burak Ekici

University of Innsbruck, Austria

August 13, 2018

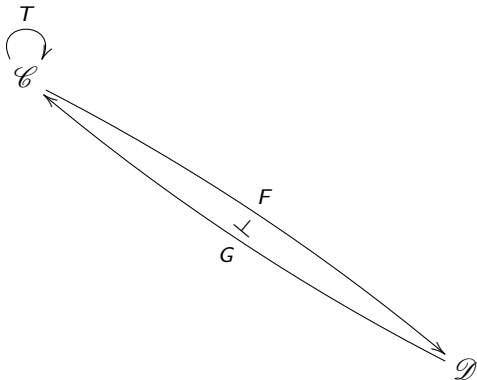


The statement



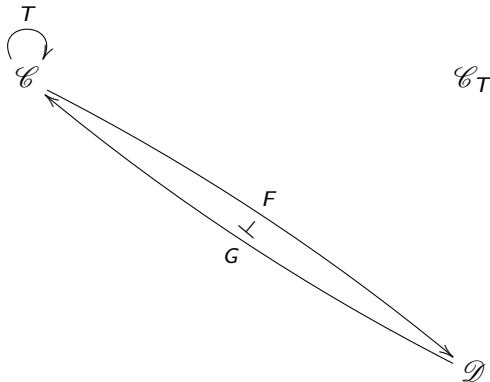


The statement



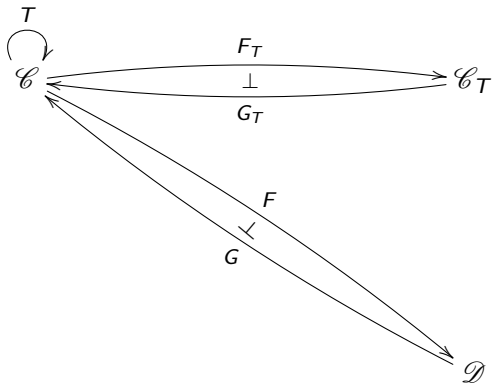


The statement



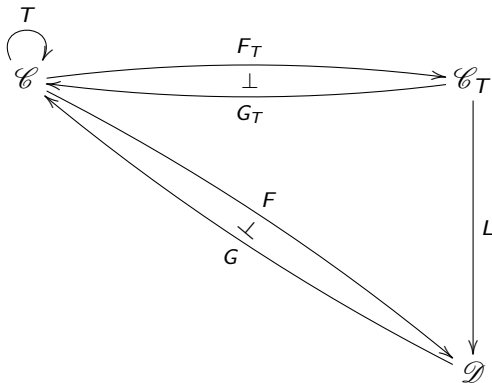


The statement



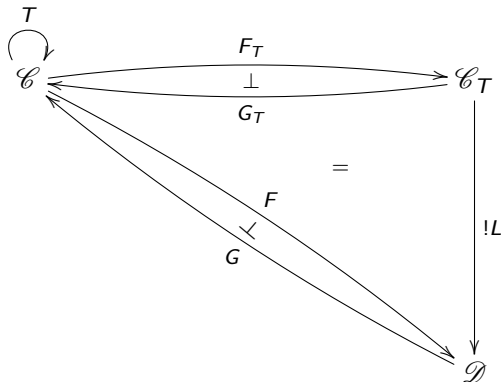


The statement





The statement



$$\exists !L: \mathcal{C}_T \rightarrow \mathcal{D}, L \circ F_T = F \wedge G \circ L = F_T$$



Adjunctions

$$\begin{array}{ccc}
 & F & \\
 \mathcal{C} & \begin{array}{c} \longleftarrow \\ \perp \\ \longrightarrow \end{array} & \mathcal{D} \\
 & G & \\
 \eta: Id \Rightarrow GF & F \dashv G & \varepsilon: FG \Rightarrow Id
 \end{array}$$

Definition

Let \mathcal{C} and \mathcal{D} be two categories. The functors $F: \mathcal{C} \rightarrow \mathcal{D}$ and $G: \mathcal{D} \rightarrow \mathcal{C}$ form an adjunction $F \dashv G: \mathcal{D} \rightarrow \mathcal{C}$ iff there exists *natural transformations* $\eta: Id_{\mathcal{C}} \Rightarrow GF$ and $\varepsilon: FG \Rightarrow Id_{\mathcal{D}}$ such that:

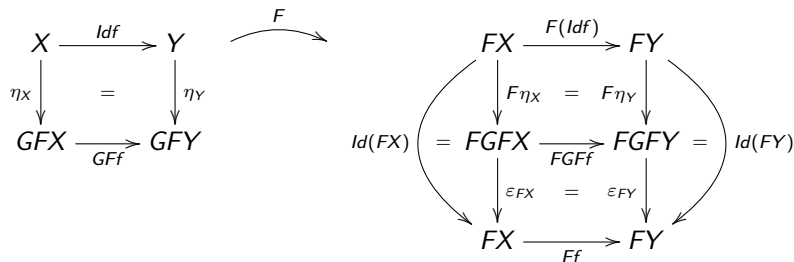
$$\varepsilon_{FX} \circ F\eta_X = id_{FX} \text{ for each } X \text{ in } \mathcal{C} \quad (1)$$

$$G\varepsilon_X \circ \eta_{GX} = id_{GX} \text{ for each } X \text{ in } \mathcal{D} \quad (2)$$



Adjunctions

$$\varepsilon_{FX} \circ F\eta_X = id_{FX} \text{ for each } X \text{ in } \mathcal{C}$$





Adjunctions: an example

In CIC, the logical \wedge and \Rightarrow are adjoint operations when Coq's *Prop* universe is defined as a category.

The proof is in the library comes with this talk.



Adjunctions: an example

In CIC, the logical \wedge and \Rightarrow are adjoint operations when Coq's *Prop* universe is defined as a category.

The proof is in the library comes with this talk.



Monads

Definition

A *monad* $T = (T, \eta, \mu)$ in a category \mathcal{C} consists of an endofunctor $T: \mathcal{C} \rightarrow \mathcal{C}$ with two natural transformations

$$\eta: Id_{\mathcal{C}} \Rightarrow T \quad \mu: T^2 \Rightarrow T \quad (3)$$

such that the following diagrams commute:

$$\begin{array}{ccc}
 T^3 & \xrightarrow{\mu T} & T^2 \\
 T\mu \downarrow & = & \downarrow \mu \\
 T^2 & \xrightarrow{\mu} & T
 \end{array}
 \qquad
 \begin{array}{ccc}
 T & \xrightarrow{\eta T} & T^2 \\
 T\eta \downarrow & \searrow id_T = & \downarrow \mu \\
 T^2 & \xrightarrow{\mu} & T
 \end{array}$$



Monad: quick examples

- A monoid is a monad in the category of endo-functors
- In CIC, the triple $(\text{fmapM}, \text{etaM}, \text{muM})$ forms a monad when Coq's *Type* universe is defined as a category.

```
Inductive maybe (A: Type)  $\triangleq$  just: A  $\rightarrow$  maybe A | nothing: maybe A.
```

```
Definition fmapM {A B: Type} (f: A  $\rightarrow$  B) (i: maybe A): maybe B  $\triangleq$ 
  match i with
  | just _ a  $\Rightarrow$  just _ (f a)
  | nothing _  $\Rightarrow$  nothing _
  end.
```

```
Definition etaM {A: Type} (a: A): maybe A  $\triangleq$  just A a.
```

```
Definition muM {A : Type} (i: maybe (maybe A)): maybe A  $\triangleq$ 
  match i with
  | just _ a  $\Rightarrow$  a
  | nothing _  $\Rightarrow$  nothing _
  end.
```

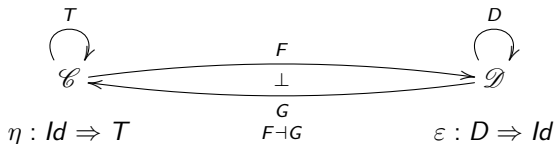


Every adjunction gives a monad

$$\begin{array}{ccc}
 & F & \\
 \mathcal{C} & \begin{array}{c} \xrightarrow{\quad} \\ \perp \\ \xleftarrow{\quad} \end{array} & \mathcal{D} \\
 & G & \\
 \eta : Id \Rightarrow GF & F \dashv G & \varepsilon : FG \Rightarrow Id
 \end{array}$$



Every adjunction gives a monad



Proposition

An adjunction $F \dashv G: \mathcal{D} \rightarrow \mathcal{C}$ determines a monad on \mathcal{C} and a comonad on \mathcal{D} as follows:

- The monad (T, η, μ) on \mathcal{C} $T = GF: \mathcal{C} \rightarrow \mathcal{C}$, $\mu_X = G(\varepsilon_{FX})$.
- The comonad (D, ε, δ) on \mathcal{D} $D = FG: \mathcal{D} \rightarrow \mathcal{D}$, $\delta_A = F(\eta_{GA})$.



Every (co)monad gives a (co)Kleisli
adjunction





Every (co)monad gives a (co)Kleisli
adjunction

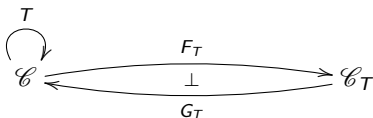

 \mathcal{C}_T

Proposition

Each monad (T, η, μ) on a category \mathcal{C} determines a Kleisli category \mathcal{C}_T ,



Every (co)monad gives a (co)Kleisli adjunction



Proposition

Each monad (T, η, μ) on a category \mathcal{C} determines a Kleisli category \mathcal{C}_T , and an associated adjunction $F_T \dashv G_T: \mathcal{C}_T \rightarrow \mathcal{C}$.



Every (co)monad gives a (co)Kleisli adjunction


 \mathcal{C}_T

- The categories \mathcal{C} and \mathcal{C}_T have the same objects and there is a morphism $f^b: X \rightarrow Y$ in \mathcal{C}_T for each $f: X \rightarrow TY$ in \mathcal{C} .
- For each object X in \mathcal{C}_T , the identity arrow is $id_X = h^b: X \rightarrow X$ in \mathcal{C}_T where

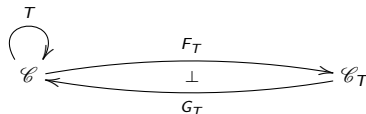
$$h = \eta_X: X \rightarrow TX \text{ in } \mathcal{C}.$$

- The composition of a pair of morphisms $f^b: X \rightarrow Y$ and $g^b: Y \rightarrow Z$ in \mathcal{C}_T is given by the Kleisli composition:

$$g^b \circ f^b = h^b: X \rightarrow Z \text{ where } h = \mu_Z \circ Tg \circ f: X \rightarrow TZ \text{ in } \mathcal{C}.$$



Every (co)monad gives a (co)Kleisli adjunction



- The functor $F_T: \mathcal{C} \rightarrow \mathcal{C}_T$ is the identity on objects. On morphisms,

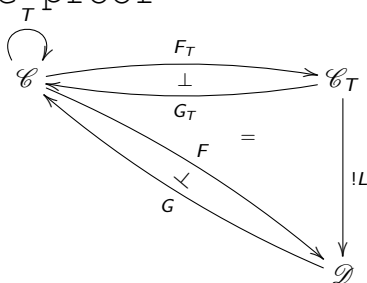
$$F_T f = (\eta_Y \circ f)^b, \text{ for each } f: X \rightarrow Y \text{ in } \mathcal{C}. \quad (4)$$

- The functor $G_T: \mathcal{C}_T \rightarrow \mathcal{C}$ maps each object X in \mathcal{C}_T to TX in \mathcal{C} . On morphisms,

$$G_T(g^b) = \mu_Y \circ Tg, \text{ for each } g^b: X \rightarrow Y \text{ in } \mathcal{C}_T. \quad (5)$$



Sketch of the proof



- 1 Characterize some map $L: \mathcal{C}_T \rightarrow \mathcal{D}$ by

$$\begin{cases} LX &= FX \\ Lf^b &= \varepsilon_{FY} \circ Ff, \text{ for each } f^b: X \rightarrow Y \text{ in } \mathcal{C}_T \end{cases}$$

- 2 Prove that L is a functor satisfying $GL = G_T$ and $LF_T = F$.
- 3 Show that L is unique



Proof: L is a functor

- For each X in \mathcal{C}_T , $id_X = (\eta_X)^b$ in \mathcal{C}_T , we have $L(id_X) = L(\eta_X)^b = \varepsilon_{FX} \circ F\eta_X$. We get

$$\varepsilon_{FX} \circ F\eta_X = id_{FX} = id_{LX}.$$

- For each pair of morphisms $f^b: X \rightarrow Y$ and $g^b: Y \rightarrow Z$ in \mathcal{C}_T , by Kleisli composition, we obtain

$$L(g^b \circ f^b) = \varepsilon_{FZ} \circ FG\varepsilon_{FZ} \circ FGFg \circ Ff.$$

Since ε is natural, we have $\varepsilon_{FZ} \circ Fg \circ \varepsilon_{FY} \circ Ff$ which is $L(g^b) \circ L(f^b)$ in \mathcal{D} .

Hence $L: \mathcal{C}_T \rightarrow \mathcal{D}$ is a functor.



Proof: L satisfies ...

- 1 For each object X in \mathcal{C}_T , $LX = FX$ in \mathcal{D} and $GLX = GFX = TX = G_TX$ in \mathcal{C} . For each morphism $f^b: X \rightarrow Y$ in \mathcal{C}_T , $Lf^b = \varepsilon_{FY} \circ Ff$ in D by definition. Hence,

$$GLf^b = G\varepsilon_{FY} \circ GFf.$$

Similarly, by definition $G_Tf^b = \mu_Y \circ Tf$. Since $\mu_Y = G(\varepsilon_{FY})$, we get

$$G_Tf^b = G\varepsilon_{FY} \circ GFf.$$

We get $GLf^b = G_Tf^b$ for each mapping f^b . Thus $GL = G_T$.

- 2 F_T is the identity on objects, thus $LF_TX = LX = FX$. For each morphism $f: X \rightarrow Y$ in \mathcal{C} , we have $F_Tf = (\eta_Y \circ f)^b$ in \mathcal{C}_T , by definition. So that

$$LF_Tf = L(\eta_Y \circ f)^b = \varepsilon_{FY} \circ F\eta_Y \circ Ff.$$

Due to ε and η being natural, we have $\varepsilon_{FY} \circ F\eta_Y = id_{FY}$ yielding $LF_Tf = Ff$ for each mapping f . Therefore $LF_T = F$.



Proof: L is unique

We need to show that $\forall R: \mathcal{C}_T \rightarrow \mathcal{D}$ such that $R \circ F_T = F$ and $G \circ R = G_T, L = R$.

■ forall X in $\mathcal{C}_T, LX = RX$

$$\begin{aligned}
 LX &= FX && \text{(by definition)} \\
 &= RF_T X && \text{(by axiom)} \\
 &= RX && (F_T \text{ is the identity})
 \end{aligned}$$

Therefore, $LX = RX$



Proof: L is unique

Lemma

Let $F \dashv G: \mathcal{D} \rightarrow \mathcal{C}$ be an adjunction. For each $f: X \rightarrow GY$ in \mathcal{C} , and $g, h: FX \rightarrow Y$ in \mathcal{D} , if $f = Gg \circ \eta_X$ and $f = Gh \circ \eta_X$ then $g = h$.

■ forall $f^b: X \rightarrow Y$ in $\mathcal{C}_T, Lf^b = Rf^b$

$$G(Rf^b) \circ \eta_X = G_T f^b \circ \eta_X \quad (\text{by axiom})$$

$$= \mu_Y \circ GFf \circ \eta_X \quad (\text{by definition of } G_T)$$

$$= \mu_Y \circ \eta_{GFY} \circ f \quad (\text{by naturality of } \eta)$$

$$= f \quad (GF \text{ is a monad})$$

$$G(Lf^b) \circ \eta_X = G\varepsilon_{FY} \circ GFf \circ \eta_X \quad (\text{by definition of } L)$$

$$= G\varepsilon_{FY} \circ \eta_{GFY} \circ f \quad (\text{by naturality of } \eta)$$

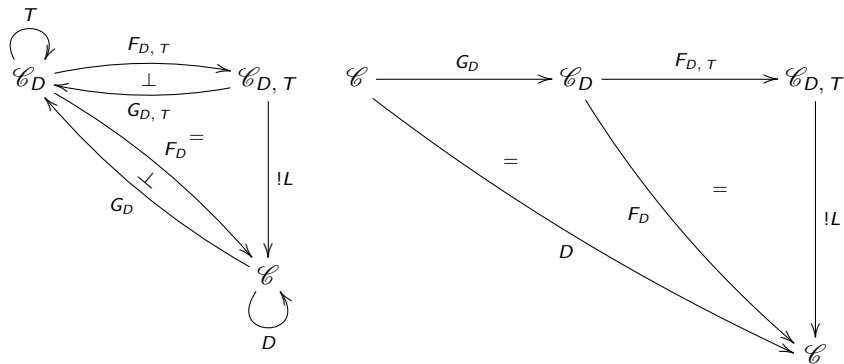
$$= f \quad (F \dashv G \text{ is an adjunction})$$

Therefore, $\mathcal{C}_T, Lf^b = Rf^b$



A Use Case

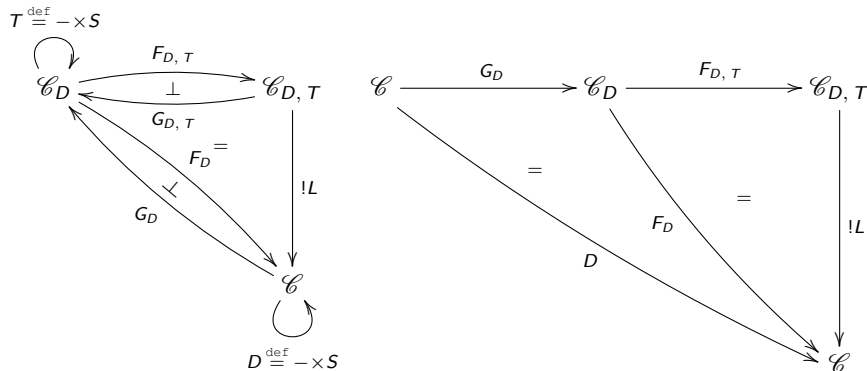
The *full image factorization* (or decomposition) of D is given by $L \circ G_D \circ F_{D,\tau}$.





A Use Case

The *full image factorization* (or decomposition) of D is given by $L \circ G_D \circ F_{D,\tau}$.



A Use Case

The *full image factorization* (or decomposition) of D^2 is given by $K \circ F_{D,T,D,T} \circ G_{D,T,D} \circ F_{D,T} \circ G_D$.

$$\begin{array}{ccccccc}
\mathcal{C} & \xrightarrow{G_D} & \mathcal{C}_D & \xrightarrow{F_{D,T}} & \mathcal{C}_{D,T} & \xrightarrow{G_{D,T,D}} & \mathcal{C}_{D,T,D} \xrightarrow{F_{D,T,D,T}} \mathcal{C}_{D,T,D,T} \\
& & & & = & & \\
& \searrow & & & & \nearrow & \\
& & D^2 & \rightarrow & \mathcal{C} & \leftarrow & !K
\end{array}$$



Category theory in Coq

Some Category theory formalized in Coq:

- J. Gross, A. Chlipala, and D. I. Spivak. Experience implementing a performant category-theory library in Coq.
- A. Timany and B. Jacobs. Category theory in Coq 8.5.
- J. Wiegley's library on github.
- As a part of UniMath library.
- ...

All (except the one in UniMath) represent category theoretical objects (i.e., categories, functors, etc.) with Coq type classes.



Category theory in Coq

Our approach is no different.

```

Class Functor (C D: Category): Type  $\triangleq$ 
  mk_Functor
  {
    fobj          : @obj C  $\rightarrow$  @obj D;
    fmap          :  $\forall$  {a b: @obj C} (f: arrow b a), (arrow (fobj b) (fobj a));
    preserve_id   :  $\forall$  {a: @obj C}, fmap (@identity C a) = (@identity D (fobj a));
    preserve_comp :  $\forall$  {a b c: @obj C} (g : @arrow C c b) (f: @arrow C b a),
                      fmap (g o f) = (fmap g) o (fmap f)
  }.

```



A difficulty

Might be difficult to prove equalities of two class instances.

```

Lemma F_split:  $\forall$  (C D : Category) (F G : Functor C D),
    fobj F = fobj G  $\rightarrow$  JMeq (fmap F) (fmap G)  $\rightarrow$  F = G.

Proof.
...
Qed.

```

Requires to struggle with explicit coercions hidden behind the heterogeneous equality.



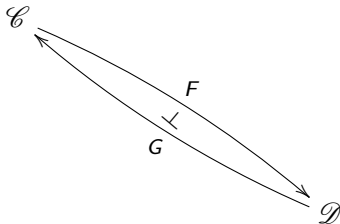
A difficulty

One way to deal with that: converting the goal into an equality on dependent pairs:

```
...
_____ (1/1)
{p : (∀ a b : obj, arrow b a → arrow (fobj F b) (fobj F a)) =
  (∀ a b : obj, arrow b a → arrow (fobj G b) (fobj G a)) &
match p in (_ = y) return y with
| eq_refl ⇒ @fmap _ _ F
end = @fmap _ _ G}
```

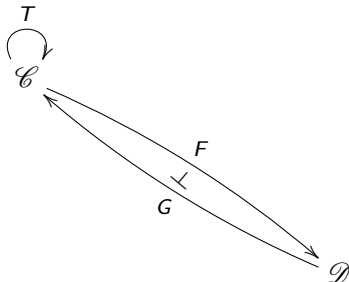



The statement in Coq





The statement in Coq



```
Theorem adj_mon: ∀ {C D: Category} (F : Functor C D) (G: Functor D C),
  Adjunction F G → Monad C (Compose_Functors F G).
```

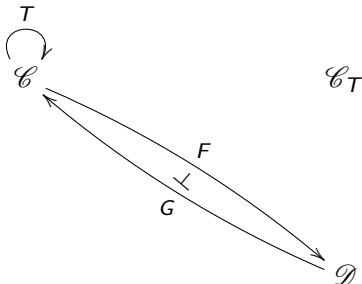
```
Proof.
```

```
...
```

```
Qed.
```



The statement in Coq



Definition Kleisli_Category

(C: Category) (T: Functor C C) (M: Monad C T): Category.

Proof. unshelve econstructor.

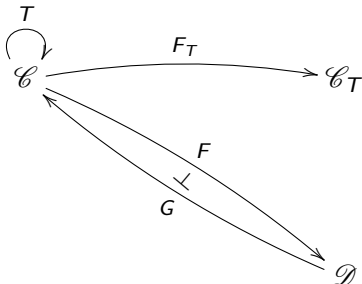
- exact (@obj C).
- intros a b. exact (@arrow C (fobj T a) b).

...

Defined.



The statement in Coq



Definition `FT` {C D: Category} (F: Functor C D) (G: Functor D C)

(T \triangleq Compose_Functors F G) (M: Monad C T)

(CT \triangleq (Kleisli_Category C T M)): Functor C CT.

Proof. intros; unshelve econstructor; simpl.

- exact id.

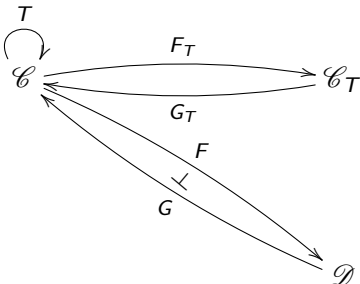
- intros a b f. exact (trans eta b o f).

...

Defined.



The statement in Coq



Definition GT {C D: Category} (F: Functor C D) (G: Functor D C)

(T \triangleq Compose_Functors F G) (M: Monad C T)

(CT \triangleq (Kleisli_Category C T M)): Functor CT C.

Proof. intros; unshelve econstructor; simpl.

- exact (fobj T).

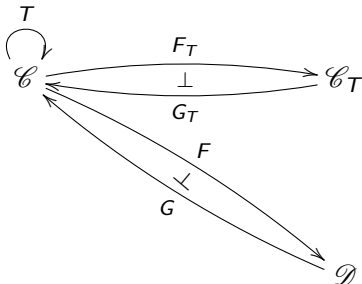
- intros a b g. exact (trans mu b o fmap T g).

...

Defined.



The statement in Coq



Theorem `mon_kladj`: $\forall \{C\ D : \text{Category}\} (F : \text{Functor } C\ D) (G : \text{Functor } D\ C)$

$(T \triangleq \text{Compose_Functors } F\ G) (M : \text{Monad } C\ T)$

$(FT \triangleq FT\ F\ G\ M) (GT \triangleq GT\ F\ G\ M), \text{Adjunction } FT\ GT.$

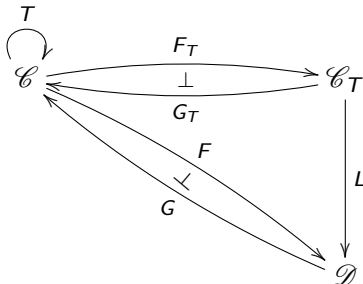
Proof.

...

Qed.



The statement in Coq



Definition L:

```

∀ {C D: Category} (F: Functor C D) (G: Functor D C) (A1: Adjunction F G),
let M ≐ (adj_mon F G A1) in let CM ≐ (adj_comon F G A1) in
let CT ≐ (Kleisli_Category C (Compose_Functors F G) M) in
let FT ≐ (FT F G M) in let GT ≐ (GT F G M) in
let A2 ≐ (mon_kladj F G M) in Functor CT D.

```

```

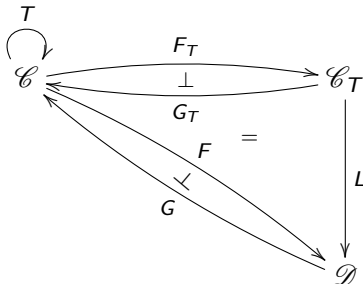
Proof. intros. cbn in *.
      unshelve econstructor.
      - exact (fobj F).
      - intros a b f. exact (trans eps (fobj F b) o fmap F f).
      ...

```

Defined.



The statement in Coq



Lemma commL:

```

∀ {C D: Category} (F: Functor C D) (G: Functor D C) (A1: Adjunction F G),
  let M ≐ (@adj_mon C D F G A1) in
  let CT ≐ (Kleisli_Category C (Compose_Functors F G) M) in
  let FT ≐ (FT F G M) in let GT ≐ (GT F G M) in
  let A2 ≐ (mon_kladj F G M) in
    (Compose_Functors FT (L F G A1)) = F ∧
    (Compose_Functors (L F G A1) G) = GT.

```

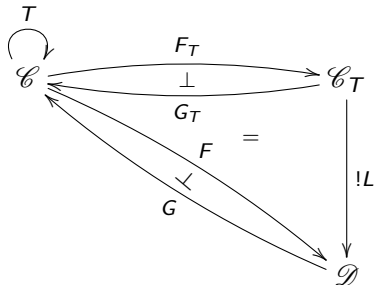
Proof.

...

Qed.



The statement in Coq



Lemma uniqueL:

```

  ∀ {C D: Category} (F: Functor C D) (G: Functor D C) (A1: Adjunction F G),
  let M ≙ (adj_mon F G A1) in
  let CK ≙ (Kleisli_Category C (Compose_Functors F G) M) in
  let FT ≙ (FT F G M) in let GT ≙ (GT F G M) in
  let A2 ≙ (mon_kladj F G M) in
  ∀ R : Functor CK D, Compose_Functors FT R = F ∧ Compose_Functors R G = GT
  → (L F G A1) = R.

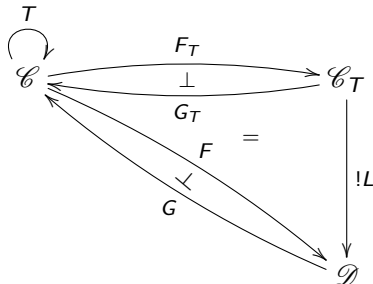
```

Proof.

...

Qed.

The statement in Coq



Lemma ComparisonMacLane:

```

∀ {C D: Category} (F: Functor C D) (G: Functor D C) (A1: Adjunction F G),
let M ≙ (adj_mon F G A1) in
let CK ≙ (Kleisli_Category C (Compose_Functors F G) M) in
let FT ≙ (FT F G M) in let GT ≙ (GT F G M) in
let A2 ≙ (mon_kladj F G M) in
  ∃ !L, (Compose_Functors FT L) = F ∧ (Compose_Functors L G) = GT.

```

Proof. intros C D F G A1 M CT FT GT A2.

∃ (L F G A1). split.

- apply commL.

- apply uniqueL.

Qed.



Thank you for your attention!

&

Questions?