Category Theory to Yoneda's Lemma

Greg O'Keefe

March 17, 2025

This development proves Yoneda's lemma and aims to be readable by humans. It only defines what is needed for the lemma: categories, functors and natural transformations. Limits, adjunctions and other important concepts are not included.

There is no explanation or discussion in this document. See [O'K04] for this and a survey of category theory formalisations.

Contents

1 Categories

theory *Cat*
imports *HOL−Library.FuncSet*
begin

1.1 Definitions

record ($'o$, $'a$) *category* =
  ob :: $'o$ *set* (‹*Ob*ı› *70*)
  ar :: $'a$ *set* (‹*Ar*ı› *70*)
  dom :: $'a \Rightarrow 'o$ (‹*Dom*ı -› [*81*] *70*)
  cod :: $'a \Rightarrow 'o$ (‹*Cod*ı -› [*81*] *70*)
  id :: $'o \Rightarrow 'a$ (‹*Id*ı -› [*81*] *80*)
  comp :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** ‹·ı› *60*)

definition
  hom :: [($'o$,$'a$,$'m$) *category-scheme*, $'o$, $'o$] $\Rightarrow$ $'a$ *set*
    (‹*Hom*ı - -› [*81*,*81*] *80*) **where**
  hom CC A B = { *f*. *f*∈*ar CC* & *dom CC f* = *A* & *cod CC f* = *B* }

locale *category* =
  fixes *CC* (**structure**)
  assumes *dom-object* [*intro*]:
  $f \in Ar \Longrightarrow Dom\ f \in Ob$
  and *cod-object* [*intro*]:
  $f \in Ar \Longrightarrow Cod\ f \in Ob$
  and *id-left* [*simp*]:
  $f \in Ar \Longrightarrow Id\ (Cod\ f) \cdot f = f$
  and *id-right* [*simp*]:
  $f \in Ar \Longrightarrow f \cdot Id\ (Dom\ f) = f$
  and *id-hom* [*intro*]:
  $A \in Ob \Longrightarrow Id\ A \in Hom\ A\ A$
  and *comp-types* [*intro*]:
  $\bigwedge A\ B\ C.\ (comp\ CC) : (Hom\ B\ C) \to (Hom\ A\ B) \to (Hom\ A\ C)$
  and *comp-associative* [*simp*]:
  $f \in Ar \Longrightarrow g \in Ar \Longrightarrow h \in Ar$
  $\Longrightarrow Cod\ h = Dom\ g \Longrightarrow Cod\ g = Dom\ f$
  $\Longrightarrow f \cdot (g \cdot h) = (f \cdot g) \cdot h$

1.2 Lemmas

lemma (**in** *category*) *homI*:
  assumes $f \in Ar$ **and** $Dom\ f = A$ **and** $Cod\ f = B$
  shows $f \in Hom\ A\ B$
  ⟨*proof*⟩

lemma (**in** *category*) *homE*:
  assumes $A \in Ob$ **and** $B \in Ob$ **and** $f \in Hom\ A\ B$
  shows $Dom\ f = A$ **and** $Cod\ f = B$

⟨*proof*⟩

lemma (**in** *category*) *id-arrow* [*intro*]:
  assumes $A \in Ob$
  shows $Id\ A \in Ar$
⟨*proof*⟩

lemma (**in** *category*) *id-dom-cod*:
  assumes $A \in Ob$
  shows $Dom\ (Id\ A) = A$ **and** $Cod\ (Id\ A) = A$
⟨*proof*⟩

lemma (**in** *category*) *compI* [*intro*]:
  assumes *f*: $f \in Ar$ **and** *g*: $g \in Ar$ **and** $Cod\ f = Dom\ g$
  shows $g \cdot f \in Ar$
  and $Dom\ (g \cdot f) = Dom\ f$
  and $Cod\ (g \cdot f) = Cod\ g$
⟨*proof*⟩

end

2 Set is a Category

theory *SetCat*
imports *Cat*
begin

2.1 Definitions

record *'c set-arrow* =
  set-dom :: *'c set*
  set-func :: *'c* ⇒ *'c*
  set-cod :: *'c set*

definition
  set-arrow :: [*'c set*, *'c set-arrow*] ⇒ *bool* **where**
  set-arrow U f ⟷ *set-dom f* ⊆ *U* & *set-cod f* ⊆ *U*
    & (*set-func f*): (*set-dom f*) → (*set-cod f*)
    & *set-func f* ∈ *extensional* (*set-dom f*)

definition
  set-id :: [*'c set*, *'c set*] ⇒ *'c set-arrow* **where**
  set-id U = (λ*s*∈*Pow U*. ⦇*set-dom*=*s*, *set-func*=λ*x*∈*s*. *x*, *set-cod*=*s*⦈)

definition
  set-comp :: [*'c set-arrow*, *'c set-arrow*] ⇒ *'c set-arrow* (**infix** ‹⊙› *70*) **where**
  set-comp g f =
  ⦇

$set\text{-}dom = set\text{-}dom\ f,$
$set\text{-}func = compose\ (set\text{-}dom\ f)\ (set\text{-}func\ g)\ (set\text{-}func\ f),$
$set\text{-}cod = set\text{-}cod\ g$
$\rparr$

definition
$set\text{-}cat :: {'c}\ set \Rightarrow ({'c}\ set,\ {'c}\ set\text{-}arrow)\ category$ **where**
$set\text{-}cat\ U =$
$\lparr$
$ob = Pow\ U,$
$ar = \{f.\ set\text{-}arrow\ U\ f\},$
$dom = set\text{-}dom,$
$cod = set\text{-}cod,$
$id = set\text{-}id\ U,$
$comp = set\text{-}comp$
$\rparr$

2.2 Simple Rules and Lemmas

lemma *set-objectI* [*intro*]: $A \subseteq U \Longrightarrow A \in ob\ (set\text{-}cat\ U)$
⟨*proof*⟩

lemma *set-objectE* [*intro*]: $A \in ob\ (set\text{-}cat\ U) \Longrightarrow A \subseteq U$
⟨*proof*⟩

lemma *set-homI* [*intro*]:
assumes $A \subseteq U$
and $B \subseteq U$
and $f : A \rightarrow B$
and $f \in extensional\ A$
shows $\lparr set\text{-}dom{=}A,\ set\text{-}func{=}f,\ set\text{-}cod{=}B \rparr \in hom\ (set\text{-}cat\ U)\ A\ B$
⟨*proof*⟩

lemma *set-dom* [*simp*]: $dom\ (set\text{-}cat\ U)\ f = set\text{-}dom\ f$
⟨*proof*⟩

lemma *set-cod* [*simp*]: $cod\ (set\text{-}cat\ U)\ f = set\text{-}cod\ f$
⟨*proof*⟩

lemma *set-id* [*simp*]: $id\ (set\text{-}cat\ U)\ A = set\text{-}id\ U\ A$
⟨*proof*⟩

lemma *set-comp* [*simp*]: $comp\ (set\text{-}cat\ U)\ g\ f = g \odot f$
⟨*proof*⟩

lemma *set-dom-cod-object-subset* [*intro*]:
assumes f: $f \in ar\ (set\text{-}cat\ U)$
shows $dom\ (set\text{-}cat\ U)\ f \in ob\ (set\text{-}cat\ U)$

and $cod\ (set\text{-}cat\ U)\ f \in ob\ (set\text{-}cat\ U)$
and $set\text{-}cod\ f \subseteq U$
and $set\text{-}dom\ f \subseteq U$
$\langle proof \rangle$

In this context, $f \in hom\ A\ B$ is quite a strong claim.

lemma *set-homE* [*intro*]:
assumes f: $f \in hom\ (set\text{-}cat\ U)\ A\ B$
shows $A \subseteq U$
and $B \subseteq U$
and $set\text{-}dom\ f = A$
and $set\text{-}func\ f : A \to B$
and $set\text{-}cod\ f = B$
$\langle proof \rangle$

2.3 Set is a Category

lemma *set-id-left*:
assumes f: $f \in ar\ (set\text{-}cat\ U)$
shows $set\text{-}id\ U\ (set\text{-}cod\ f) \odot f = f$
$\langle proof \rangle$

lemma *set-id-right*:
assumes f: $f \in ar\ (set\text{-}cat\ U)$
shows $f \odot (set\text{-}id\ U\ (set\text{-}dom\ f)) = f$
$\langle proof \rangle$

lemma *set-id-hom*:
assumes $A \in ob\ (set\text{-}cat\ U)$
shows $id\ (set\text{-}cat\ U)\ A \in hom\ (set\text{-}cat\ U)\ A\ A$
$\langle proof \rangle$

lemma *set-comp-types*:
$comp\ (set\text{-}cat\ U) \in hom\ (set\text{-}cat\ U)\ B\ C \to hom\ (set\text{-}cat\ U)\ A\ B \to hom\ (set\text{-}cat\ U)\ A\ C$
$\langle proof \rangle$

We reason explicitly about the function component of the composite arrow, leaving the rest to the simplifier.

lemma *set-comp-associative*:
fixes f **and** g **and** h
assumes f: $f \in ar\ (set\text{-}cat\ U)$
and g: $g \in ar\ (set\text{-}cat\ U)$
and h: $h \in ar\ (set\text{-}cat\ U)$
and hg: $cod\ (set\text{-}cat\ U)\ h = dom\ (set\text{-}cat\ U)\ g$
and gf: $cod\ (set\text{-}cat\ U)\ g = dom\ (set\text{-}cat\ U)\ f$
shows $comp\ (set\text{-}cat\ U)\ f\ (comp\ (set\text{-}cat\ U)\ g\ h) =$
$comp\ (set\text{-}cat\ U)\ (comp\ (set\text{-}cat\ U)\ f\ g)\ h$

⟨*proof*⟩

theorem *set-cat-cat*: *category* (*set-cat U*)
⟨*proof*⟩

end

3 Functors

theory *Functors*
imports *Cat*
begin

3.1 Definitions

record ($'o1$,$'a1$,$'o2$,$'a2$) *functor* =
  om :: $'o1 \Rightarrow 'o2$
  am :: $'a1 \Rightarrow 'a2$

abbreviation
  om-syn (‹- $_{\mathrm{o}}$› [81]) **where**
  $F_{\mathrm{o}} \equiv om\ F$

abbreviation
  am-syn (‹- $_{\mathrm{a}}$› [81]) **where**
  $F_{\mathrm{a}} \equiv am\ F$

locale *two-cats* = *AA?*: *category AA* + *BB?*: *category BB*
  for *AA* :: ($'o1$,$'a1$,$'m1$)*category-scheme* (**structure**)
  and *BB* :: ($'o2$,$'a2$,$'m2$)*category-scheme* (**structure**) +
 fixes *preserves-dom* :: ($'o1$,$'a1$,$'o2$,$'a2$)*functor* ⇒ *bool*
  and *preserves-cod* :: ($'o1$,$'a1$,$'o2$,$'a2$)*functor* ⇒ *bool*
  and *preserves-id* :: ($'o1$,$'a1$,$'o2$,$'a2$)*functor* ⇒ *bool*
  and *preserves-comp* :: ($'o1$,$'a1$,$'o2$,$'a2$)*functor* ⇒ *bool*
 defines *preserves-dom* $G \equiv \forall f{\in}Ar_{AA}.\ G_{\mathrm{o}}\ (Dom_{AA}\ f) = Dom_{BB}\ (G_{\mathrm{a}}\ f)$
  and *preserves-cod* $G \equiv \forall f{\in}Ar_{AA}.\ G_{\mathrm{o}}\ (Cod_{AA}\ f) = Cod_{BB}\ (G_{\mathrm{a}}\ f)$
  and *preserves-id* $G \equiv \forall A{\in}Ob_{AA}.\ G_{\mathrm{a}}\ (Id_{AA}\ A) = Id_{BB}\ (G_{\mathrm{o}}\ A)$
  and *preserves-comp* $G \equiv$
   $\forall f{\in}Ar_{AA}.\ \forall g{\in}Ar_{AA}.\ Cod_{AA}\ f = Dom_{AA}\ g \longrightarrow G_{\mathrm{a}}\ (g \cdot_{AA} f) = (G_{\mathrm{a}}\ g)$
$\cdot_{BB}\ (G_{\mathrm{a}}\ f)$

locale *functor* = *two-cats* +
 fixes *F* (**structure**)
 assumes *F-preserves-arrows*: $F_{\mathrm{a}} : Ar_{AA} \to Ar_{BB}$
  and *F-preserves-objects*: $F_{\mathrm{o}} : Ob_{AA} \to Ob_{BB}$
  and *F-preserves-dom*: *preserves-dom F*
  and *F-preserves-cod*: *preserves-cod F*
  and *F-preserves-id*: *preserves-id F*

and *F-preserves-comp*: *preserves-comp F*
begin

lemmas *F-axioms* = *F-preserves-arrows F-preserves-objects F-preserves-dom F-preserves-cod F-preserves-id F-preserves-comp*

lemmas *func-pred-defs* = *preserves-dom-def preserves-cod-def preserves-id-def preserves-comp-def*

end

This gives us nicer notation for asserting that things are functors.

abbreviation
Functor (‹*Functor* - : - ⟶ -› [*81*]) **where**
Functor F : *AA* ⟶ *BB* ≡ *functor AA BB F*

3.2 Simple Lemmas

For example:

lemma (**in** *functor*) *Functor F* : *AA* ⟶ *BB* ⟨*proof*⟩

lemma *functors-preserve-arrows* [*intro*]:
assumes *Functor F* : *AA* ⟶ *BB*
and $f \in ar\ AA$
shows $F_{\mathrm{a}}\ f \in ar\ BB$
⟨*proof*⟩

lemma (**in** *functor*) *functors-preserve-homsets*:
assumes *1*: $A \in Ob_{AA}$
and *2*: $B \in Ob_{AA}$
and *3*: $f \in Hom_{AA}\ A\ B$
shows $F_{\mathrm{a}}\ f \in Hom_{BB}\ (F_{\mathrm{o}}\ A)\ (F_{\mathrm{o}}\ B)$
⟨*proof*⟩

lemma *functors-preserve-objects* [*intro*]:
assumes *Functor F* : *AA* ⟶ *BB*
and $A \in ob\ AA$
shows $F_{\mathrm{o}}\ A \in ob\ BB$
⟨*proof*⟩

3.3 Identity Functor

definition
id-func :: ('o,'a,'m) *category-scheme* ⇒ ('o,'a,'o,'a) *functor* **where**
id-func CC = (| $om=(\lambda A{\in}ob\ CC.\ A)$, $am=(\lambda f{\in}ar\ CC.\ f)$ |)

locale *one-cat* = *two-cats* +
  assumes *endo*: $BB = AA$

lemma (**in** *one-cat*) *id-func-preserves-arrows*:
  shows $(\textit{id-func}\ AA)_{\mathrm{a}} : Ar \rightarrow Ar$
  ⟨*proof*⟩

lemma (**in** *one-cat*) *id-func-preserves-objects*:
  shows $(\textit{id-func}\ AA)_{\mathrm{o}} : Ob \rightarrow Ob$
  ⟨*proof*⟩

lemma (**in** *one-cat*) *id-func-preserves-dom*:
  shows *preserves-dom* (*id-func AA*)
⟨*proof*⟩

lemma (**in** *one-cat*) *id-func-preserves-cod*:
  preserves-cod (*id-func AA*)
⟨*proof*⟩

lemma (**in** *one-cat*) *id-func-preserves-id*:
  preserves-id (*id-func AA*)
⟨*proof*⟩

lemma (**in** *one-cat*) *id-func-preserves-comp*:
  preserves-comp (*id-func AA*)
⟨*proof*⟩

theorem (**in** *one-cat*) *id-func-functor*:
  Functor (*id-func AA*) : $AA \longrightarrow AA$
⟨*proof*⟩

end

4 HomFunctors

theory *HomFunctors*
imports *SetCat Functors*
begin

locale *into-set* = *two-cats AA BB*
    for $AA :: ('o,'a,'m)$*category-scheme* (**structure**)
    and *BB* (**structure**) +
  fixes *U* **and** *Set*
  defines $U \equiv (\mathit{UNIV}::'a\ set)$
  defines $Set \equiv$ *set-cat U*

assumes *BB-Set*: $BB = Set$
fixes *homf* (‹Hom'(-,′-′)›)
defines *homf* $A \equiv$ ⦇
om = ($\lambda B \in Ob$. *Hom A B*),
am = ($\lambda f \in Ar$. ⦇*set-dom*=*Hom A* (*Dom f*),*set-func*=($\lambda g \in$*Hom A* (*Dom f*). $f \cdot g$),*set-cod*=*Hom A* (*Cod f*)⦈)
⦈

lemma (**in** *into-set*) *homf-preserves-arrows*:
$Hom(A,\text{-})_{\mathrm{a}} : Ar \to ar\ Set$
⟨*proof*⟩

lemma (**in** *into-set*) *homf-preserves-objects*:
$Hom(A,\text{-})_{\mathrm{o}} : Ob \to ob\ Set$
⟨*proof*⟩

lemma (**in** *into-set*) *homf-preserves-dom*:
assumes *f*: $f \in Ar$
shows $Hom(A,\text{-})_{\mathrm{o}}\ (Dom\ f) = dom\ Set\ (Hom(A,\text{-})_{\mathrm{a}}\ f)$
⟨*proof*⟩

lemma (**in** *into-set*) *homf-preserves-cod*:
assumes *f*: $f \in Ar$
shows $Hom(A,\text{-})_{\mathrm{o}}\ (Cod\ f) = cod\ Set\ (Hom(A,\text{-})_{\mathrm{a}}\ f)$
⟨*proof*⟩

lemma (**in** *into-set*) *homf-preserves-id*:
assumes *B*: $B \in Ob$
shows $Hom(A,\text{-})_{\mathrm{a}}\ (Id\ B) = id\ Set\ (Hom(A,\text{-})_{\mathrm{o}}\ B)$
⟨*proof*⟩

lemma (**in** *into-set*) *homf-preserves-comp*:
assumes *f*: $f \in Ar$
and *g*: $g \in Ar$
and *fg*: $Cod\ f = Dom\ g$
shows $Hom(A,\text{-})_{\mathrm{a}}\ (g \cdot f) = (Hom(A,\text{-})_{\mathrm{a}}\ g) \odot (Hom(A,\text{-})_{\mathrm{a}}\ f)$
⟨*proof*⟩

theorem (**in** *into-set*) *homf-into-set*:
$Functor\ Hom(A,\text{-}) : AA \longrightarrow Set$
⟨*proof*⟩

end

5 Natural Transformations

theory *NatTrans*
imports *Functors*
begin

locale *natural-transformation* = *two-cats* +
fixes F **and** G **and** u
assumes *Functor* $F : AA \longrightarrow BB$
and *Functor* $G : AA \longrightarrow BB$
and $u : ob\ AA \to ar\ BB$
and $u \in extensional\ (ob\ AA)$
and $\forall A{\in}Ob.\ u\ A \in Hom_{BB}\ (F_o\ A)\ (G_o\ A)$
and $\forall A{\in}Ob.\ \forall B{\in}Ob.\ \forall f{\in}Hom\ A\ B.\ (G_a\ f) \cdot_{BB} (u\ A) = (u\ B) \cdot_{BB} (F_a\ f)$

abbreviation
nt-syn (‹- : - ⇒ - *in Func* ′(- , - ′)› [*81*]) **where**
$u : F \Rightarrow G$ *in Func*$(AA,\ BB) \equiv$ *natural-transformation* $AA\ BB\ F\ G\ u$

locale *endoNT* = *natural-transformation* + *one-cat*

theorem (**in** *endoNT*) *id-restrict-natural*:
$(\lambda A{\in}Ob.\ Id\ A) : (id\text{-}func\ AA) \Rightarrow (id\text{-}func\ AA)$ *in Func*$(AA{,}AA)$
⟨*proof*⟩

end

6 Yonedas Lemma

theory *Yoneda*
imports *HomFunctors NatTrans*
begin

6.1 The Sandwich Natural Transformation

locale *Yoneda* = *functor* + *into-set* +
assumes *TERM* $(AA :: ('o,'a,'m)$*category-scheme*$)$
fixes *sandwich* :: $['o,'a,'o] \Rightarrow 'a$ *set-arrow* (‹σ′(-,-′)›)
defines *sandwich* $A\ a \equiv (\lambda B{\in}Ob.$ ⦇
set-dom=*Hom* $A\ B$,
set-func=$(\lambda f{\in}Hom\ A\ B.$ *set-func* $(F_a\ f)\ a)$,
set-cod=$F_o\ B$
⦈)
fixes *unsandwich* :: $['o,'o \Rightarrow 'a$ *set-arrow*$] \Rightarrow 'a$ (‹$\sigma^{\leftarrow}$′(-,-′)›)
defines *unsandwich* $A\ u \equiv$ *set-func* $(u\ A)\ (Id\ A)$

lemma (**in** *Yoneda*) *F-into-set*:

Functor F : *AA* ⟶ *Set*
⟨*proof*⟩

lemma (**in** *Yoneda*) *F-comp-func*:
assumes *1*: $A \in Ob$ **and** *2*: $B \in Ob$ **and** *3*: $C \in Ob$
and *4*: $g \in Hom\ A\ B$ **and** *5*: $f \in Hom\ B\ C$
shows *set-func* (F_a ($f \cdot g$)) = *compose* (F_o A) (*set-func* (F_a f)) (*set-func* (F_a g))
⟨*proof*⟩

lemma (**in** *Yoneda*) *sandwich-funcset*:
assumes *A*: $A \in Ob$
and $a \in F_o\ A$
shows $\sigma(A,a) : Ob \rightarrow ar\ Set$
⟨*proof*⟩

lemma (**in** *Yoneda*) *sandwich-type*:
assumes *A*: $A \in Ob$ **and** *B*: $B \in Ob$
and $a \in F_o\ A$
shows $\sigma(A,a)\ B \in hom\ Set\ (Hom\ A\ B)\ (F_o\ B)$
⟨*proof*⟩

lemma (**in** *Yoneda*) *sandwich-commutes*:
assumes *AOb*: $A \in Ob$ **and** *BOb*: $B \in Ob$ **and** *COb*: $C \in Ob$
and *aFa*: $a \in F_o\ A$
and *fBC*: $f \in Hom\ B\ C$
shows $(F_a\ f) \odot (\sigma(A,a)\ B) = (\sigma(A,a)\ C) \odot (Hom(A,\text{-})_a\ f)$
⟨*proof*⟩

lemma (**in** *Yoneda*) *sandwich-natural*:
assumes $A \in Ob$
and $a \in F_o\ A$
shows $\sigma(A,a) : Hom(A,\text{-}) \Rightarrow F\ in\ Func(AA,Set)$
⟨*proof*⟩

6.2 Sandwich Components are Bijective

lemma (**in** *Yoneda*) *unsandwich-left-inverse*:
assumes *1*: $A \in Ob$
and *2*: $a \in F_o\ A$
shows $\sigma^{\leftarrow}(A,\sigma(A,a)) = a$
⟨*proof*⟩

lemma (**in** *Yoneda*) *unsandwich-right-inverse*:

assumes *1*: $A \in Ob$
and *2*: $u : Hom(A,\text{-}) \Rightarrow F \ in \ Func(AA,Set)$
shows $\sigma(A,\sigma^{\leftarrow}(A,u)) = u$
⟨*proof*⟩

In order to state the lemma, we must rectify a curious omission from the Isabelle/HOL library. They define the idea of injectivity on a given set, but surjectivity is only defined relative to the entire universe of the target type.

definition
surj-on :: $['a \Rightarrow 'b, \ 'a \ set, \ 'b \ set] \Rightarrow bool$ **where**
surj-on $f \ A \ B \longleftrightarrow (\forall y{\in}B. \ \exists x{\in}A. \ f(x){=}y)$

definition
bij-on :: $['a \Rightarrow 'b, \ 'a \ set, \ 'b \ set] \Rightarrow bool$ **where**
bij-on $f \ A \ B \longleftrightarrow$ *inj-on* $f \ A$ & *surj-on* $f \ A \ B$

definition
equinumerous :: $['a \ set, \ 'b \ set] \Rightarrow bool$ (**infix** ‹≅› *40*) **where**
equinumerous $A \ B \longleftrightarrow (\exists f.$ *bij-betw* $f \ A \ B)$

lemma *bij-betw-eq*:
bij-betw $f \ A \ B \longleftrightarrow$
inj-on $f \ A \wedge (\forall y{\in}B. \ \exists x{\in}A. \ f(x){=}y) \wedge (\forall x{\in}A. \ f \ x \in B)$
⟨*proof*⟩

theorem (**in** *Yoneda*) *Yoneda*:
assumes *1*: $A \in Ob$
shows $F_o \ A \cong \{u. \ u : Hom(A,\text{-}) \Rightarrow F \ in \ Func(AA,Set)\}$
⟨*proof*⟩

end

References

[O'K04] Greg O'Keefe. Towards a readable formalisation of category theory. In Mike Atkinson, editor, *Computing: The Australasian Theory Symposium*, volume 91 of *Electronic Notes in Theoretical Computer Science*, pages 212–228. Elsevier, 2004.