Category Theory for ZFC in HOL III
Universal Constructions for 1-Categories

Mihails Milehins

March 19, 2025

Abstract

This article provides a formalization of elements of the theory of universal constructions for 1-categories (such as limits, adjoints and Kan extensions) in the object logic *ZFC in HOL* ([13], also see [11]) of the formal proof assistant *Isabelle* [12].

Acknowledgements

The author would like to acknowledge the assistance that he received from the users of the mailing list of Isabelle in the form of answers given to his general queries. Special thanks go to Andreas Lochbihler for suggesting the use of the combination of unrestricted overloading and locales for structuring mathematical knowledge on the mailing list of Isabelle: the design pattern that is used in this study builds upon this idea. Special thanks also go to Thomas Sewell for suggesting a trick for rewriting expressions modulo associativity on the mailing list of Isabelle, which was used on numerous occasions throughout the development of this work. Furthermore, the author would like to mention that the tool "Sketch-and-Explore" [5] from the standard distribution of Isabelle was used extensively in the development of this work. Moreover, the author would like to acknowledge the positive role that numerous Q&A posted on the Stack Exchange network (especially Mathematics Stack Exchange, Stack Overflow and TeX Stack Exchange) played in the development of this work. Lastly, the author would like to express gratitude to all members of his family and friends for their continuous support.

Contents

1 Introduction

This article provides a formalization of further elements of the theory of 1-categories without an additional structure. More specifically, this article explores canonical universal constructions [1][1] and their properties, building upon the formalization of the foundations of category theory in [10].

[1] https://ncatlab.org/nlab/show/universal+construction

2 Universal arrow

2.1 Background

The following section is based, primarily, on the elements of the content of Chapter III-1 in [9].

named-theorems *ua-field-simps*

definition $UObj :: V$ **where** $[ua\text{-}field\text{-}simps]$: $UObj = 0$
definition $UArr :: V$ **where** $[ua\text{-}field\text{-}simps]$: $UArr = 1_{\mathbb{N}}$

lemma $[cat\text{-}cs\text{-}simps]$:
  shows $UObj\text{-}simp$: $[a, b]_\circ(\!|UObj|\!) = a$
    and $UArr\text{-}simp$: $[a, b]_\circ(\!|UArr|\!) = b$
  $\langle proof \rangle$

2.2 Universal map

The universal map is a convenience utility that allows treating a part of the definition of the universal arrow as an arrow in the category *Set*.

2.2.1 Definition and elementary properties

definition $umap\text{-}of :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ d =$
    $[$
      $(\lambda f' \in_\circ Hom\ (\mathfrak{F}(\!|HomDom|\!))\ r\ d.\ \mathfrak{F}(\!|ArrMap|\!)(\!|f'|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} u)$,
      $Hom\ (\mathfrak{F}(\!|HomDom|\!))\ r\ d$,
      $Hom\ (\mathfrak{F}(\!|HomCod|\!))\ c\ (\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!))$
    $]_\circ$

definition $umap\text{-}fo :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where $umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d = umap\text{-}of\ (op\text{-}cf\ \mathfrak{F})\ c\ r\ u\ d$

Components.

lemma (**in** *is-functor*) *umap-of-components*:
  assumes $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
  shows $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrVal|\!) = (\lambda f' \in_\circ Hom\ \mathfrak{A}\ r\ d.\ \mathfrak{F}(\!|ArrMap|\!)(\!|f'|\!) \circ_{A\mathfrak{B}} u)$
    and $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrDom|\!) = Hom\ \mathfrak{A}\ r\ d$
    and $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrCod|\!) = Hom\ \mathfrak{B}\ c\ (\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!))$
  $\langle proof \rangle$

lemma (**in** *is-functor*) *umap-fo-components*:
  assumes $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
  shows $umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrVal|\!) = (\lambda f' \in_\circ Hom\ \mathfrak{A}\ d\ r.\ u \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f'|\!))$
    and $umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrDom|\!) = Hom\ \mathfrak{A}\ d\ r$
    and $umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d(\!|ArrCod|\!) = Hom\ \mathfrak{B}\ (\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!))\ c$
  $\langle proof \rangle$

Universal maps for the opposite functor.

lemma (**in** *is-functor*) $op\text{-}umap\text{-}of[cat\text{-}op\text{-}simps]$: $umap\text{-}of\ (op\text{-}cf\ \mathfrak{F}) = umap\text{-}fo\ \mathfrak{F}$
  $\langle proof \rangle$

lemma (**in** *is-functor*) $op\text{-}umap\text{-}fo[cat\text{-}op\text{-}simps]$: $umap\text{-}fo\ (op\text{-}cf\ \mathfrak{F}) = umap\text{-}of\ \mathfrak{F}$
  $\langle proof \rangle$

lemmas $[cat\text{-}op\text{-}simps]$ =

is-functor.op-umap-of
is-functor.op-umap-fo

2.2.2 Arrow value

lemma *umap-of-ArrVal-vsv*[*cat-cs-intros*]: *vsv* (*umap-of* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$)
⟨*proof*⟩

lemma *umap-fo-ArrVal-vsv*[*cat-cs-intros*]: *vsv* (*umap-fo* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$)
⟨*proof*⟩

lemma (**in** *is-functor*) *umap-of-ArrVal-vdomain*:
assumes $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
shows $\mathcal{D}_\circ$ (*umap-of* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$) = *Hom* $\mathfrak{A}$ *r d*
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.umap-of-ArrVal-vdomain*

lemma (**in** *is-functor*) *umap-fo-ArrVal-vdomain*:
assumes $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
shows $\mathcal{D}_\circ$ (*umap-fo* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$) = *Hom* $\mathfrak{A}$ *d r*
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.umap-fo-ArrVal-vdomain*

lemma (**in** *is-functor*) *umap-of-ArrVal-app*:
assumes $f' : r \mapsto_{\mathfrak{A}} d$ **and** $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
shows *umap-of* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)(\!|f'|\!)$ = $\mathfrak{F}(\!|ArrMap|\!)(\!|f'|\!) \circ_{A\mathfrak{B}} u$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.umap-of-ArrVal-app*

lemma (**in** *is-functor*) *umap-fo-ArrVal-app*:
assumes $f' : d \mapsto_{\mathfrak{A}} r$ **and** $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
shows *umap-fo* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)(\!|f'|\!)$ = $u \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f'|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.umap-fo-ArrVal-app*

lemma (**in** *is-functor*) *umap-of-ArrVal-vrange*:
assumes $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
shows $\mathcal{R}_\circ$ (*umap-of* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$) $\subseteq_\circ$ *Hom* $\mathfrak{B}$ *c* ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$)
⟨*proof*⟩

lemma (**in** *is-functor*) *umap-fo-ArrVal-vrange*:
assumes $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
shows $\mathcal{R}_\circ$ (*umap-fo* $\mathfrak{F}$ *c r u d*$(\!|$*ArrVal*$|\!)$) $\subseteq_\circ$ *Hom* $\mathfrak{B}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$) *c*
⟨*proof*⟩

2.2.3 Universal map is an arrow in the category *Set*

lemma (**in** *is-functor*) *cf-arr-Set-umap-of*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and *r*: $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
and *d*: $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
and *u*: $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
shows *arr-Set* α (*umap-of* $\mathfrak{F}$ *c r u d*)

⟨*proof*⟩

lemma (**in** *is-functor*) *cf-arr-Set-umap-fo*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and *r*: $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and *d*: $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and *u*: $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
  shows *arr-Set* α (*umap-fo* $\mathfrak{F}$ *c r u d*)
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-umap-of-is-arr*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
  shows *umap-of* $\mathfrak{F}$ *c r u d* : *Hom* $\mathfrak{A}$ *r d* $\mapsto_{cat\text{-}Set\ \alpha}$ *Hom* $\mathfrak{B}$ *c* ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$)
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-umap-of-is-arr'*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
    and A = *Hom* $\mathfrak{A}$ *r d*
    and B = *Hom* $\mathfrak{B}$ *c* ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$)
    and $\mathfrak{C}$ = *cat-Set* α
  shows *umap-of* $\mathfrak{F}$ *c r u d* : $A \mapsto_{\mathfrak{C}} B$
  ⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-umap-of-is-arr'*

lemma (**in** *is-functor*) *cf-umap-fo-is-arr*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
  shows *umap-fo* $\mathfrak{F}$ *c r u d* : *Hom* $\mathfrak{A}$ *d r* $\mapsto_{cat\text{-}Set\ \alpha}$ *Hom* $\mathfrak{B}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$) *c*
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-umap-fo-is-arr'*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $d \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
    and A = *Hom* $\mathfrak{A}$ *d r*
    and B = *Hom* $\mathfrak{B}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|d|\!)$) *c*
    and $\mathfrak{C}$ = *cat-Set* α
  shows *umap-fo* $\mathfrak{F}$ *c r u d* : $A \mapsto_{\mathfrak{C}} B$
  ⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-umap-fo-is-arr'*

2.3 Universal arrow: definition and elementary properties

See Chapter III-1 in [9].

definition *universal-arrow-of* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *universal-arrow-of* $\mathfrak{F}$ c r u $\longleftrightarrow$
(
$r \in_\circ \mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!) \wedge$
$u : c \mapsto_{\mathfrak{F}(\!|HomCod|\!)} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \wedge$
(
$\forall r'\ u'.$
$r' \in_\circ \mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!) \longrightarrow$
$u' : c \mapsto_{\mathfrak{F}(\!|HomCod|\!)} \mathfrak{F}(\!|ObjMap|\!)(\!|r'|\!) \longrightarrow$
$(\exists!f'.\ f' : r \mapsto_{\mathfrak{F}(\!|HomDom|\!)} r' \wedge u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!))$
)
)

definition *universal-arrow-fo* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *universal-arrow-fo* $\mathfrak{F}$ c r u $\equiv$ *universal-arrow-of* (*op-cf* $\mathfrak{F}$) c r u

Rules.

mk-ide (**in** *is-functor*) **rf**
universal-arrow-of-def[**where** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cf-HomDom cf-HomCod*]
|*intro universal-arrow-ofI*|
|*dest universal-arrow-ofD*[*dest*]|
|*elim universal-arrow-ofE*[*elim*]|

lemma (**in** *is-functor*) *universal-arrow-foI*:
assumes $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r'\ u'.\ [\![\ r' \in_\circ \mathfrak{A}(\!|Obj|\!);\ u' : \mathfrak{F}(\!|ObjMap|\!)(\!|r'|\!) \mapsto_{\mathfrak{B}} c\ ]\!] \Longrightarrow$
$\exists!f'.\ f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!)$
shows *universal-arrow-fo* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *universal-arrow-foD*[*dest*]:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
shows $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r'\ u'.\ [\![\ r' \in_\circ \mathfrak{A}(\!|Obj|\!);\ u' : \mathfrak{F}(\!|ObjMap|\!)(\!|r'|\!) \mapsto_{\mathfrak{B}} c\ ]\!] \Longrightarrow$
$\exists!f'.\ f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *universal-arrow-foE*[*elim*]:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
obtains $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $u : \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r'\ u'.\ [\![\ r' \in_\circ \mathfrak{A}(\!|Obj|\!);\ u' : \mathfrak{F}(\!|ObjMap|\!)(\!|r'|\!) \mapsto_{\mathfrak{B}} c\ ]\!] \Longrightarrow$
$\exists!f'.\ f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!)$
⟨*proof*⟩

Elementary properties.

lemma (**in** *is-functor*) *op-cf-universal-arrow-of*[*cat-op-simps*]:
universal-arrow-of (*op-cf* $\mathfrak{F}$) c r u $\longleftrightarrow$ *universal-arrow-fo* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *op-cf-universal-arrow-fo*[*cat-op-simps*]:
universal-arrow-fo (*op-cf* $\mathfrak{F}$) c r u $\longleftrightarrow$ *universal-arrow-of* $\mathfrak{F}$ c r u

⟨*proof*⟩

lemmas (**in** *is-functor*) [*cat-op-simps*] =
is-functor.op-cf-universal-arrow-of
is-functor.op-cf-universal-arrow-fo

2.4 Uniqueness

The following properties are related to the uniqueness of the universal arrow. These properties can be inferred from the content of Chapter III-1 in [9].

lemma (**in** *is-functor*) *cf-universal-arrow-of-ex-is-iso-arr*:
— The proof is based on the ideas expressed in the proof of Theorem 5.2 in Chapter Introduction in [6].
assumes *universal-arrow-of* $\mathfrak{F}$ c r u **and** *universal-arrow-of* $\mathfrak{F}$ c r' u'
obtains f **where** $f : r \mapsto_{iso\mathfrak{A}} r'$ **and** $u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-fo-ex-is-iso-arr*:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
and *universal-arrow-fo* $\mathfrak{F}$ c r' u'
obtains f **where** $f : r' \mapsto_{iso\mathfrak{A}} r$ **and** $u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-of-unique*:
assumes *universal-arrow-of* $\mathfrak{F}$ c r u
and *universal-arrow-of* $\mathfrak{F}$ c r' u'
shows $\exists!f'.\ f' : r \mapsto_{\mathfrak{A}} r' \wedge u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-fo-unique*:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
and *universal-arrow-fo* $\mathfrak{F}$ c r' u'
shows $\exists!f'.\ f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-of-is-iso-arr*:
assumes *universal-arrow-of* $\mathfrak{F}$ c r u
and *universal-arrow-of* $\mathfrak{F}$ c r' u'
and $f : r \mapsto_{\mathfrak{A}} r'$
and $u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
shows $f : r \mapsto_{iso\mathfrak{A}} r'$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-fo-is-iso-arr*:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
and *universal-arrow-fo* $\mathfrak{F}$ c r' u'
and $f : r' \mapsto_{\mathfrak{A}} r$
and $u' = umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
shows $f : r' \mapsto_{iso\mathfrak{A}} r$
⟨*proof*⟩

lemma (**in** *is-functor*) *universal-arrow-of-if-universal-arrow-of*:
assumes *universal-arrow-of* $\mathfrak{F}$ c r u
and $f : r \mapsto_{iso\mathfrak{A}} r'$
and $u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
shows *universal-arrow-of* $\mathfrak{F}$ c r' u'
⟨*proof*⟩

lemma (**in** *is-functor*) *universal-arrow-fo-if-universal-arrow-fo*:
  assumes *universal-arrow-fo* $\mathfrak{F}$ *c r u*
    and $f : r' \mapsto_{iso\mathfrak{A}} r$
    and $u' = \textit{umap-fo}\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f|\!)$
  shows *universal-arrow-fo* $\mathfrak{F}$ $c\ r'\ u'$
  ⟨*proof*⟩

2.5 Universal natural transformation

2.5.1 Definition and elementary properties

The concept of the universal natural transformation is introduced for the statement of the elements of a variant of Proposition 1 in Chapter III-2 in [9].

definition *ntcf-ua-of* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u$ =
    [
      $(\lambda d{\in_\circ}\mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!).\ \textit{umap-of}\ \mathfrak{F}\ c\ r\ u\ d)$,
      $Hom_{O.C\alpha}\mathfrak{F}(\!|HomDom|\!)(r,-)$,
      $Hom_{O.C\alpha}\mathfrak{F}(\!|HomCod|\!)(c,-) \circ_{CF} \mathfrak{F}$,
      $\mathfrak{F}(\!|HomDom|\!)$,
      cat-Set α
    ]$_\circ$

definition *ntcf-ua-fo* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u$ = *ntcf-ua-of* α (*op-cf* $\mathfrak{F}$) *c r u*

Components.

lemma *ntcf-ua-of-components*:
  shows *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTMap|\!) = (\lambda d{\in_\circ}\mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!).\ \textit{umap-of}\ \mathfrak{F}\ c\ r\ u\ d)$
    and *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDom|\!) = Hom_{O.C\alpha}\mathfrak{F}(\!|HomDom|\!)(r,-)$
    and *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTCod|\!) = Hom_{O.C\alpha}\mathfrak{F}(\!|HomCod|\!)(c,-) \circ_{CF} \mathfrak{F}$
    and *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDGDom|\!) = \mathfrak{F}(\!|HomDom|\!)$
    and *ntcf-ua-of* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDGCod|\!)$ = *cat-Set* α
  ⟨*proof*⟩

lemma *ntcf-ua-fo-components*:
  shows *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTMap|\!) = (\lambda d{\in_\circ}\mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!).\ \textit{umap-fo}\ \mathfrak{F}\ c\ r\ u\ d)$
    and *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDom|\!) = Hom_{O.C\alpha}\textit{op-cat}\ (\mathfrak{F}(\!|HomDom|\!))(r,-)$
    and *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTCod|\!)$ =
      $Hom_{O.C\alpha}\textit{op-cat}\ (\mathfrak{F}(\!|HomCod|\!))(c,-) \circ_{CF} \textit{op-cf}\ \mathfrak{F}$
    and *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDGDom|\!) = \textit{op-cat}\ (\mathfrak{F}(\!|HomDom|\!))$
    and *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTDGCod|\!)$ = *cat-Set* α
  ⟨*proof*⟩

context *is-functor*
begin

lemmas *ntcf-ua-of-components'* =
  ntcf-ua-of-components[**where** α=α **and** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cat-cs-simps*]

lemmas [*cat-cs-simps*] = *ntcf-ua-of-components'*(*2–5*)

lemma *ntcf-ua-fo-components'*:
  assumes $c \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $r \in_\circ \mathfrak{A}(\!|Obj|\!)$
  shows *ntcf-ua-fo* $\alpha\ \mathfrak{F}\ c\ r\ u(\!|NTMap|\!) = (\lambda d{\in_\circ}\mathfrak{A}(\!|Obj|\!).\ \textit{umap-fo}\ \mathfrak{F}\ c\ r\ u\ d)$
    and [*cat-cs-simps*]:

ntcf-ua-fo α $\mathfrak{F}$ *c r u*⦇*NTDom*⦈ = $Hom_{O.C\alpha}\mathfrak{A}(-,r)$
and [*cat-cs-simps*]:
ntcf-ua-fo α $\mathfrak{F}$ *c r u*⦇*NTCod*⦈ = $Hom_{O.C\alpha}\mathfrak{B}(-,c) \circ_{CF}$ *op-cf* $\mathfrak{F}$
and [*cat-cs-simps*]: *ntcf-ua-fo* α $\mathfrak{F}$ *c r u*⦇*NTDGDom*⦈ = *op-cat* $\mathfrak{A}$
and [*cat-cs-simps*]: *ntcf-ua-fo* α $\mathfrak{F}$ *c r u*⦇*NTDGCod*⦈ = *cat-Set* α
⟨*proof*⟩

end

lemmas [*cat-cs-simps*] =
is-functor.ntcf-ua-of-components′(*2–5*)
is-functor.ntcf-ua-fo-components′(*2–5*)

2.5.2 Natural transformation map

mk-VLambda (**in** *is-functor*)
ntcf-ua-of-components(*1*)[**where** α=α **and** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cf-HomDom*]
|*vsv ntcf-ua-of-NTMap-vsv*|
|*vdomain ntcf-ua-of-NTMap-vdomain*|
|*app ntcf-ua-of-NTMap-app*|

context *is-functor*
begin

context
fixes *c r*
assumes *r*: *r* $\in_\circ$ $\mathfrak{A}$⦇*Obj*⦈ **and** *c*: *c* $\in_\circ$ $\mathfrak{B}$⦇*Obj*⦈
begin

mk-VLambda *ntcf-ua-fo-components*′(*1*)[*OF c r*]
|*vsv ntcf-ua-fo-NTMap-vsv*|
|*vdomain ntcf-ua-fo-NTMap-vdomain*|
|*app ntcf-ua-fo-NTMap-app*|

end

end

lemmas [*cat-cs-intros*] =
is-functor.ntcf-ua-fo-NTMap-vsv
is-functor.ntcf-ua-of-NTMap-vsv

lemmas [*cat-cs-simps*] =
is-functor.ntcf-ua-fo-NTMap-vdomain
is-functor.ntcf-ua-fo-NTMap-app
is-functor.ntcf-ua-of-NTMap-vdomain
is-functor.ntcf-ua-of-NTMap-app

lemma (**in** *is-functor*) *ntcf-ua-of-NTMap-vrange*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and *r* $\in_\circ$ $\mathfrak{A}$⦇*Obj*⦈
and *u* : *c* $\mapsto_{\mathfrak{B}}$ $\mathfrak{F}$⦇*ObjMap*⦈⦇*r*⦈
shows $\mathcal{R}_\circ$ (*ntcf-ua-of* α $\mathfrak{F}$ *c r u*⦇*NTMap*⦈) $\subseteq_\circ$ *cat-Set* α⦇*Arr*⦈
⟨*proof*⟩

2.5.3 Commutativity of the universal maps and *hom*-functions

lemma (**in** *is-functor*) *cf-umap-of-cf-hom-commute*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $c \in_\circ \mathfrak{B}(|Obj|)$
    and $r \in_\circ \mathfrak{A}(|Obj|)$
    and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(|ObjMap|)(|r|)$
    and $f : a \mapsto_{\mathfrak{A}} b$
  shows
    $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ b \circ_{A\,cat\text{-}Set\ \alpha} cf\text{-}hom\ \mathfrak{A}\ [\mathfrak{A}(|CId|)(|r|), f]_\circ =$
      $cf\text{-}hom\ \mathfrak{B}\ [\mathfrak{B}(|CId|)(|c|), \mathfrak{F}(|ArrMap|)(|f|)]_\circ \circ_{A\,cat\text{-}Set\ \alpha} umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ a$
  (**is** ‹$?uof\text{-}b \circ_{A\,cat\text{-}Set\ \alpha} ?rf = ?cf \circ_{A\,cat\text{-}Set\ \alpha} ?uof\text{-}a$›)
⟨*proof*⟩

lemma *cf-umap-of-cf-hom-unit-commute*:
  assumes *category* α $\mathfrak{C}$
    and *category* α $\mathfrak{D}$
    and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\eta : cf\text{-}id\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $g : c' \mapsto_{\mathfrak{C}} c$
    and $f : d \mapsto_{\mathfrak{D}} d'$
  shows
    $umap\text{-}of\ \mathfrak{G}\ c'\ (\mathfrak{F}(|ObjMap|)(|c'|))\ (\eta(|NTMap|)(|c'|))\ d' \circ_{A\,cat\text{-}Set\ \alpha}$
      $cf\text{-}hom\ \mathfrak{D}\ [\mathfrak{F}(|ArrMap|)(|g|), f]_\circ =$
        $cf\text{-}hom\ \mathfrak{C}\ [g, \mathfrak{G}(|ArrMap|)(|f|)]_\circ \circ_{A\,cat\text{-}Set\ \alpha}$
          $umap\text{-}of\ \mathfrak{G}\ c\ (\mathfrak{F}(|ObjMap|)(|c|))\ (\eta(|NTMap|)(|c|))\ d$
  (**is** ‹$?uof\text{-}c'd' \circ_{A\,cat\text{-}Set\ \alpha} ?\mathfrak{F}gf = ?g\mathfrak{G}f \circ_{A\,cat\text{-}Set\ \alpha} ?uof\text{-}cd$›)
⟨*proof*⟩

2.5.4 Universal natural transformation is a natural transformation

lemma (**in** *is-functor*) *cf-ntcf-ua-of-is-ntcf*:
  assumes $r \in_\circ \mathfrak{A}(|Obj|)$
    and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(|ObjMap|)(|r|)$
  shows $ntcf\text{-}ua\text{-}of\ \alpha\ \mathfrak{F}\ c\ r\ u$:
    $Hom_{O.C\alpha}\mathfrak{A}(r,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{B}(c,-) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ntcf-ua-fo-is-ntcf*:
  assumes $r \in_\circ \mathfrak{A}(|Obj|)$ **and** $u : \mathfrak{F}(|ObjMap|)(|r|) \mapsto_{\mathfrak{B}} c$
  shows $ntcf\text{-}ua\text{-}fo\ \alpha\ \mathfrak{F}\ c\ r\ u$:
    $Hom_{O.C\alpha}\mathfrak{A}(-,r) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{B}(-,c) \circ_{CF} op\text{-}cf\ \mathfrak{F}$:
    $op\text{-}cat\ \mathfrak{A} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

2.5.5 Universal natural transformation and universal arrow

The lemmas in this subsection correspond to variants of elements of Proposition 1 in Chapter III-2 in [9].

lemma (**in** *is-functor*) *cf-ntcf-ua-of-is-iso-ntcf*:
  assumes *universal-arrow-of* $\mathfrak{F}$ c r u
  shows $ntcf\text{-}ua\text{-}of\ \alpha\ \mathfrak{F}\ c\ r\ u$:
    $Hom_{O.C\alpha}\mathfrak{A}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(c,-) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-ntcf-ua-of-is-iso-ntcf*

lemma (**in** *is-functor*) *cf-ntcf-ua-fo-is-iso-ntcf*:
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
shows *ntcf-ua-fo* α $\mathfrak{F}$ c r u :
$Hom_{O.C\alpha}\mathfrak{A}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(-,c) \circ_{CF}$ *op-cf* $\mathfrak{F}$:
op-cat $\mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-ntcf-ua-fo-is-iso-ntcf*

lemma (**in** *is-functor*) *cf-ua-of-if-ntcf-ua-of-is-iso-ntcf*:
assumes $r \in_\circ \mathfrak{A}(|Obj|)$
and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(|ObjMap|)(|r|)$
and *ntcf-ua-of* α $\mathfrak{F}$ c r u :
$Hom_{O.C\alpha}\mathfrak{A}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(c,-) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows *universal-arrow-of* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ua-fo-if-ntcf-ua-fo-is-iso-ntcf*:
assumes $r \in_\circ \mathfrak{A}(|Obj|)$
and $u : \mathfrak{F}(|ObjMap|)(|r|) \mapsto_{\mathfrak{B}} c$
and *ntcf-ua-fo* α $\mathfrak{F}$ c r u :
$Hom_{O.C\alpha}\mathfrak{A}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(-,c) \circ_{CF}$ *op-cf* $\mathfrak{F}$:
op-cat $\mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows *universal-arrow-fo* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-of-if-is-iso-ntcf*:
assumes $r \in_\circ \mathfrak{A}(|Obj|)$
and $c \in_\circ \mathfrak{B}(|Obj|)$
and φ :
$Hom_{O.C\alpha}\mathfrak{A}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(c,-) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows *universal-arrow-of* $\mathfrak{F}$ c r $(\varphi(|NTMap|)(|r|)(|ArrVal|)(|\mathfrak{A}(|CId|)(|r|)|))$
(**is** ‹*universal-arrow-of* $\mathfrak{F}$ c r *?u*›)
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-universal-arrow-fo-if-is-iso-ntcf*:
assumes $r \in_\circ \mathfrak{A}(|Obj|)$
and $c \in_\circ \mathfrak{B}(|Obj|)$
and φ :
$Hom_{O.C\alpha}\mathfrak{A}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{B}(-,c) \circ_{CF}$ *op-cf* $\mathfrak{F}$:
op-cat $\mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows *universal-arrow-fo* $\mathfrak{F}$ c r $(\varphi(|NTMap|)(|r|)(|ArrVal|)(|\mathfrak{A}(|CId|)(|r|)|))$
⟨*proof*⟩

3 Limits and colimits

3.1 Background

named-theorems *cat-lim-cs-simps*
named-theorems *cat-lim-cs-intros*

3.2 Limit and colimit

3.2.1 Definition and elementary properties

The concept of a limit is introduced in Chapter III-4 in [9]; the concept of a colimit is introduced in Chapter III-3 in [9].

locale *is-cat-limit* = *is-cat-cone* $\alpha\ r\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ u$ **for** $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$ +
  assumes *cat-lim-ua-fo*: $\bigwedge u'\ r'.\ u' : r' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
    $\exists!f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \cdot_{NTCF}$ *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f'$

syntax *-is-cat-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $<_{CF.lim}$ - :/ - $\mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-limit* ⇌ *is-cat-limit*
translations $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
  CONST is-cat-limit $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$

locale *is-cat-colimit* = *is-cat-cocone* $\alpha\ r\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ u$ **for** $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$ +
  assumes *cat-colim-ua-of*: $\bigwedge u'\ r'.\ u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
    $\exists!f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge u' =$ *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f' \cdot_{NTCF} u$

syntax *-is-cat-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $>_{CF.colim}$ - :/ - $\mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-colimit* ⇌ *is-cat-colimit*
translations $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
  CONST is-cat-colimit $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$

Rules.

lemma (**in** *is-cat-limit*) *is-cat-limit-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $r' = r$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
  shows $u : r' <_{CF.lim} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-limit-def*[*unfolded is-cat-limit-axioms-def*]
  |*intro is-cat-limitI*|
  |*dest is-cat-limitD*[*dest*]|
  |*elim is-cat-limitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-limitD*(*1*)

lemma (**in** *is-cat-colimit*) *is-cat-colimit-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $r' = r$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
  shows $u : \mathfrak{F}' >_{CF.colim} r' : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-colimit-def*[*unfolded is-cat-colimit-axioms-def*]
  |*intro is-cat-colimitI*|
  |*dest is-cat-colimitD*[*dest*]|
  |*elim is-cat-colimitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-colimitD*(*1*)

Limits, colimits and universal arrows.

lemma (**in** *is-cat-limit*) *cat-lim-is-universal-arrow-fo*:
universal-arrow-fo $(\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C})$ (*cf-map* $\mathfrak{F}$) r (*ntcf-arrow* u)
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *cat-cone-is-cat-limit*:
assumes *universal-arrow-fo* $(\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C})$ (*cf-map* $\mathfrak{F}$) c (*ntcf-arrow* $\mathfrak{N}$)
shows $\mathfrak{N} : c <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-is-universal-arrow-of*:
universal-arrow-of $(\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C})$ (*cf-map* $\mathfrak{F}$) r (*ntcf-arrow* u)
⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *cat-cocone-is-cat-colimit*:
assumes *universal-arrow-of* $(\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C})$ (*cf-map* $\mathfrak{F}$) c (*ntcf-arrow* $\mathfrak{N}$)
shows $\mathfrak{N} : \mathfrak{F} >_{CF.colim} c : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

Duality.

lemma (**in** *is-cat-limit*) *is-cat-colimit-op*:
op-ntcf u : *op-cf* $\mathfrak{F} >_{CF.colim} r$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-limit*) *is-cat-colimit-op'*[*cat-op-intros*]:
assumes $\mathfrak{F}' =$ *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}' =$ *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : \mathfrak{F}' >_{CF.colim} r : \mathfrak{J}' \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-limit.is-cat-colimit-op'*

lemma (**in** *is-cat-colimit*) *is-cat-limit-op*:
op-ntcf $u : r <_{CF.lim}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *is-cat-colimit-op'*[*cat-op-intros*]:
assumes $\mathfrak{F}' =$ *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}' =$ *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : r <_{CF.lim} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-colimit.is-cat-colimit-op'*

3.2.2 Universal property

lemma (**in** *is-cat-limit*) *cat-lim-unique-cone'*:
assumes $u' : r' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists!f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge (\forall j \in_{\circ} \mathfrak{J}(\|Obj\|).\ u'(\|NTMap\|)(\|j\|) = u(\|NTMap\|)(\|j\|) \circ_{A\mathfrak{C}} f')$
⟨*proof*⟩

lemma (**in** *is-cat-limit*) *cat-lim-unique*:
assumes $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \bullet_{NTCF}$ *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (**in** *is-cat-limit*) *cat-lim-unique'*:
assumes $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows

$\exists!f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge (\forall j \in_\circ \mathfrak{J}(|Obj|).\ u'(|NTMap|)(|j|) = u(|NTMap|)(|j|) \circ_{A\mathfrak{C}} f')$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-unique-cocone*:
assumes $u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \bullet_{NTCF} u$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-unique-cocone′*:
assumes $u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists!f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge (\forall j \in_\circ \mathfrak{J}(|Obj|).\ u'(|NTMap|)(|j|) = f' \circ_{A\mathfrak{C}} u(|NTMap|)(|j|))$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-unique*:
assumes $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \bullet_{NTCF} u$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-unique′*:
assumes $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists!f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge (\forall j \in_\circ \mathfrak{J}(|Obj|).\ u'(|NTMap|)(|j|) = f' \circ_{A\mathfrak{C}} u(|NTMap|)(|j|))$
⟨*proof*⟩

lemma *cat-lim-ex-is-iso-arr*:
assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : r' \mapsto_{iso\mathfrak{C}} r$ **and** $u' = u \bullet_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-lim-ex-is-iso-arr′*:
assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : r' \mapsto_{iso\mathfrak{C}} r$
and $\bigwedge j.\ j \in_\circ \mathfrak{J}(|Obj|) \Longrightarrow u'(|NTMap|)(|j|) = u(|NTMap|)(|j|) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-colim-ex-is-iso-arr*:
assumes $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : r \mapsto_{iso\mathfrak{C}} r'$ **and** $u' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f \bullet_{NTCF} u$
⟨*proof*⟩

lemma *cat-colim-ex-is-iso-arr′*:
assumes $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : r \mapsto_{iso\mathfrak{C}} r'$
and $\bigwedge j.\ j \in_\circ \mathfrak{J}(|Obj|) \Longrightarrow u'(|NTMap|)(|j|) = f \circ_{A\mathfrak{C}} u(|NTMap|)(|j|)$
⟨*proof*⟩

3.2.3 Further properties

lemma (**in** *is-cat-limit*) *cat-lim-is-cat-limit-if-is-iso-arr*:
assumes $f : r' \mapsto_{iso\mathfrak{C}} r$
shows $u \bullet_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-is-cat-colimit-if-is-iso-arr*:
assumes $f : r \mapsto_{iso\mathfrak{C}} r'$

shows *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f $\bullet_{NTCF}$ u : $\mathfrak{F}$ $>_{CF.colim}$ r' : $\mathfrak{J}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-cf-comp-is-cat-limit-if-is-iso-functor*:
assumes u : r $<_{CF.lim}$ $\mathfrak{F}$: $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ **and** $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.iso\alpha}$ $\mathfrak{B}$
shows u $\circ_{NTCF-CF}$ $\mathfrak{G}$: r $<_{CF.lim}$ $\mathfrak{F}$ $\circ_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-cf-comp-is-cat-limit-if-is-iso-functor'*[*cat-lim-cs-intros*]:
assumes u : r $<_{CF.lim}$ $\mathfrak{F}$: $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
and $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.iso\alpha}$ $\mathfrak{B}$
and $\mathfrak{A}'$ = $\mathfrak{F}$ $\circ_{CF}$ $\mathfrak{G}$
shows u $\circ_{NTCF-CF}$ $\mathfrak{G}$: r $<_{CF.lim}$ $\mathfrak{A}'$: $\mathfrak{A}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

3.3 Small limit and small colimit

3.3.1 Definition and elementary properties

The concept of a limit is introduced in Chapter III-4 in [9]; the concept of a colimit is introduced in Chapter III-3 in [9]. The definitions of small limits were tailored for ZFC in HOL.

locale *is-tm-cat-limit* = *is-tm-cat-cone* α r $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ u **for** α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u +
assumes *tm-cat-lim-ua-fo*:
$\bigwedge u'$ r'. u' : r' $<_{CF.cone}$ $\mathfrak{F}$: $\mathfrak{J}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ $\Longrightarrow$
$\exists!f'$. f' : r' $\mapsto_{\mathfrak{C}}$ r $\wedge$ u' = u $\bullet_{NTCF}$ *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f'

syntax *-is-tm-cat-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $<_{CF.tm.lim}$ - :/ - $\mapsto\mapsto_{C.tm}$1 -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-tm-cat-limit* ⇌ *is-tm-cat-limit*
translations u : r $<_{CF.tm.lim}$ $\mathfrak{F}$: $\mathfrak{J}$ $\mapsto\mapsto_{C.tm\alpha}$ $\mathfrak{C}$ ⇌
CONST is-tm-cat-limit α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u

locale *is-tm-cat-colimit* = *is-tm-cat-cocone* α r $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ u **for** α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u +
assumes *tm-cat-colim-ua-of*:
$\bigwedge u'$ r'. u' : $\mathfrak{F}$ $>_{CF.cocone}$ r' : $\mathfrak{J}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ $\Longrightarrow$
$\exists!f'$. f' : r $\mapsto_{\mathfrak{C}}$ r' $\wedge$ u' = *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f' $\bullet_{NTCF}$ u

syntax *-is-tm-cat-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $>_{CF.tm.colim}$ - :/ - $\mapsto\mapsto_{C.tm}$1 -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-tm-cat-colimit* ⇌ *is-tm-cat-colimit*
translations u : $\mathfrak{F}$ $>_{CF.tm.colim}$ r : $\mathfrak{J}$ $\mapsto\mapsto_{C.tm\alpha}$ $\mathfrak{C}$ ⇌
CONST is-tm-cat-colimit α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u

Rules.

lemma (in *is-tm-cat-limit*) *is-tm-cat-limit-axioms'*[*cat-lim-cs-intros*]:
assumes α' = α **and** r' = r **and** $\mathfrak{J}'$ = $\mathfrak{J}$ **and** $\mathfrak{C}'$ = $\mathfrak{C}$ **and** $\mathfrak{F}'$ = $\mathfrak{F}$
shows u : r' $<_{CF.tm.lim}$ $\mathfrak{F}'$: $\mathfrak{J}'$ $\mapsto\mapsto_{C.tm_{\alpha'}}$ $\mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-tm-cat-limit-def*[*unfolded is-tm-cat-limit-axioms-def*]
|*intro is-tm-cat-limitI*|
|*dest is-tm-cat-limitD*[*dest*]|
|*elim is-tm-cat-limitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-tm-cat-limitD*(*1*)

lemma (in *is-tm-cat-colimit*) *is-tm-cat-colimit-axioms'*[*cat-lim-cs-intros*]:

assumes $\alpha' = \alpha$ **and** $r' = r$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
shows $u : \mathfrak{F}' >_{CF.tm.colim} r' : \mathfrak{J}' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-tm-cat-colimit-def*[*unfolded is-tm-cat-colimit-axioms-def*]
|*intro is-tm-cat-colimitI*|
|*dest is-tm-cat-colimitD*[*dest*]|
|*elim is-tm-cat-colimitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-tm-cat-colimitD*(*1*)

lemma *is-tm-cat-limitI'*:
assumes $u : r <_{CF.tm.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge u'\ r'.\ u' : r' <_{CF.tm.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \Longrightarrow$
$\exists !f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \bullet_{NTCF} \textit{ntcf-const}\ \mathfrak{J}\ \mathfrak{C}\ f'$
shows $u : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *is-tm-cat-colimitI'*:
assumes $u : \mathfrak{F} >_{CF.tm.cocone} r : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge u'\ r'.\ u' : \mathfrak{F} >_{CF.tm.cocone} r' : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \Longrightarrow$
$\exists !f'.\ f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = \textit{ntcf-const}\ \mathfrak{J}\ \mathfrak{C}\ f' \bullet_{NTCF} u$
shows $u : \mathfrak{F} >_{CF.tm.colim} r : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

Elementary properties.

sublocale *is-tm-cat-limit* ⊆ *is-cat-limit*
⟨*proof*⟩

sublocale *is-tm-cat-colimit* ⊆ *is-cat-colimit*
⟨*proof*⟩

lemma (**in** *is-cat-limit*) *cat-lim-is-tm-cat-limit*:
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $u : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-colimit*) *cat-colim-is-tm-cat-colimit*:
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $u : \mathfrak{F} >_{CF.tm.colim} r : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

Limits, colimits and universal arrows.

lemma (**in** *is-tm-cat-limit*) *tm-cat-lim-is-universal-arrow-fo*:
universal-arrow-fo ($\Delta_{CF.tm}\ \alpha\ \mathfrak{J}\ \mathfrak{C}$) (*cf-map* $\mathfrak{F}$) r (*ntcf-arrow* u)
⟨*proof*⟩

lemma (**in** *is-tm-cat-cone*) *tm-cat-cone-is-tm-cat-limit*:
assumes *universal-arrow-fo* ($\Delta_{CF.tm}\ \alpha\ \mathfrak{J}\ \mathfrak{C}$) (*cf-map* $\mathfrak{F}$) c (*ntcf-arrow* $\mathfrak{N}$)
shows $\mathfrak{N} : c <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-colimit*) *tm-cat-colim-is-universal-arrow-of*:
universal-arrow-of ($\Delta_{CF.tm}\ \alpha\ \mathfrak{J}\ \mathfrak{C}$) (*cf-map* $\mathfrak{F}$) r (*ntcf-arrow* u)
⟨*proof*⟩

lemma (**in** *is-tm-cat-cocone*) *tm-cat-cocone-is-tm-cat-colimit*:
assumes *universal-arrow-of* ($\Delta_{CF.tm}\ \alpha\ \mathfrak{J}\ \mathfrak{C}$) (*cf-map* $\mathfrak{F}$) c (*ntcf-arrow* $\mathfrak{N}$)

shows $\mathfrak{N} : \mathfrak{F} >_{CF.tm.colim} c : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

Duality.

lemma (**in** *is-tm-cat-limit*) *is-tm-cat-colimit-op*:
op-ntcf u : *op-cf* $\mathfrak{F} >_{CF.tm.colim} r$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-limit*) *is-tm-cat-colimit-op'*[*cat-op-intros*]:
assumes $\mathfrak{F}' =$ *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}' =$ *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : \mathfrak{F}' >_{CF.tm.colim} r : \mathfrak{J}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-tm-cat-limit.is-tm-cat-colimit-op'*

lemma (**in** *is-tm-cat-colimit*) *is-tm-cat-limit-op*:
op-ntcf $u : r <_{CF.tm.lim}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-colimit*) *is-tm-cat-colimit-op'*[*cat-op-intros*]:
assumes $\mathfrak{F}' =$ *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}' =$ *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : r <_{CF.tm.lim} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-tm-cat-colimit.is-tm-cat-colimit-op'*

3.3.2 Further properties

lemma (**in** *is-tm-cat-limit*) *tm-cat-lim-is-tm-cat-limit-if-iso-arr*:
assumes $f : r' \mapsto_{iso\mathfrak{C}} r$
shows $u \cdot_{NTCF}$ *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f : r' <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-colimit*) *tm-cat-colim-is-tm-cat-colimit-if-iso-arr*:
assumes $f : r \mapsto_{iso\mathfrak{C}} r'$
shows *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f \cdot_{NTCF} u : \mathfrak{F} >_{CF.tm.colim} r' : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

3.4 Finite limit and finite colimit

locale *is-cat-finite-limit* =
is-cat-limit $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$ + *NTDom.HomDom*: *finite-category* $\alpha\ \mathfrak{J}$
for $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$

syntax *-is-cat-finite-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $<_{CF.lim.fin}$ - :/ - $\mapsto\mapsto_{C}$1 -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-finite-limit* ⇌ *is-cat-finite-limit*
translations $u : r <_{CF.lim.fin} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
CONST is-cat-finite-limit $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$

locale *is-cat-finite-colimit* =
is-cat-colimit $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$ + *NTDom.HomDom*: *finite-category* $\alpha\ \mathfrak{J}$
for $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}\ r\ u$

syntax *-is-cat-finite-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $>_{CF.colim.fin}$ - :/ - $\mapsto\mapsto_{C}$1 -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-finite-colimit* ⇌ *is-cat-finite-colimit*
translations $u : \mathfrak{F} >_{CF.colim.fin} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌

CONST is-cat-finite-colimit α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u

Rules.

lemma (**in** *is-cat-finite-limit*) *is-cat-finite-limit-axioms*′[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $r' = r$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
shows $u : r' <_{CF.lim.fin} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-finite-limit-def*
|*intro is-cat-finite-limitI*|
|*dest is-cat-finite-limitD*[*dest*]|
|*elim is-cat-finite-limitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-finite-limitD*

lemma (**in** *is-cat-finite-colimit*)
is-cat-finite-colimit-axioms′[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $r' = r$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
shows $u : \mathfrak{F}' >_{CF.colim.fin} r' : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-finite-colimit-def*[*unfolded is-cat-colimit-axioms-def*]
|*intro is-cat-finite-colimitI*|
|*dest is-cat-finite-colimitD*[*dest*]|
|*elim is-cat-finite-colimitE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-finite-colimitD*

Duality.

lemma (**in** *is-cat-finite-limit*) *is-cat-finite-colimit-op*:
op-ntcf u : *op-cf* $\mathfrak{F} >_{CF.colim.fin} r$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-finite-limit*) *is-cat-finite-colimit-op*′[*cat-op-intros*]:
assumes $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}'$ = *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : \mathfrak{F}' >_{CF.colim.fin} r : \mathfrak{J}' \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-finite-limit.is-cat-finite-colimit-op*′

lemma (**in** *is-cat-finite-colimit*) *is-cat-finite-limit-op*:
op-ntcf $u : r <_{CF.lim.fin}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{J} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-finite-colimit*) *is-cat-finite-colimit-op*′[*cat-op-intros*]:
assumes $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$ **and** $\mathfrak{J}'$ = *op-cat* $\mathfrak{J}$ **and** $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* $u : r <_{CF.lim.fin} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-finite-colimit.is-cat-finite-colimit-op*′

Elementary properties.

sublocale *is-cat-finite-limit* ⊆ *is-tm-cat-limit*
⟨*proof*⟩

sublocale *is-cat-finite-colimit* ⊆ *is-tm-cat-colimit*
⟨*proof*⟩

3.5 Creation of limits

See Chapter V-1 in [9].

```
definition cf-creates-limits :: V ⇒ V ⇒ V ⇒ bool
  where cf-creates-limits α 𝔊 𝔉 =
    (
      ∀τ b.
        τ : b <CF.lim 𝔊 ∘CF 𝔉 : 𝔉⦇HomDom⦈ ↦↦Cα 𝔊⦇HomCod⦈ ⟶
        (
          (
            ∃!σa. ∃σ a. σa = ⟨σ, a⟩ ∧
              σ : a <CF.cone 𝔉 : 𝔉⦇HomDom⦈ ↦↦Cα 𝔉⦇HomCod⦈ ∧
              τ = 𝔊 ∘CF-NTCF σ ∧
              b = 𝔊⦇ObjMap⦈⦇a⦈
          ) ∧
          (
            ∀σ a.
              σ : a <CF.cone 𝔉 : 𝔉⦇HomDom⦈ ↦↦Cα 𝔉⦇HomCod⦈ ⟶
              τ = 𝔊 ∘CF-NTCF σ ⟶
              b = 𝔊⦇ObjMap⦈⦇a⦈ ⟶
              σ : a <CF.lim 𝔉 : 𝔉⦇HomDom⦈ ↦↦Cα 𝔉⦇HomCod⦈
          )
        )
    )
```

Rules.

```
context
  fixes α 𝔍 𝔄 𝔅 𝔊 𝔉
  assumes 𝔉: 𝔉 : 𝔍 ↦↦Cα 𝔄
    and 𝔊: 𝔊 : 𝔄 ↦↦Cα 𝔅
begin

interpretation 𝔉: is-functor α 𝔍 𝔄 𝔉 ⟨proof⟩
interpretation 𝔊: is-functor α 𝔄 𝔅 𝔊 ⟨proof⟩

mk-ide rf cf-creates-limits-def[
  where α=α and 𝔉=𝔉 and 𝔊=𝔊, unfolded cat-cs-simps
  ]
  |intro cf-creates-limitsI|
  |dest cf-creates-limitsD'|
  |elim cf-creates-limitsE'|

end

lemmas cf-creates-limitsD[dest!] = cf-creates-limitsD'[rotated 2]
  and cf-creates-limitsE[elim!] = cf-creates-limitsE'[rotated 2]

lemma cf-creates-limitsE'':
  assumes cf-creates-limits α 𝔊 𝔉
    and τ : b <CF.lim 𝔊 ∘CF 𝔉 : 𝔍 ↦↦Cα 𝔅
    and 𝔉 : 𝔍 ↦↦Cα 𝔄
    and 𝔊 : 𝔄 ↦↦Cα 𝔅
  obtains σ r where σ : r <CF.lim 𝔉 : 𝔍 ↦↦Cα 𝔄
    and τ = 𝔊 ∘CF-NTCF σ
    and b = 𝔊⦇ObjMap⦈⦇r⦈
⟨proof⟩
```

3.6 Preservation of limits and colimits

3.6.1 Definitions and elementary properties

See Chapter V-4 in [9].

definition *cf-preserves-limits* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *cf-preserves-limits* α $\mathfrak{G}$ $\mathfrak{F}$ =
(
$\forall \sigma\ a.$
$\sigma : a <_{CF.lim} \mathfrak{F} : \mathfrak{F}(\!|HomDom|\!) \mapsto\mapsto_{C\alpha} \mathfrak{F}(\!|HomCod|\!) \longrightarrow$
$\mathfrak{G} \circ_{CF-NTCF} \sigma : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) <_{CF.lim} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{F}(\!|HomDom|\!) \mapsto\mapsto_{C\alpha} \mathfrak{G}(\!|HomCod|\!)$
)

definition *cf-preserves-colimits* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *cf-preserves-colimits* α $\mathfrak{G}$ $\mathfrak{F}$ =
(
$\forall \sigma\ a.$
$\sigma : \mathfrak{F} >_{CF.colim} a : \mathfrak{F}(\!|HomDom|\!) \mapsto\mapsto_{C\alpha} \mathfrak{F}(\!|HomCod|\!) \longrightarrow$
$\mathfrak{G} \circ_{CF-NTCF} \sigma : \mathfrak{G} \circ_{CF} \mathfrak{F} >_{CF.colim} \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) : \mathfrak{F}(\!|HomDom|\!) \mapsto\mapsto_{C\alpha} \mathfrak{G}(\!|HomCod|\!)$
)

Rules.

context
fixes α $\mathfrak{J}$ $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ $\mathfrak{F}$
assumes $\mathfrak{F}$: $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G}$: $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
begin

interpretation $\mathfrak{F}$: *is-functor* α $\mathfrak{J}$ $\mathfrak{A}$ $\mathfrak{F}$ ⟨*proof*⟩
interpretation $\mathfrak{G}$: *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ ⟨*proof*⟩

mk-ide rf *cf-preserves-limits-def*[
where α=α **and** $\mathfrak{F}$=$\mathfrak{F}$ **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded cat-cs-simps*
]
|*intro cf-preserves-limitsI*|
|*dest cf-preserves-limitsD'*|
|*elim cf-preserves-limitsE'*|

mk-ide rf *cf-preserves-colimits-def*[
where α=α **and** $\mathfrak{F}$=$\mathfrak{F}$ **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded cat-cs-simps*
]
|*intro cf-preserves-colimitsI*|
|*dest cf-preserves-colimitsD'*|
|*elim cf-preserves-colimitsE'*|

end

lemmas *cf-preserves-limitsD*[*dest!*] = *cf-preserves-limitsD'*[*rotated 2*]
and *cf-preserves-limitsE*[*elim!*] = *cf-preserves-limitsE'*[*rotated 2*]

lemmas *cf-preserves-colimitsD*[*dest!*] = *cf-preserves-colimitsD'*[*rotated 2*]
and *cf-preserves-colimitsE*[*elim!*] = *cf-preserves-colimitsE'*[*rotated 2*]

Duality.

lemma *cf-preserves-colimits-op*[*cat-op-simps*]:
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows
cf-preserves-colimits α (*op-cf* $\mathfrak{G}$) (*op-cf* $\mathfrak{F}$) $\longleftrightarrow$

cf-preserves-limits α $\mathfrak{G}$ $\mathfrak{F}$
⟨*proof*⟩

lemma *cf-preserves-limits-op*[*cat-op-simps*]:
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows
cf-preserves-limits α (*op-cf* $\mathfrak{G}$) (*op-cf* $\mathfrak{F}$) $\longleftrightarrow$
cf-preserves-colimits α $\mathfrak{G}$ $\mathfrak{F}$
⟨*proof*⟩

3.6.2 Further properties

lemma *cf-preserves-limits-if-cf-creates-limits*:
— See Theorem 2 in Chapter V-4 in [9].
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\psi : b <_{CF.lim} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *cf-creates-limits* α $\mathfrak{G}$ $\mathfrak{F}$
shows *cf-preserves-limits* α $\mathfrak{G}$ $\mathfrak{F}$
⟨*proof*⟩

3.7 Continuous and cocontinuous functor

3.7.1 Definition and elementary properties

definition *is-cf-continuous* :: $V \Rightarrow V \Rightarrow$ *bool*
where *is-cf-continuous* α $\mathfrak{G}$ $\longleftrightarrow$
($\forall \mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{G}(\!|HomDom|\!) \longrightarrow$ *cf-preserves-limits* α $\mathfrak{G}$ $\mathfrak{F}$)

definition *is-cf-cocontinuous* :: $V \Rightarrow V \Rightarrow$ *bool*
where *is-cf-cocontinuous* α $\mathfrak{G}$ $\longleftrightarrow$
($\forall \mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{G}(\!|HomDom|\!) \longrightarrow$ *cf-preserves-colimits* α $\mathfrak{G}$ $\mathfrak{F}$)

Rules.

context
fixes α $\mathfrak{J}$ $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ $\mathfrak{F}$
assumes $\mathfrak{G}$: $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
begin

interpretation $\mathfrak{G}$: *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ ⟨*proof*⟩

mk-ide rf *is-cf-continuous-def*[**where** α=α **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded cat-cs-simps*]
|*intro is-cf-continuousI*|
|*dest is-cf-continuousD'*|
|*elim is-cf-continuousE'*|

mk-ide rf *is-cf-cocontinuous-def*[**where** α=α **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded cat-cs-simps*]
|*intro is-cf-cocontinuousI*|
|*dest is-cf-cocontinuousD'*|
|*elim is-cf-cocontinuousE'*|

end

lemmas *is-cf-continuousD*[*dest*!] = *is-cf-continuousD'*[*rotated*]
and *is-cf-continuousE*[*elim*!] = *is-cf-continuousE'*[*rotated*]

lemmas *is-cf-cocontinuousD*[*dest*!] = *is-cf-cocontinuousD'*[*rotated*]
and *is-cf-cocontinuousE*[*elim*!] = *is-cf-cocontinuousE'*[*rotated*]

Duality.

lemma *is-cf-continuous-op*[*cat-op-simps*]:
  assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
  shows *is-cf-continuous* α (*op-cf* $\mathfrak{G}$) $\longleftrightarrow$ *is-cf-cocontinuous* α $\mathfrak{G}$
⟨*proof*⟩

lemma *is-cf-cocontinuous-op*[*cat-op-simps*]:
  assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
  shows *is-cf-cocontinuous* α (*op-cf* $\mathfrak{G}$) $\longleftrightarrow$ *is-cf-continuous* α $\mathfrak{G}$
⟨*proof*⟩

3.7.2 Category isomorphisms are continuous and cocontinuous

lemma (**in** *is-iso-functor*) *iso-cf-is-cf-continuous*: *is-cf-continuous* α $\mathfrak{F}$
⟨*proof*⟩

lemma (**in** *is-iso-functor*) *iso-cf-is-cf-cocontinuous*: *is-cf-cocontinuous* α $\mathfrak{F}$
  ⟨*proof*⟩

3.8 Tiny-continuous and tiny-cocontinuous functor

3.8.1 Definition and elementary properties

definition *is-tm-cf-continuous* :: $V \Rightarrow V \Rightarrow$ *bool*
  where *is-tm-cf-continuous* α $\mathfrak{G}$ =
    ($\forall \mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{G}(\!|HomDom|\!) \longrightarrow$ *cf-preserves-limits* α $\mathfrak{G}$ $\mathfrak{F}$)

Rules.

context
  fixes α $\mathfrak{J}$ $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ $\mathfrak{F}$
  assumes $\mathfrak{G}$: $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
begin

interpretation $\mathfrak{G}$: *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ ⟨*proof*⟩

mk-ide rf *is-tm-cf-continuous-def*[**where** α=α **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded cat-cs-simps*]
  |*intro is-tm-cf-continuousI*|
  |*dest is-tm-cf-continuousD′*|
  |*elim is-tm-cf-continuousE′*|

end

lemmas *is-tm-cf-continuousD*[*dest*!] = *is-tm-cf-continuousD′*[*rotated*]
  and *is-tm-cf-continuousE*[*elim*!] = *is-tm-cf-continuousE′*[*rotated*]

Elementary properties.

lemma (**in** *is-functor*) *cf-continuous-is-tm-cf-continuous*:
  assumes *is-cf-continuous* α $\mathfrak{F}$
  shows *is-tm-cf-continuous* α $\mathfrak{F}$
⟨*proof*⟩

4 Initial and terminal objects as limits and colimits

4.1 Initial and terminal objects as limits/colimits of an empty diagram

4.1.1 Definition and elementary properties

See [1][2], [1][3] and Chapter X-1 in [9].

locale *is-cat-obj-empty-terminal* = *is-cat-limit* α *cat-0* $\mathfrak{C}$ ‹*cf-0* $\mathfrak{C}$› z $\mathfrak{Z}$
  for α $\mathfrak{C}$ z $\mathfrak{Z}$

syntax *-is-cat-obj-empty-terminal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $<_{CF.1}$ 0_{CF} :/ 0_C $\mapsto\mapsto_{C1}$ -)› [*51*, *51*] *51*)
syntax-consts *-is-cat-obj-empty-terminal* ⇌ *is-cat-obj-empty-terminal*
translations $\mathfrak{Z}$: z $<_{CF.1}$ 0_{CF} : 0_C $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-obj-empty-terminal α $\mathfrak{C}$ z $\mathfrak{Z}$

locale *is-cat-obj-empty-initial* = *is-cat-colimit* α *cat-0* $\mathfrak{C}$ ‹*cf-0* $\mathfrak{C}$› z $\mathfrak{Z}$
  for α $\mathfrak{C}$ z $\mathfrak{Z}$

syntax *-is-cat-obj-empty-initial* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ 0_{CF} $>_{CF.0}$ - :/ 0_C $\mapsto\mapsto_{C1}$ -)› [*51*, *51*] *51*)
syntax-consts *-is-cat-obj-empty-initial* ⇌ *is-cat-obj-empty-initial*
translations $\mathfrak{Z}$: 0_{CF} $>_{CF.0}$ z : 0_C $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-obj-empty-initial α $\mathfrak{C}$ z $\mathfrak{Z}$

Rules.

lemma (**in** *is-cat-obj-empty-terminal*)
  is-cat-obj-empty-terminal-axioms'[*cat-lim-cs-intros*]:
  assumes α' = α **and** z' = z **and** $\mathfrak{C}'$ = $\mathfrak{C}$
  shows $\mathfrak{Z}$: z' $<_{CF.1}$ 0_{CF} : 0_C $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-obj-empty-terminal-def*
  |*intro is-cat-obj-empty-terminalI*|
  |*dest is-cat-obj-empty-terminalD*[*dest*]|
  |*elim is-cat-obj-empty-terminalE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-empty-terminalD*

lemma (**in** *is-cat-obj-empty-initial*)
  is-cat-obj-empty-initial-axioms'[*cat-lim-cs-intros*]:
  assumes α' = α **and** z' = z **and** $\mathfrak{C}'$ = $\mathfrak{C}$
  shows $\mathfrak{Z}$: 0_{CF} $>_{CF.0}$ z' : 0_C $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-obj-empty-initial-def*
  |*intro is-cat-obj-empty-initialI*|
  |*dest is-cat-obj-empty-initialD*[*dest*]|
  |*elim is-cat-obj-empty-initialE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-empty-initialD*

Duality.

lemma (**in** *is-cat-obj-empty-terminal*) *is-cat-obj-empty-initial-op*:
  op-ntcf $\mathfrak{Z}$: 0_{CF} $>_{CF.0}$ z : 0_C $\mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$

[2] https://ncatlab.org/nlab/show/initial+object
[3] https://ncatlab.org/nlab/show/terminal+object

⟨*proof*⟩

lemma (**in** *is-cat-obj-empty-terminal*) *is-cat-obj-empty-initial-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
shows $op\text{-}ntcf\ \mathfrak{Z} : 0_{CF} >_{CF.0} z : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-empty-terminal.is-cat-obj-empty-initial-op'*

lemma (**in** *is-cat-obj-empty-initial*) *is-cat-obj-empty-terminal-op*:
$op\text{-}ntcf\ \mathfrak{Z} : z <_{CF.1} 0_{CF} : 0_C \mapsto\mapsto_{C\alpha} op\text{-}cat\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-empty-initial*) *is-cat-obj-empty-terminal-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
shows $op\text{-}ntcf\ \mathfrak{Z} : z <_{CF.1} 0_{CF} : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-empty-initial.is-cat-obj-empty-terminal-op'*

Elementary properties.

lemma (**in** *is-cat-obj-empty-terminal*) *cat-oet-ntcf-0*: $\mathfrak{Z} = ntcf\text{-}0\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-empty-initial*) *cat-oei-ntcf-0*: $\mathfrak{Z} = ntcf\text{-}0\ \mathfrak{C}$
⟨*proof*⟩

4.1.2 Initial and terminal objects as limits/colimits of an empty diagram are initial and terminal objects

lemma (**in** *category*) *cat-obj-terminal-is-cat-obj-empty-terminal*:
assumes *obj-terminal* $\mathfrak{C}$ *z*
shows $ntcf\text{-}0\ \mathfrak{C} : z <_{CF.1} 0_{CF} : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-initial-is-cat-obj-empty-initial*:
assumes *obj-initial* $\mathfrak{C}$ *z*
shows $ntcf\text{-}0\ \mathfrak{C} : 0_{CF} >_{CF.0} z : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-empty-terminal*) *cat-oet-obj-terminal*: *obj-terminal* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *is-cat-obj-empty-initial*) *cat-oei-obj-initial*: *obj-initial* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *category*) *cat-is-cat-obj-empty-terminal-obj-terminal-iff*:
$(ntcf\text{-}0\ \mathfrak{C} : z <_{CF.1} 0_{CF} : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}) \longleftrightarrow obj\text{-}terminal\ \mathfrak{C}\ z$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-cat-obj-empty-initial-obj-initial-iff*:
$(ntcf\text{-}0\ \mathfrak{C} : 0_{CF} >_{CF.0} z : 0_C \mapsto\mapsto_{C\alpha} \mathfrak{C}) \longleftrightarrow obj\text{-}initial\ \mathfrak{C}\ z$
⟨*proof*⟩

4.2 Initial cone and terminal cocone

4.2.1 Definitions and elementary properties

definition *ntcf-initial* :: $V \Rightarrow V \Rightarrow V$

where *ntcf-initial* $\mathfrak{C}$ *z* =
  [
    (λb∈∘$\mathfrak{C}$(|*Obj*|). *THE f. f* : *z* $\mapsto_{\mathfrak{C}}$ *b*),
    cf-const $\mathfrak{C}$ $\mathfrak{C}$ *z*,
    cf-id $\mathfrak{C}$,
    $\mathfrak{C}$,
    $\mathfrak{C}$
  ]∘

definition *ntcf-terminal* :: *V* ⇒ *V* ⇒ *V*
  where *ntcf-terminal* $\mathfrak{C}$ *z* =
  [
    (λb∈∘$\mathfrak{C}$(|*Obj*|). *THE f. f* : *b* $\mapsto_{\mathfrak{C}}$ *z*),
    cf-id $\mathfrak{C}$,
    cf-const $\mathfrak{C}$ $\mathfrak{C}$ *z*,
    $\mathfrak{C}$,
    $\mathfrak{C}$
  ]∘

Components.

lemma *ntcf-initial-components*:
  shows *ntcf-initial* $\mathfrak{C}$ *z*(|*NTMap*|) = (λc∈∘$\mathfrak{C}$(|*Obj*|). *THE f. f* : *z* $\mapsto_{\mathfrak{C}}$ *c*)
    and *ntcf-initial* $\mathfrak{C}$ *z*(|*NTDom*|) = *cf-const* $\mathfrak{C}$ $\mathfrak{C}$ *z*
    and *ntcf-initial* $\mathfrak{C}$ *z*(|*NTCod*|) = *cf-id* $\mathfrak{C}$
    and *ntcf-initial* $\mathfrak{C}$ *z*(|*NTDGDom*|) = $\mathfrak{C}$
    and *ntcf-initial* $\mathfrak{C}$ *z*(|*NTDGCod*|) = $\mathfrak{C}$
  ⟨*proof*⟩

lemmas [*cat-lim-cs-simps*] = *ntcf-initial-components*(*2–5*)

lemma *ntcf-terminal-components*:
  shows *ntcf-terminal* $\mathfrak{C}$ *z*(|*NTMap*|) = (λc∈∘$\mathfrak{C}$(|*Obj*|). *THE f. f* : *c* $\mapsto_{\mathfrak{C}}$ *z*)
    and *ntcf-terminal* $\mathfrak{C}$ *z*(|*NTDom*|) = *cf-id* $\mathfrak{C}$
    and *ntcf-terminal* $\mathfrak{C}$ *z*(|*NTCod*|) = *cf-const* $\mathfrak{C}$ $\mathfrak{C}$ *z*
    and *ntcf-terminal* $\mathfrak{C}$ *z*(|*NTDGDom*|) = $\mathfrak{C}$
    and *ntcf-terminal* $\mathfrak{C}$ *z*(|*NTDGCod*|) = $\mathfrak{C}$
  ⟨*proof*⟩

lemmas [*cat-lim-cs-simps*] = *ntcf-terminal-components*(*2–5*)

Duality.

lemma *ntcf-initial-op*[*cat-op-simps*]:
  op-ntcf (*ntcf-initial* $\mathfrak{C}$ *z*) = *ntcf-terminal* (*op-cat* $\mathfrak{C}$) *z*
  ⟨*proof*⟩

lemma *ntcf-cone-terminal-op*[*cat-op-simps*]:
  op-ntcf (*ntcf-terminal* $\mathfrak{C}$ *z*) = *ntcf-initial* (*op-cat* $\mathfrak{C}$) *z*
  ⟨*proof*⟩

4.2.2 Natural transformation map

mk-VLambda *ntcf-initial-components*(*1*)
  |*vsv ntcf-initial-vsv*[*cat-lim-cs-intros*]|
  |*vdomain ntcf-initial-vdomain*[*cat-lim-cs-simps*]|
  |*app ntcf-initial-app*|

mk-VLambda *ntcf-terminal-components*(*1*)
  |*vsv ntcf-terminal-vsv*[*cat-lim-cs-intros*]|

|*vdomain ntcf-terminal-vdomain*[*cat-lim-cs-simps*]|
|*app ntcf-terminal-app*|

lemma (**in** *category*)
assumes *obj-initial* $\mathfrak{C}$ *z* **and** $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-initial-NTMap-app-is-arr*:
ntcf-initial $\mathfrak{C}$ $z(\!|NTMap|\!)(\!|c|\!) : z \mapsto_{\mathfrak{C}} c$
and *ntcf-initial-NTMap-app-unique*:
$\bigwedge f'. f' : z \mapsto_{\mathfrak{C}} c \Longrightarrow f' = ntcf\text{-}initial\ \mathfrak{C}\ z(\!|NTMap|\!)(\!|c|\!)$
⟨*proof*⟩

lemma (**in** *category*) *ntcf-initial-NTMap-app-is-arr'*[*cat-lim-cs-intros*]:
assumes *obj-initial* $\mathfrak{C}$ *z*
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{C}' = \mathfrak{C}$
and $z' = z$
and $c' = c$
shows *ntcf-initial* $\mathfrak{C}$ $z(\!|NTMap|\!)(\!|c|\!) : z' \mapsto_{\mathfrak{C}'} c'$
⟨*proof*⟩

lemma (**in** *category*)
assumes *obj-terminal* $\mathfrak{C}$ *z* **and** $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-terminal-NTMap-app-is-arr*:
ntcf-terminal $\mathfrak{C}$ $z(\!|NTMap|\!)(\!|c|\!) : c \mapsto_{\mathfrak{C}} z$
and *ntcf-terminal-NTMap-app-unique*:
$\bigwedge f'. f' : c \mapsto_{\mathfrak{C}} z \Longrightarrow f' = ntcf\text{-}terminal\ \mathfrak{C}\ z(\!|NTMap|\!)(\!|c|\!)$
⟨*proof*⟩

lemma (**in** *category*) *ntcf-terminal-NTMap-app-is-arr'*[*cat-lim-cs-intros*]:
assumes *obj-terminal* $\mathfrak{C}$ *z*
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{C}' = \mathfrak{C}$
and $z' = z$
and $c' = c$
shows *ntcf-terminal* $\mathfrak{C}$ $z(\!|NTMap|\!)(\!|c|\!) : c' \mapsto_{\mathfrak{C}'} z'$
⟨*proof*⟩

4.3 Initial and terminal objects as limits/colimits of the identity functor

4.3.1 Definition and elementary properties

See [1][4], [1][5] and Chapter X-1 in [9].

locale *is-cat-obj-id-initial* = *is-cat-limit* α $\mathfrak{C}$ $\mathfrak{C}$ ‹*cf-id* $\mathfrak{C}$› *z* $\mathfrak{Z}$ **for** α $\mathfrak{C}$ *z* $\mathfrak{Z}$

syntax *-is-cat-obj-id-initial* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - $<_{CF.0}$ id_C :/ $\mapsto\mapsto_{C}$1 -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-cat-obj-id-initial* ⇌ *is-cat-obj-id-initial*
translations $\mathfrak{Z}$: *z* $<_{CF.0}$ id_C : $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ ⇌
CONST is-cat-obj-id-initial α $\mathfrak{C}$ *z* $\mathfrak{Z}$

locale *is-cat-obj-id-terminal* = *is-cat-colimit* α $\mathfrak{C}$ $\mathfrak{C}$ ‹*cf-id* $\mathfrak{C}$› *z* $\mathfrak{Z}$ **for** α $\mathfrak{C}$ *z* $\mathfrak{Z}$

syntax *-is-cat-obj-id-terminal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ id_C $>_{CF.1}$ - :/ $\mapsto\mapsto_{C}$1 -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-cat-obj-id-terminal* ⇌ *is-cat-obj-id-terminal*

[4] https://ncatlab.org/nlab/show/initial+object
[5] https://ncatlab.org/nlab/show/terminal+object

translations $\mathfrak{Z} : id_C >_{CF.1} z : \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-obj-id-terminal α $\mathfrak{C}$ z $\mathfrak{Z}$

Rules.

lemma (**in** *is-cat-obj-id-initial*)
is-cat-obj-id-initial-axioms'[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $z' = z$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{Z} : z' <_{CF.0} id_C : \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-obj-id-initial-def*
|*intro is-cat-obj-id-initialI*|
|*dest is-cat-obj-id-initialD*[*dest*]|
|*elim is-cat-obj-id-initialE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-id-initialD*

lemma (**in** *is-cat-obj-id-terminal*)
is-cat-obj-id-terminal-axioms'[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $z' = z$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{Z} : id_C >_{CF.1} z' : \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-obj-id-terminal-def*
|*intro is-cat-obj-id-terminalI*|
|*dest is-cat-obj-id-terminalD*[*dest*]|
|*elim is-cat-obj-id-terminalE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-id-terminalD*

Duality.

lemma (**in** *is-cat-obj-id-initial*) *is-cat-obj-id-terminal-op*:
op-ntcf $\mathfrak{Z} : id_C >_{CF.1} z : \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-id-initial*) *is-cat-obj-id-terminal-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\mathfrak{Z} : id_C >_{CF.1} z : \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-id-initial.is-cat-obj-id-terminal-op'*

lemma (**in** *is-cat-obj-id-terminal*) *is-cat-obj-id-initial-op*:
op-ntcf $\mathfrak{Z} : z <_{CF.0} id_C : \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-id-terminal*) *is-cat-obj-id-initial-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\mathfrak{Z} : z <_{CF.0} id_C : \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-id-terminal.is-cat-obj-id-initial-op'*

4.3.2 Initial and terminal objects as limits/colimits are initial and terminal objects

lemma (**in** *category*) *cat-obj-initial-is-cat-obj-id-initial*:
assumes *obj-initial* $\mathfrak{C}$ z
shows *ntcf-initial* $\mathfrak{C}$ $z : z <_{CF.0} id_C : \mapsto\mapsto_{C\alpha} \mathfrak{C}$

⟨*proof*⟩

lemma (**in** *category*) *cat-obj-terminal-is-cat-obj-id-terminal*:
assumes *obj-terminal* $\mathfrak{C}$ *z*
shows *ntcf-terminal* $\mathfrak{C}$ $z : id_C >_{CF.1} z : \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-cone-CId-obj-initial*:
assumes $\mathfrak{Z} : z <_{CF.cone}$ *cf-id* $\mathfrak{C} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{Z}(|NTMap|)(|z|) = \mathfrak{C}(|CId|)(|z|)$
shows *obj-initial* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma *cat-cocone-CId-obj-terminal*:
assumes $\mathfrak{Z}$: *cf-id* $\mathfrak{C} >_{CF.cocone} z : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{Z}(|NTMap|)(|z|) = \mathfrak{C}(|CId|)(|z|)$
shows *obj-terminal* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *is-cat-obj-id-initial*) *cat-oii-obj-initial*: *obj-initial* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *is-cat-obj-id-terminal*) *cat-oit-obj-terminal*: *obj-terminal* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *category*) *cat-is-cat-obj-id-initial-obj-initial-iff*:
(*ntcf-initial* $\mathfrak{C}$ $z : z <_{CF.0} id_C : \mapsto\mapsto_{C\alpha} \mathfrak{C}$) ⟷ *obj-initial* $\mathfrak{C}$ *z*
⟨*proof*⟩

lemma (**in** *category*) *cat-is-cat-obj-id-terminal-obj-terminal-iff*:
(*ntcf-terminal* $\mathfrak{C}$ $z : id_C >_{CF.1} z : \mapsto\mapsto_{C\alpha} \mathfrak{C}$) ⟷ *obj-terminal* $\mathfrak{C}$ *z*
⟨*proof*⟩

5 Products and coproducts as limits and colimits

5.1 Product and coproduct

5.1.1 Definition and elementary properties

The definition of the product object is a specialization of the definition presented in Chapter III-4 in [9]. In the definition presented below, the discrete category that is used in the definition presented in [9] is parameterized by an index set and the functor from the discrete category is parameterized by a function from the index set to the set of the objects of the category.

locale *is-cat-obj-prod* =
  is-cat-limit α ‹:$_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› P π + *cf-discrete* α I A $\mathfrak{C}$
  for α I A $\mathfrak{C}$ P π

syntax *-is-cat-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $<_{CF.\prod}$ - :/ - $\mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-obj-prod* ⇌ *is-cat-obj-prod*
translations π : P $<_{CF.\prod}$ A : I $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-obj-prod α I A $\mathfrak{C}$ P π

locale *is-cat-obj-coprod* =
  is-cat-colimit α ‹:$_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› U π + *cf-discrete* α I A $\mathfrak{C}$
  for α I A $\mathfrak{C}$ U π

syntax *-is-cat-obj-coprod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $>_{CF.\coprod}$ - :/ - $\mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-obj-coprod* ⇌ *is-cat-obj-coprod*
translations π : A $>_{CF.\coprod}$ U : I $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-obj-coprod α I A $\mathfrak{C}$ U π

Rules.

lemma (**in** *is-cat-obj-prod*) *is-cat-obj-prod-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows π : P' $<_{CF.\prod}$ A' : I' $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-obj-prod-def*
  |*intro is-cat-obj-prodI*|
  |*dest is-cat-obj-prodD*[*dest*]|
  |*elim is-cat-obj-prodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-prodD*

lemma (**in** *is-cat-obj-coprod*) *is-cat-obj-coprod-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows π : A' $>_{CF.\coprod}$ U' : I' $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-obj-coprod-def*
  |*intro is-cat-obj-coprodI*|
  |*dest is-cat-obj-coprodD*[*dest*]|
  |*elim is-cat-obj-coprodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-coprodD*

Duality.

lemma (**in** *is-cat-obj-prod*) *is-cat-obj-coprod-op*:

$op\text{-}ntcf\ \pi : A >_{CF.\coprod}\ P : I \mapsto\mapsto_{C\alpha}\ op\text{-}cat\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod*) *is-cat-obj-coprod-op′*[*cat-op-intros*]:
assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
shows $op\text{-}ntcf\ \pi : A >_{CF.\coprod}\ P : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-prod.is-cat-obj-coprod-op′*

lemma (**in** *is-cat-obj-coprod*) *is-cat-obj-prod-op*:
$op\text{-}ntcf\ \pi : U <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ op\text{-}cat\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod*) *is-cat-obj-prod-op′*[*cat-op-intros*]:
assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
shows $op\text{-}ntcf\ \pi : U <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-coprod.is-cat-obj-prod-op′*

5.1.2 Universal property

lemma (**in** *is-cat-obj-prod*) *cat-obj-prod-unique-cone′*:
assumes $\pi' : P' <_{CF.cone} :\rightarrow: I\ A\ \mathfrak{C} : :_C\ I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge (\forall j\in_\circ I.\ \pi'(|NTMap|)(|j|) = \pi(|NTMap|)(|j|) \circ_{A\mathfrak{C}} f')$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod*) *cat-obj-prod-unique*:
assumes $\pi' : P' <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge \pi' = \pi \bullet_{NTCF}\ ntcf\text{-}const\ (:_C\ I)\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod*) *cat-obj-prod-unique′*:
assumes $\pi' : P' <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge (\forall i\in_\circ I.\ \pi'(|NTMap|)(|i|) = \pi(|NTMap|)(|i|) \circ_{A\mathfrak{C}} f')$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod*) *cat-obj-coprod-unique-cocone′*:
assumes $\pi' : :\rightarrow: I\ A\ \mathfrak{C} >_{CF.cocone}\ U' : :_C\ I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : U \mapsto_{\mathfrak{C}} U' \wedge (\forall j\in_\circ I.\ \pi'(|NTMap|)(|j|) = f' \circ_{A\mathfrak{C}} \pi(|NTMap|)(|j|))$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod*) *cat-obj-coprod-unique*:
assumes $\pi' : A >_{CF.\coprod}\ U' : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : U \mapsto_{\mathfrak{C}} U' \wedge \pi' = ntcf\text{-}const\ (:_C\ I)\ \mathfrak{C}\ f' \bullet_{NTCF}\ \pi$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod*) *cat-obj-coprod-unique′*:
assumes $\pi' : A >_{CF.\coprod}\ U' : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
shows $\exists!f'.\ f' : U \mapsto_{\mathfrak{C}} U' \wedge (\forall j\in_\circ I.\ \pi'(|NTMap|)(|j|) = f' \circ_{A\mathfrak{C}} \pi(|NTMap|)(|j|))$
⟨*proof*⟩

lemma *cat-obj-prod-ex-is-iso-arr*:
assumes $\pi : P <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$ **and** $\pi' : P' <_{CF.\prod}\ A : I \mapsto\mapsto_{C\alpha}\ \mathfrak{C}$
obtains f **where** $f : P' \mapsto_{iso\mathfrak{C}} P$ **and** $\pi' = \pi \bullet_{NTCF}\ ntcf\text{-}const\ (:_C\ I)\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-obj-prod-ex-is-iso-arr'*:
assumes $\pi : P <_{CF.\prod} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : P' <_{CF.\prod} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : P' \mapsto_{iso\mathfrak{C}} P$
and $\bigwedge j.\ j \in_\circ I \Longrightarrow \pi'(\!|NTMap|\!)(\!|j|\!) = \pi(\!|NTMap|\!)(\!|j|\!) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-obj-coprod-ex-is-iso-arr*:
assumes $\pi : A >_{CF.\coprod} U : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : A >_{CF.\coprod} U' : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : U \mapsto_{iso\mathfrak{C}} U'$ **and** $\pi' = \textit{ntcf-const}\ (:_C\ I)\ \mathfrak{C}\ f \bullet_{NTCF} \pi$
⟨*proof*⟩

lemma *cat-obj-coprod-ex-is-iso-arr'*:
assumes $\pi : A >_{CF.\coprod} U : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : A >_{CF.\coprod} U' : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : U \mapsto_{iso\mathfrak{C}} U'$
and $\bigwedge j.\ j \in_\circ I \Longrightarrow \pi'(\!|NTMap|\!)(\!|j|\!) = f \circ_{A\mathfrak{C}} \pi(\!|NTMap|\!)(\!|j|\!)$
⟨*proof*⟩

5.2 Small product and small coproduct

5.2.1 Definition and elementary properties

locale *is-tm-cat-obj-prod* =
is-cat-limit α ‹$:_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› P π + *tm-cf-discrete* α I A $\mathfrak{C}$
for α I A $\mathfrak{C}$ P π

syntax *-is-tm-cat-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $<_{CF.tm.\prod}$ - :/ - $\mapsto\mapsto_{C.tm}$1 -)› [*51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-tm-cat-obj-prod* ⇌ *is-tm-cat-obj-prod*
translations $\pi : P <_{CF.tm.\prod} A : I \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ ⇌
CONST is-tm-cat-obj-prod α I A $\mathfrak{C}$ P π

locale *is-tm-cat-obj-coprod* =
is-cat-colimit α ‹$:_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› U π + *tm-cf-discrete* α I A $\mathfrak{C}$
for α I A $\mathfrak{C}$ U π

syntax *-is-tm-cat-obj-coprod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $>_{CF.tm.\coprod}$ - :/ - $\mapsto\mapsto_{C.tm}$1 -)› [*51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-tm-cat-obj-coprod* ⇌ *is-tm-cat-obj-coprod*
translations $\pi : A >_{CF.tm.\coprod} U : I \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ ⇌
CONST is-tm-cat-obj-coprod α I A $\mathfrak{C}$ U π

Rules.

lemma (**in** *is-tm-cat-obj-prod*) *is-tm-cat-obj-prod-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\pi : P' <_{CF.tm.\prod} A' : I' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-tm-cat-obj-prod-def*
|*intro is-tm-cat-obj-prodI*|
|*dest is-tm-cat-obj-prodD*[*dest*]|
|*elim is-tm-cat-obj-prodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-tm-cat-obj-prodD*

lemma (**in** *is-tm-cat-obj-coprod*)
is-tm-cat-obj-coprod-axioms'[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\pi : A' >_{CF.tm.\coprod} U' : I' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{C}'$

⟨*proof*⟩

mk-ide rf *is-tm-cat-obj-coprod-def*
|*intro is-tm-cat-obj-coprodI*|
|*dest is-tm-cat-obj-coprodD*[*dest*]|
|*elim is-tm-cat-obj-coprodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-tm-cat-obj-coprodD*

Elementary properties.

sublocale *is-tm-cat-obj-prod* ⊆ *is-cat-obj-prod*
⟨*proof*⟩

lemmas (**in** *is-tm-cat-obj-prod*) *tm-cat-obj-prod-is-cat-obj-prod* =
is-cat-obj-prod-axioms

sublocale *is-tm-cat-obj-coprod* ⊆ *is-cat-obj-coprod*
⟨*proof*⟩

lemmas (**in** *is-tm-cat-obj-coprod*) *tm-cat-obj-coprod-is-cat-obj-coprod* =
is-cat-obj-coprod-axioms

sublocale *is-tm-cat-obj-prod* ⊆ *is-tm-cat-limit* α ‹:$_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› P π
⟨*proof*⟩

lemmas (**in** *is-tm-cat-obj-prod*) *tm-cat-obj-prod-is-tm-cat-limit* =
is-tm-cat-limit-axioms

sublocale *is-tm-cat-obj-coprod* ⊆ *is-tm-cat-colimit* α ‹:$_C$ I› $\mathfrak{C}$ ‹:→: I A $\mathfrak{C}$› U π
⟨*proof*⟩

lemmas (**in** *is-tm-cat-obj-coprod*) *tm-cat-obj-coprod-is-tm-cat-colimit* =
is-tm-cat-colimit-axioms

Duality.

lemma (**in** *is-tm-cat-obj-prod*) *is-tm-cat-obj-coprod-op*:
op-ntcf π : $A >_{CF.tm.\coprod} P : I \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-obj-prod*) *is-tm-cat-obj-coprod-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* π : $A >_{CF.tm.\coprod} P : I \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-tm-cat-obj-prod.is-tm-cat-obj-coprod-op'*

lemma (**in** *is-tm-cat-obj-coprod*) *is-tm-cat-obj-coprod-op*:
op-ntcf π : $U <_{CF.tm.\prod} A : I \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-obj-coprod*) *is-tm-cat-obj-prod-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* π : $U <_{CF.tm.\prod} A : I \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-tm-cat-obj-coprod.is-tm-cat-obj-prod-op'*

5.3 Finite product and finite coproduct

locale *is-cat-finite-obj-prod* = *is-cat-obj-prod* α I A $\mathfrak{C}$ P π
  for α I A $\mathfrak{C}$ P π +
  assumes *cat-fin-obj-prod-index-in-ω*: $I \in_\circ \omega$

syntax *-is-cat-finite-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-} <_{CF.\prod.fin} \text{-} :/ \text{-} \mapsto\mapsto_{C}1 \text{-})\rangle$ $[51, 51, 51, 51, 51]$ $51)$
syntax-consts *-is-cat-finite-obj-prod* ⇌ *is-cat-finite-obj-prod*
translations $\pi : P <_{CF.\prod.fin} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
  CONST is-cat-finite-obj-prod α I A $\mathfrak{C}$ P π

locale *is-cat-finite-obj-coprod* = *is-cat-obj-coprod* α I A $\mathfrak{C}$ U π
  for α I A $\mathfrak{C}$ U π +
  assumes *cat-fin-obj-coprod-index-in-ω*: $I \in_\circ \omega$

syntax *-is-cat-finite-obj-coprod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-} >_{CF.\coprod.fin} \text{-} :/ \text{-} \mapsto\mapsto_{C}1 \text{-})\rangle$ $[51, 51, 51, 51, 51]$ $51)$
syntax-consts *-is-cat-finite-obj-coprod* ⇌ *is-cat-finite-obj-coprod*
translations $\pi : A >_{CF.\coprod.fin} U : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
  CONST is-cat-finite-obj-coprod α I A $\mathfrak{C}$ U π

lemma (**in** *is-cat-finite-obj-prod*) *cat-fin-obj-prod-index-vfinite*: *vfinite I*
  ⟨*proof*⟩

sublocale *is-cat-finite-obj-prod* ⊆ *I*: *finite-category* α ‹$:_C$ I›
  ⟨*proof*⟩

lemma (**in** *is-cat-finite-obj-coprod*) *cat-fin-obj-coprod-index-vfinite*:
  vfinite I
  ⟨*proof*⟩

sublocale *is-cat-finite-obj-coprod* ⊆ *I*: *finite-category* α ‹$:_C$ I›
  ⟨*proof*⟩

Rules.

lemma (**in** *is-cat-finite-obj-prod*)
  is-cat-finite-obj-prod-axioms′[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $\pi : P' <_{CF.\prod.fin} A' : I' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf
  is-cat-finite-obj-prod-def[*unfolded is-cat-finite-obj-prod-axioms-def*]
  |*intro is-cat-finite-obj-prodI*|
  |*dest is-cat-finite-obj-prodD*[*dest*]|
  |*elim is-cat-finite-obj-prodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-finite-obj-prodD*

lemma (**in** *is-cat-finite-obj-coprod*)
  is-cat-finite-obj-coprod-axioms′[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $\pi : A' >_{CF.\coprod.fin} U' : I' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf
  is-cat-finite-obj-coprod-def[*unfolded is-cat-finite-obj-coprod-axioms-def*]

|*intro is-cat-finite-obj-coprodI*|
|*dest is-cat-finite-obj-coprodD*[*dest*]|
|*elim is-cat-finite-obj-coprodE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-finite-obj-coprodD*

Duality.

lemma (**in** *is-cat-finite-obj-prod*) *is-cat-finite-obj-coprod-op*:
op-ntcf π : $A >_{CF.\coprod.fin} P : I \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-finite-obj-prod*) *is-cat-finite-obj-coprod-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* π : $A >_{CF.\coprod.fin} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-finite-obj-prod.is-cat-finite-obj-coprod-op'*

lemma (**in** *is-cat-finite-obj-coprod*) *is-cat-finite-obj-prod-op*:
op-ntcf π : $U <_{CF.\prod.fin} A : I \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-finite-obj-coprod*) *is-cat-finite-obj-prod-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* π : $U <_{CF.\prod.fin} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-finite-obj-coprod.is-cat-finite-obj-prod-op'*

5.4 Product and coproduct of two objects

5.4.1 Definition and elementary properties

locale *is-cat-obj-prod-2* = *is-cat-obj-prod* α ‹$2_{\mathbb{N}}$› ‹*if2 a b*› $\mathfrak{C}$ P π
for α a b $\mathfrak{C}$ P π

syntax *-is-cat-obj-prod-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $<_{CF.\times}$ {-,-} :/ $2_C \mapsto\mapsto_{C^1}$ -)› [*51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-cat-obj-prod-2* ⇌ *is-cat-obj-prod-2*
translations π : $P <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
CONST is-cat-obj-prod-2 α a b $\mathfrak{C}$ P π

locale *is-cat-obj-coprod-2* = *is-cat-obj-coprod* α ‹$2_{\mathbb{N}}$› ‹*if2 a b*› $\mathfrak{C}$ P π
for α a b $\mathfrak{C}$ P π

syntax *-is-cat-obj-coprod-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ {-,-} $>_{CF.\uplus}$ - :/ $2_C \mapsto\mapsto_{C^1}$ -)› [*51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-cat-obj-coprod-2* ⇌ *is-cat-obj-coprod-2*
translations π : $\{a,b\} >_{CF.\uplus} U : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$ ⇌
CONST is-cat-obj-coprod-2 α a b $\mathfrak{C}$ U π

abbreviation *proj-fst* **where** *proj-fst* π ≡ *vpfst* (π(|*NTMap*|))
abbreviation *proj-snd* **where** *proj-snd* π ≡ *vpsnd* (π(|*NTMap*|))

Rules.

lemma (**in** *is-cat-obj-prod-2*) *is-cat-obj-prod-2-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $a' = a$ **and** $b' = b$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows π : $P' <_{CF.\times} \{a',b'\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$

⟨*proof*⟩

mk-ide rf *is-cat-obj-prod-2-def*
|*intro is-cat-obj-prod-2I*|
|*dest is-cat-obj-prod-2D*[*dest*]|
|*elim is-cat-obj-prod-2E*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-prod-2D*

lemma (**in** *is-cat-obj-coprod-2*) *is-cat-obj-coprod-2-axioms′*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $a' = a$ **and** $b' = b$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\pi : \{a',b'\} >_{CF.\uplus} P' : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-obj-coprod-2-def*
|*intro is-cat-obj-coprod-2I*|
|*dest is-cat-obj-coprod-2D*[*dest*]|
|*elim is-cat-obj-coprod-2E*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-coprod-2D*

Duality.

lemma (**in** *is-cat-obj-prod-2*) *is-cat-obj-coprod-2-op*:
op-ntcf $\pi : \{a,b\} >_{CF.\uplus} P : 2_C \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod-2*) *is-cat-obj-coprod-2-op′*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\pi : \{a,b\} >_{CF.\uplus} P : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-prod-2.is-cat-obj-coprod-2-op′*

lemma (**in** *is-cat-obj-coprod-2*) *is-cat-obj-prod-2-op*:
op-ntcf $\pi : P <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod-2*) *is-cat-obj-prod-2-op′*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\pi : P <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-obj-coprod-2.is-cat-obj-prod-2-op′*

Product/coproduct of two objects is a finite product/coproduct.

sublocale *is-cat-obj-prod-2* ⊆ *is-cat-finite-obj-prod* α ‹$2_{\mathbb{N}}$› ‹*if2 a b*› $\mathfrak{C}$ P π
⟨*proof*⟩

sublocale *is-cat-obj-coprod-2* ⊆ *is-cat-finite-obj-coprod* α ‹$2_{\mathbb{N}}$› ‹*if2 a b*› $\mathfrak{C}$ P π
⟨*proof*⟩

Elementary properties.

lemma (**in** *is-cat-obj-prod-2*) *cat-obj-prod-2-lr-in-Obj*:
shows *cat-obj-prod-2-left-in-Obj*[*cat-lim-cs-intros*]: $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
and *cat-obj-prod-2-right-in-Obj*[*cat-lim-cs-intros*]: $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-prod-2.cat-obj-prod-2-lr-in-Obj*

lemma (**in** *is-cat-obj-coprod-2*) *cat-obj-coprod-2-lr-in-Obj*:
shows *cat-obj-coprod-2-left-in-Obj*[*cat-lim-cs-intros*]: $a \in_\circ \mathfrak{C}(|Obj|)$
and *cat-obj-coprod-2-right-in-Obj*[*cat-lim-cs-intros*]: $b \in_\circ \mathfrak{C}(|Obj|)$
⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-obj-coprod-2.cat-obj-coprod-2-lr-in-Obj*

Utilities/help lemmas.

lemma *helper-I2-proj-fst-proj-snd-iff*:
$(\forall j \in_\circ 2_{\mathbb{N}}.\ \pi'(|NTMap|)(|j|) = \pi(|NTMap|)(|j|) \circ_{A\mathfrak{C}} f') \longleftrightarrow$
$(proj\text{-}fst\ \pi' = proj\text{-}fst\ \pi \circ_{A\mathfrak{C}} f' \wedge proj\text{-}snd\ \pi' = proj\text{-}snd\ \pi \circ_{A\mathfrak{C}} f')$
⟨*proof*⟩

lemma *helper-I2-proj-fst-proj-snd-iff'*:
$(\forall j \in_\circ 2_{\mathbb{N}}.\ \pi'(|NTMap|)(|j|) = f' \circ_{A\mathfrak{C}} \pi(|NTMap|)(|j|)) \longleftrightarrow$
$(proj\text{-}fst\ \pi' = f' \circ_{A\mathfrak{C}} proj\text{-}fst\ \pi \wedge proj\text{-}snd\ \pi' = f' \circ_{A\mathfrak{C}} proj\text{-}snd\ \pi)$
⟨*proof*⟩

5.4.2 Universal property

lemma (**in** *is-cat-obj-prod-2*) *cat-obj-prod-2-unique-cone'*:
assumes $\pi' : P' <_{CF.cone} :\rightarrow: (2_{\mathbb{N}})\ (if2\ a\ b)\ \mathfrak{C} : :_C (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists! f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge$
$proj\text{-}fst\ \pi' = proj\text{-}fst\ \pi \circ_{A\mathfrak{C}} f' \wedge$
$proj\text{-}snd\ \pi' = proj\text{-}snd\ \pi \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod-2*) *cat-obj-prod-2-unique*:
assumes $\pi' : P' <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists! f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge \pi' = \pi \cdot_{NTCF} ntcf\text{-}const\ (:_C (2_{\mathbb{N}}))\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (**in** *is-cat-obj-prod-2*) *cat-obj-prod-2-unique'*:
assumes $\pi' : P' <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists! f'.\ f' : P' \mapsto_{\mathfrak{C}} P \wedge$
$proj\text{-}fst\ \pi' = proj\text{-}fst\ \pi \circ_{A\mathfrak{C}} f' \wedge$
$proj\text{-}snd\ \pi' = proj\text{-}snd\ \pi \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod-2*) *cat-obj-coprod-2-unique-cocone'*:
assumes $\pi' : :\rightarrow: (2_{\mathbb{N}})\ (if2\ a\ b)\ \mathfrak{C} >_{CF.cocone} P' : :_C (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists! f'.\ f' : P \mapsto_{\mathfrak{C}} P' \wedge$
$proj\text{-}fst\ \pi' = f' \circ_{A\mathfrak{C}} proj\text{-}fst\ \pi \wedge$
$proj\text{-}snd\ \pi' = f' \circ_{A\mathfrak{C}} proj\text{-}snd\ \pi$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod-2*) *cat-obj-coprod-2-unique*:
assumes $\pi' : \{a,b\} >_{CF.\uplus} P' : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists! f'.\ f' : P \mapsto_{\mathfrak{C}} P' \wedge \pi' = ntcf\text{-}const\ (:_C (2_{\mathbb{N}}))\ \mathfrak{C}\ f' \cdot_{NTCF} \pi$
⟨*proof*⟩

lemma (**in** *is-cat-obj-coprod-2*) *cat-obj-coprod-2-unique'*:
assumes $\pi' : \{a,b\} >_{CF.\uplus} P' : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows
$\exists! f'. \ f' : P \mapsto_{\mathfrak{C}} P' \wedge$
proj-fst $\pi' = f' \circ_{A\mathfrak{C}}$ *proj-fst* $\pi \wedge$
proj-snd $\pi' = f' \circ_{A\mathfrak{C}}$ *proj-snd* π
⟨*proof*⟩

lemma *cat-obj-prod-2-ex-is-iso-arr*:
assumes $\pi : P <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\pi' : P' <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : P' \mapsto_{iso\mathfrak{C}} P$ **and** $\pi' = \pi \cdot_{NTCF}$ *ntcf-const* $(:_C (2_{\mathbb{N}}))$ $\mathfrak{C}$ f
⟨*proof*⟩

lemma *cat-obj-coprod-2-ex-is-iso-arr*:
assumes $\pi : \{a,b\} >_{CF.\uplus} U : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\pi' : \{a,b\} >_{CF.\uplus} U' : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : U \mapsto_{iso\mathfrak{C}} U'$ **and** $\pi' =$ *ntcf-const* $(:_C (2_{\mathbb{N}}))$ $\mathfrak{C}$ $f \cdot_{NTCF} \pi$
⟨*proof*⟩

5.5 Projection cone

5.5.1 Definition and elementary properties

definition *ntcf-obj-prod-base* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *ntcf-obj-prod-base* $\mathfrak{C}$ I F P f =
$[(\lambda j \in_{\circ} :_C I(|Obj|). \ f\ j)$, *cf-const* $(:_C I)$ $\mathfrak{C}$ P, $:\!\rightarrow\!: I\ F\ \mathfrak{C}$, $:_C I$, $\mathfrak{C}]_{\circ}$

definition *ntcf-obj-coprod-base* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P f =
$[(\lambda j \in_{\circ} :_C I(|Obj|). \ f\ j)$, $:\!\rightarrow\!: I\ F\ \mathfrak{C}$, *cf-const* $(:_C I)$ $\mathfrak{C}$ P, $:_C I$, $\mathfrak{C}]_{\circ}$

Components.

lemma *ntcf-obj-prod-base-components*:
shows *ntcf-obj-prod-base* $\mathfrak{C}$ I F P $f(|NTMap|) = (\lambda j \in_{\circ} :_C I(|Obj|). \ f\ j)$
and *ntcf-obj-prod-base* $\mathfrak{C}$ I F P $f(|NTDom|)$ = *cf-const* $(:_C I)$ $\mathfrak{C}$ P
and *ntcf-obj-prod-base* $\mathfrak{C}$ I F P $f(|NTCod|) = :\!\rightarrow\!: I\ F\ \mathfrak{C}$
and *ntcf-obj-prod-base* $\mathfrak{C}$ I F P $f(|NTDGDom|) = :_C I$
and *ntcf-obj-prod-base* $\mathfrak{C}$ I F P $f(|NTDGCod|) = \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-obj-coprod-base-components*:
shows *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P $f(|NTMap|) = (\lambda j \in_{\circ} :_C I(|Obj|). \ f\ j)$
and *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P $f(|NTDom|) = :\!\rightarrow\!: I\ F\ \mathfrak{C}$
and *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P $f(|NTCod|)$ = *cf-const* $(:_C I)$ $\mathfrak{C}$ P
and *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P $f(|NTDGDom|) = :_C I$
and *ntcf-obj-coprod-base* $\mathfrak{C}$ I F P $f(|NTDGCod|) = \mathfrak{C}$
⟨*proof*⟩

Duality.

lemma (**in** *cf-discrete*) *op-ntcf-ntcf-obj-coprod-base*[*cat-op-simps*]:
op-ntcf (*ntcf-obj-coprod-base* $\mathfrak{C}$ I F P f) =
ntcf-obj-prod-base (*op-cat* $\mathfrak{C}$) I F P f
⟨*proof*⟩

lemma (**in** *cf-discrete*) *op-ntcf-ntcf-obj-prod-base*[*cat-op-simps*]:
op-ntcf (*ntcf-obj-prod-base* $\mathfrak{C}$ I F P f) =
ntcf-obj-coprod-base (*op-cat* $\mathfrak{C}$) I F P f
⟨*proof*⟩

5.5.2 Natural transformation map

mk-VLambda *ntcf-obj-prod-base-components*(*1*)
|*vsv ntcf-obj-prod-base-NTMap-vsv*[*cat-cs-intros*]|
|*vdomain ntcf-obj-prod-base-NTMap-vdomain*[*cat-cs-simps*]|
|*app ntcf-obj-prod-base-NTMap-app*[*cat-cs-simps*]|

mk-VLambda *ntcf-obj-coprod-base-components*(*1*)
|*vsv ntcf-obj-coprod-base-NTMap-vsv*[*cat-cs-intros*]|
|*vdomain ntcf-obj-coprod-base-NTMap-vdomain*[*cat-cs-simps*]|
|*app ntcf-obj-coprod-base-NTMap-app*[*cat-cs-simps*]|

5.5.3 Projection natural transformation is a cone

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-cone*:
assumes $P \in_\circ \mathfrak{C}(|Obj|)$ **and** $\bigwedge a.\ a \in_\circ I \Longrightarrow f\ a : P \mapsto_{\mathfrak{C}} F\ a$
shows *ntcf-obj-prod-base* $\mathfrak{C}\ I\ F\ P\ f : P <_{CF.cone} :\!\rightarrow\!: I\ F\ \mathfrak{C} : :_C I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-coprod-base-is-cat-cocone*:
assumes $P \in_\circ \mathfrak{C}(|Obj|)$ **and** $\bigwedge a.\ a \in_\circ I \Longrightarrow f\ a : F\ a \mapsto_{\mathfrak{C}} P$
shows *ntcf-obj-coprod-base* $\mathfrak{C}\ I\ F\ P\ f$:
$:\!\rightarrow\!: I\ F\ \mathfrak{C} >_{CF.cocone} P : :_C I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-obj-prod*:
assumes $P \in_\circ \mathfrak{C}(|Obj|)$
and $\bigwedge a.\ a \in_\circ I \Longrightarrow f\ a : P \mapsto_{\mathfrak{C}} F\ a$
and $\bigwedge u'\ r'.$
$[\![\ u' : r' <_{CF.cone} :\!\rightarrow\!: I\ F\ \mathfrak{C} : :_C I \mapsto\mapsto_{C\alpha} \mathfrak{C}\ ]\!] \Longrightarrow$
$\exists!f'.$
$f' : r' \mapsto_{\mathfrak{C}} P \wedge$
$u' =$ *ntcf-obj-prod-base* $\mathfrak{C}\ I\ F\ P\ f \bullet_{NTCF}$ *ntcf-const* $(:_C I)\ \mathfrak{C}\ f'$
shows *ntcf-obj-prod-base* $\mathfrak{C}\ I\ F\ P\ f : P <_{CF.\prod} F : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-coprod-base-is-cat-obj-coprod*:
assumes $P \in_\circ \mathfrak{C}(|Obj|)$
and $\bigwedge a.\ a \in_\circ I \Longrightarrow f\ a : F\ a \mapsto_{\mathfrak{C}} P$
and $\bigwedge u'\ r'.\ [\![\ u' : :\!\rightarrow\!: I\ F\ \mathfrak{C} >_{CF.cocone} r' : :_C I \mapsto\mapsto_{C\alpha} \mathfrak{C}\ ]\!] \Longrightarrow$
$\exists!f'.$
$f' : P \mapsto_{\mathfrak{C}} r' \wedge$
$u' =$ *ntcf-const* $(:_C I)\ \mathfrak{C}\ f' \bullet_{NTCF}$ *ntcf-obj-coprod-base* $\mathfrak{C}\ I\ F\ P\ f$
shows *ntcf-obj-coprod-base* $\mathfrak{C}\ I\ F\ P\ f : F >_{CF.\coprod} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
(**is** ‹*?nc* $: F >_{CF.\coprod} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$›)
⟨*proof*⟩

6 Pullbacks and pushouts as limits and colimits

6.1 Pullback and pushout

6.1.1 Definition and elementary properties

The definitions and the elementary properties of the pullbacks and the pushouts can be found, for example, in Chapter III-3 and Chapter III-4 in [9].

locale *is-cat-pullback* =
  is-cat-limit α ‹→•←$_C$› $\mathfrak{C}$ ‹⟨$\mathfrak{a}$→$\mathfrak{g}$→$\mathfrak{o}$←$\mathfrak{f}$←$\mathfrak{b}$⟩$_{CF\mathfrak{C}}$› X x +
  cf-scospan α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$
  for α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$ X x

syntax *-is-cat-pullback* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - <$_{CF.pb}$ -→-→-←-←- ↦↦$_C$ı -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-pullback* ⇌ *is-cat-pullback*
translations x : X <$_{CF.pb}$ $\mathfrak{a}$→$\mathfrak{g}$→$\mathfrak{o}$←$\mathfrak{f}$←$\mathfrak{b}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-pullback α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$ X x

locale *is-cat-pushout* =
  is-cat-colimit α ‹←•→$_C$› $\mathfrak{C}$ ‹⟨$\mathfrak{a}$←$\mathfrak{g}$←$\mathfrak{o}$→$\mathfrak{f}$→$\mathfrak{b}$⟩$_{CF\mathfrak{C}}$› X x +
  cf-sspan α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$
  for α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$ X x

syntax *-is-cat-pushout* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ -←-←-→-→- >$_{CF.po}$ - ↦↦$_C$ı -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-pushout* ⇌ *is-cat-pushout*
translations x : $\mathfrak{a}$←$\mathfrak{g}$←$\mathfrak{o}$→$\mathfrak{f}$→$\mathfrak{b}$ >$_{CF.po}$ X ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-pushout α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$ X x

Rules.

lemma (**in** *is-cat-pullback*) *is-cat-pullback-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$
    and $\mathfrak{a}' = \mathfrak{a}$
    and $\mathfrak{g}' = \mathfrak{g}$
    and $\mathfrak{o}' = \mathfrak{o}$
    and $\mathfrak{f}' = \mathfrak{f}$
    and $\mathfrak{b}' = \mathfrak{b}$
    and $\mathfrak{C}' = \mathfrak{C}$
    and $X' = X$
  shows x : X' <$_{CF.pb}$ $\mathfrak{a}'$→$\mathfrak{g}'$→$\mathfrak{o}'$←$\mathfrak{f}'$←$\mathfrak{b}'$ ↦↦$_{C_{\alpha'}}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-pullback-def*
  |*intro is-cat-pullbackI*|
  |*dest is-cat-pullbackD*[*dest*]|
  |*elim is-cat-pullbackE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-pullbackD*

lemma (**in** *is-cat-pushout*) *is-cat-pushout-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$
    and $\mathfrak{a}' = \mathfrak{a}$
    and $\mathfrak{g}' = \mathfrak{g}$
    and $\mathfrak{o}' = \mathfrak{o}$
    and $\mathfrak{f}' = \mathfrak{f}$
    and $\mathfrak{b}' = \mathfrak{b}$
    and $\mathfrak{C}' = \mathfrak{C}$

and $X' = X$
shows $x : \mathfrak{a}'{\leftarrow}\mathfrak{g}'{\leftarrow}\mathfrak{o}'{\rightarrow}\mathfrak{f}'{\rightarrow}\mathfrak{b}' >_{CF.po} X' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-pushout-def*
|*intro is-cat-pushoutI*|
|*dest is-cat-pushoutD*[*dest*]|
|*elim is-cat-pushoutE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-pushoutD*

Duality.

lemma (**in** *is-cat-pullback*) *is-cat-pushout-op*:
op-ntcf $x : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-pullback*) *is-cat-pushout-op*′[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $x : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-pullback.is-cat-pushout-op*′

lemma (**in** *is-cat-pushout*) *is-cat-pullback-op*:
op-ntcf $x : X <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-pushout*) *is-cat-pullback-op*′[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $x : X <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-pushout.is-cat-pullback-op*′

Elementary properties.

lemma *cat-cone-cospan*:
assumes $x : X <_{CF.cone} \langle\mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b}\rangle_{CF\mathfrak{C}} : {\rightarrow}{\bullet}{\leftarrow}_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *cf-scospan* α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$
shows $x(|NTMap|)(|\mathfrak{o}_{SS}|) = \mathfrak{g} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|)$
and $x(|NTMap|)(|\mathfrak{o}_{SS}|) = \mathfrak{f} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
and $\mathfrak{g} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|) = \mathfrak{f} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
⟨*proof*⟩

lemma (**in** *is-cat-pullback*) *cat-pb-cone-cospan*:
shows $x(|NTMap|)(|\mathfrak{o}_{SS}|) = \mathfrak{g} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|)$
and $x(|NTMap|)(|\mathfrak{o}_{SS}|) = \mathfrak{f} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
and $\mathfrak{g} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|) = \mathfrak{f} \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
⟨*proof*⟩

lemma *cat-cocone-span*:
assumes $x : \langle\mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b}\rangle_{CF\mathfrak{C}} >_{CF.cocone} X : {\leftarrow}{\bullet}{\rightarrow}_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *cf-sspan* α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$
shows $x(|NTMap|)(|\mathfrak{o}_{SS}|) = x(|NTMap|)(|\mathfrak{a}_{SS}|) \circ_{A\mathfrak{C}} \mathfrak{g}$
and $x(|NTMap|)(|\mathfrak{o}_{SS}|) = x(|NTMap|)(|\mathfrak{b}_{SS}|) \circ_{A\mathfrak{C}} \mathfrak{f}$
and $x(|NTMap|)(|\mathfrak{a}_{SS}|) \circ_{A\mathfrak{C}} \mathfrak{g} = x(|NTMap|)(|\mathfrak{b}_{SS}|) \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

lemma (**in** *is-cat-pushout*) *cat-po-cocone-span*:

shows $x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{o}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} \mathfrak{g}$
and $x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{o}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} \mathfrak{f}$
and $x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} \mathfrak{g} = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

6.1.2 Universal property

lemma *is-cat-pullbackI′*:
assumes $x : X <_{CF.cone} \langle\mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} : \rightarrow\bullet\leftarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *cf-scospan* $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}$
and $\bigwedge x'\ X'.\ x' : X' <_{CF.cone} \langle\mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} : \rightarrow\bullet\leftarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
$\exists!f'.$
$f' : X' \mapsto_{\mathfrak{C}} X\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'$
shows $x : X <_{CF.pb} \mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *is-cat-pushoutI′*:
assumes $x : \langle\mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} >_{CF.cocone} X : \leftarrow\bullet\rightarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *cf-sspan* $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}$
and $\bigwedge x'\ X'.\ x' : \langle\mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} >_{CF.cocone} X' : \leftarrow\bullet\rightarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
$\exists!f'.$
$f' : X \mapsto_{\mathfrak{C}} X'\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis = f' \circ_{A\mathfrak{C}} x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis = f' \circ_{A\mathfrak{C}} x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis$
shows $x : \mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-pullback*) *cat-pb-unique-cone*:
assumes $x' : X' <_{CF.cone} \langle\mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} : \rightarrow\bullet\leftarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.$
$f' : X' \mapsto_{\mathfrak{C}} X\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-pullback*) *cat-pb-unique*:
assumes $x' : X' <_{CF.pb} \mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : X' \mapsto_{\mathfrak{C}} X \wedge x' = x \bullet_{NTCF}$ *ntcf-const* $\rightarrow\bullet\leftarrow_C\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (**in** *is-cat-pullback*) *cat-pb-unique′*:
assumes $x' : X' <_{CF.pb} \mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.$
$f' : X' \mapsto_{\mathfrak{C}} X\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis = x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-pushout*) *cat-po-unique-cocone*:
assumes $x' : \langle\mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b}\rangle_{CF\mathfrak{C}} >_{CF.cocone} X' : \leftarrow\bullet\rightarrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.$
$f' : X \mapsto_{\mathfrak{C}} X'\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis = f' \circ_{A\mathfrak{C}} x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{SS}\rrparenthesis\ \wedge$
$x'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis = f' \circ_{A\mathfrak{C}} x\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{SS}\rrparenthesis$
⟨*proof*⟩

lemma (**in** *is-cat-pushout*) *cat-po-unique*:
  assumes $x' : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'.\ f' : X \mapsto_{\mathfrak{C}} X' \wedge x' = ntcf\text{-}const \leftarrow\!\bullet\!\rightarrow_C \mathfrak{C}\ f' \bullet_{NTCF}\ x$
  ⟨*proof*⟩

lemma (**in** *is-cat-pushout*) *cat-po-unique′*:
  assumes $x' : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'.$
    $f' : X \mapsto_{\mathfrak{C}} X' \wedge$
    $x'(|NTMap|)(|\mathfrak{a}_{SS}|) = f' \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|) \wedge$
    $x'(|NTMap|)(|\mathfrak{b}_{SS}|) = f' \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
⟨*proof*⟩

lemma *cat-pullback-ex-is-iso-arr*:
  assumes $x : X <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $x' : X' <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : X' \mapsto_{iso\mathfrak{C}} X$
    and $x' = x \bullet_{NTCF}\ ntcf\text{-}const \rightarrow\!\bullet\!\leftarrow_C\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-pullback-ex-is-iso-arr′*:
  assumes $x : X <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $x' : X' <_{CF.pb} \mathfrak{a}{\rightarrow}\mathfrak{g}{\rightarrow}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : X' \mapsto_{iso\mathfrak{C}} X$
    and $x'(|NTMap|)(|\mathfrak{a}_{SS}|) = x(|NTMap|)(|\mathfrak{a}_{SS}|) \circ_{A\mathfrak{C}} f$
    and $x'(|NTMap|)(|\mathfrak{b}_{SS}|) = x(|NTMap|)(|\mathfrak{b}_{SS}|) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-pushout-ex-is-iso-arr*:
  assumes $x : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $x' : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : X \mapsto_{iso\mathfrak{C}} X'$
    and $x' = ntcf\text{-}const \leftarrow\!\bullet\!\rightarrow_C \mathfrak{C}\ f \bullet_{NTCF}\ x$
⟨*proof*⟩

lemma *cat-pushout-ex-is-iso-arr′*:
  assumes $x : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $x' : \mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\rightarrow}\mathfrak{f}{\rightarrow}\mathfrak{b} >_{CF.po} X' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : X \mapsto_{iso\mathfrak{C}} X'$
    and $x'(|NTMap|)(|\mathfrak{a}_{SS}|) = f \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{a}_{SS}|)$
    and $x'(|NTMap|)(|\mathfrak{b}_{SS}|) = f \circ_{A\mathfrak{C}} x(|NTMap|)(|\mathfrak{b}_{SS}|)$
⟨*proof*⟩

7 Equalizers and coequalizers as limits and colimits

7.1 Equalizer and coequalizer

7.1.1 Definition and elementary properties

See [2][6].

locale *is-cat-equalizer* =
is-cat-limit α ‹$\Uparrow_C$ ($\mathfrak{a}_{PL}$ F) ($\mathfrak{b}_{PL}$ F) F› $\mathfrak{C}$ ‹$\Uparrow\rightarrow\Uparrow_{CF}$ $\mathfrak{C}$ ($\mathfrak{a}_{PL}$ F) ($\mathfrak{b}_{PL}$ F) F $\mathfrak{a}$ $\mathfrak{b}$ F'› E ε +
F': *vsv* F'
for α $\mathfrak{a}$ $\mathfrak{b}$ F F' $\mathfrak{C}$ E ε +
assumes *cat-eq-F-in-Vset*[*cat-lim-cs-intros*]: $F \in_\circ$ *Vset* α
and *cat-eq-F-ne*[*cat-lim-cs-intros*]: $F \neq 0$
and *cat-eq-F'-vdomain*[*cat-lim-cs-simps*]: $\mathcal{D}_\circ$ $F' = F$
and *cat-eq-F'-app-is-arr*[*cat-lim-cs-intros*]: $\mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$

syntax *-is-cat-equalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $<_{CF.eq}$ '(-,-,-,-') :/ $\Uparrow_C$ $\mapsto\mapsto_{C1}$ -)› [*51, 51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-cat-equalizer* $\rightleftharpoons$ *is-cat-equalizer*
translations ε : E $<_{CF.eq}$ ($\mathfrak{a}$,$\mathfrak{b}$,F,F') : $\Uparrow_C$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ $\rightleftharpoons$
CONST is-cat-equalizer α $\mathfrak{a}$ $\mathfrak{b}$ F F' $\mathfrak{C}$ E ε

locale *is-cat-coequalizer* =
is-cat-colimit α ‹$\Uparrow_C$ ($\mathfrak{b}_{PL}$ F) ($\mathfrak{a}_{PL}$ F) F› $\mathfrak{C}$ ‹$\Uparrow\rightarrow\Uparrow_{CF}$ $\mathfrak{C}$ ($\mathfrak{b}_{PL}$ F) ($\mathfrak{a}_{PL}$ F) F $\mathfrak{b}$ $\mathfrak{a}$ F'› E ε +
F': *vsv* F'
for α $\mathfrak{a}$ $\mathfrak{b}$ F F' $\mathfrak{C}$ E ε +
assumes *cat-coeq-F-in-Vset*[*cat-lim-cs-intros*]: $F \in_\circ$ *Vset* α
and *cat-coeq-F-ne*[*cat-lim-cs-intros*]: $F \neq 0$
and *cat-coeq-F'-vdomain*[*cat-lim-cs-simps*]: $\mathcal{D}_\circ$ $F' = F$
and *cat-coeq-F'-app-is-arr*[*cat-lim-cs-intros*]: $\mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$

syntax *-is-cat-coequalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ '(-,-,-,-') $>_{CF.coeq}$ - :/ $\Uparrow_C$ $\mapsto\mapsto_{C1}$ -)› [*51, 51, 51, 51, 51, 51*] *51*)
syntax-consts *-is-cat-coequalizer* $\rightleftharpoons$ *is-cat-coequalizer*
translations ε : ($\mathfrak{a}$,$\mathfrak{b}$,F,F') $>_{CF.coeq}$ E : $\Uparrow_C$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$ $\rightleftharpoons$
CONST is-cat-coequalizer α $\mathfrak{a}$ $\mathfrak{b}$ F F' $\mathfrak{C}$ E ε

Rules.

lemma (**in** *is-cat-equalizer*) *is-cat-equalizer-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$
and $E' = E$
and $\mathfrak{a}' = \mathfrak{a}$
and $\mathfrak{b}' = \mathfrak{b}$
and $F'' = F$
and $F''' = F'$
and $\mathfrak{C}' = \mathfrak{C}$
shows ε : E' $<_{CF.eq}$ ($\mathfrak{a}'$,$\mathfrak{b}'$,F'',F''') : $\Uparrow_C$ $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-equalizer-def*[*unfolded is-cat-equalizer-axioms-def*]
|*intro is-cat-equalizerI*|
|*dest is-cat-equalizerD*[*dest*]|
|*elim is-cat-equalizerE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-equalizerD*(*1*)

[6] https://en.wikipedia.org/wiki/Equaliser_(mathematics)

lemma (**in** *is-cat-coequalizer*) *is-cat-coequalizer-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$
and $E' = E$
and $\mathfrak{a}' = \mathfrak{a}$
and $\mathfrak{b}' = \mathfrak{b}$
and $F'' = F$
and $F''' = F'$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : (\mathfrak{a}',\mathfrak{b}',F'',F''') >_{CF.coeq} E' : \Uparrow_C \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-coequalizer-def*[*unfolded is-cat-coequalizer-axioms-def*]
|*intro is-cat-coequalizerI*|
|*dest is-cat-coequalizerD*[*dest*]|
|*elim is-cat-coequalizerE*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-coequalizerD*(*1*)

Elementary properties.

lemma (**in** *is-cat-equalizer*)
cat-eq-$\mathfrak{a}$[*cat-lim-cs-intros*]: $\mathfrak{a} \in_\circ \mathfrak{C}(|Obj|)$
and *cat-eq-*$\mathfrak{b}$[*cat-lim-cs-intros*]: $\mathfrak{b} \in_\circ \mathfrak{C}(|Obj|)$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*)
cat-coeq-$\mathfrak{a}$[*cat-lim-cs-intros*]: $\mathfrak{a} \in_\circ \mathfrak{C}(|Obj|)$
and *cat-coeq-*$\mathfrak{b}$[*cat-lim-cs-intros*]: $\mathfrak{b} \in_\circ \mathfrak{C}(|Obj|)$
⟨*proof*⟩

sublocale *is-cat-equalizer* ⊆ *cf-parallel* α ‹$\mathfrak{a}_{PL}$ F› ‹$\mathfrak{b}_{PL}$ F› F $\mathfrak{a}$ $\mathfrak{b}$ F' $\mathfrak{C}$
⟨*proof*⟩

sublocale *is-cat-coequalizer* ⊆ *cf-parallel* α ‹$\mathfrak{b}_{PL}$ F› ‹$\mathfrak{a}_{PL}$ F› F $\mathfrak{b}$ $\mathfrak{a}$ F' $\mathfrak{C}$
⟨*proof*⟩

Duality.

lemma (**in** *is-cat-equalizer*) *is-cat-coequalizer-op*:
op-ntcf $\varepsilon : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E : \Uparrow_C \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer*) *is-cat-coequalizer-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\varepsilon : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-equalizer.is-cat-coequalizer-op'*

lemma (**in** *is-cat-coequalizer*) *is-cat-equalizer-op*:
op-ntcf $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*) *is-cat-equalizer-op'*[*cat-op-intros*]:
assumes $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-coequalizer.is-cat-equalizer-op'*

Further properties.

lemma (**in** *category*) *cat-cf-parallel-*$\mathfrak{ab}$:
assumes *vsv* F'
and $F \in_\circ Vset\ \alpha$
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
and $\mathfrak{a} \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{b} \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *cf-parallel* $\alpha\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F\ \mathfrak{a}\ \mathfrak{b}\ F'\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-parallel-*$\mathfrak{ba}$:
assumes *vsv* F'
and $F \in_\circ Vset\ \alpha$
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and $\mathfrak{a} \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{b} \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *cf-parallel* $\alpha\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F\ \mathfrak{b}\ \mathfrak{a}\ F'\ \mathfrak{C}$
⟨*proof*⟩

lemma *cat-cone-cf-par-eps-NTMap-app*:
assumes ε :
$E <_{CF.cone} \Uparrow\!\rightarrow\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F\ \mathfrak{a}\ \mathfrak{b}\ F'$:
$\Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *vsv* F'
and $F \in_\circ Vset\ \alpha$
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
and $\mathfrak{f} \in_\circ F$
shows $\varepsilon(\!|NTMap|\!)(\!|\mathfrak{b}_{PL}\ F|\!) = F'(\!|\mathfrak{f}|\!) \circ_{A\mathfrak{C}} \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!)$
⟨*proof*⟩

lemma *cat-cocone-cf-par-eps-NTMap-app*:
assumes ε :
$\Uparrow\!\rightarrow\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F\ \mathfrak{b}\ \mathfrak{a}\ F' >_{CF.cocone} E$:
$\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *vsv* F'
and $F \in_\circ Vset\ \alpha$
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and $\mathfrak{f} \in_\circ F$
shows $\varepsilon(\!|NTMap|\!)(\!|\mathfrak{b}_{PL}\ F|\!) = \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!) \circ_{A\mathfrak{C}} F'(\!|\mathfrak{f}|\!)$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer*) *cat-eq-eps-NTMap-app*:
assumes $\mathfrak{f} \in_\circ F$
shows $\varepsilon(\!|NTMap|\!)(\!|\mathfrak{b}_{PL}\ F|\!) = F'(\!|\mathfrak{f}|\!) \circ_{A\mathfrak{C}} \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!)$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*) *cat-coeq-eps-NTMap-app*:
assumes $\mathfrak{f} \in_\circ F$
shows $\varepsilon(\!|NTMap|\!)(\!|\mathfrak{b}_{PL}\ F|\!) = \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!) \circ_{A\mathfrak{C}} F'(\!|\mathfrak{f}|\!)$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer*) *cat-eq-Comp-eq*:
assumes $\mathfrak{g} \in_\circ F$ **and** $\mathfrak{f} \in_\circ F$
shows $F'(\!|\mathfrak{g}|\!) \circ_{A\mathfrak{C}} \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!) = F'(\!|\mathfrak{f}|\!) \circ_{A\mathfrak{C}} \varepsilon(\!|NTMap|\!)(\!|\mathfrak{a}_{PL}\ F|\!)$

$\langle proof \rangle$

lemma (**in** *is-cat-coequalizer*) *cat-coeq-Comp-eq*:
assumes $\mathfrak{g} \in_\circ F$ **and** $\mathfrak{f} \in_\circ F$
shows $\varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} F'\llparenthesis \mathfrak{g}\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} F'\llparenthesis \mathfrak{f}\rrparenthesis$
$\langle proof \rangle$

7.1.2 Universal property

lemma *is-cat-equalizerI'*:
assumes ε :
$E <_{CF.cone} \Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F\ \mathfrak{a}\ \mathfrak{b}\ F'$:
$\Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *vsv* F'
and $F \in_\circ$ *Vset* α
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'\llparenthesis \mathfrak{f}\rrparenthesis : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
and $\mathfrak{f} \in_\circ F$
and $\bigwedge\varepsilon'\ E'.\ \varepsilon'$:
$E' <_{CF.cone} \Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F\ \mathfrak{a}\ \mathfrak{b}\ F'$:
$\Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
$\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} f'$
shows $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
$\langle proof \rangle$

lemma *is-cat-coequalizerI'*:
assumes ε :
$\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F\ \mathfrak{b}\ \mathfrak{a}\ F' >_{CF.cocone} E$:
$\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *vsv* F'
and $F \in_\circ$ *Vset* α
and $\mathcal{D}_\circ\ F' = F$
and $\bigwedge\mathfrak{f}.\ \mathfrak{f} \in_\circ F \Longrightarrow F'\llparenthesis \mathfrak{f}\rrparenthesis : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and $\mathfrak{f} \in_\circ F$
and $\bigwedge\varepsilon'\ E'.\ \varepsilon'$:
$\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F\ \mathfrak{b}\ \mathfrak{a}\ F' >_{CF.cocone} E'$:
$\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
$\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis = f' \circ_{A\mathfrak{C}} \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis$
shows $\varepsilon : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
$\langle proof \rangle$

lemma (**in** *is-cat-equalizer*) *cat-eq-unique-cone*:
assumes ε' :
$E' <_{CF.cone} \Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F\ \mathfrak{a}\ \mathfrak{b}\ F' : \Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
(**is** ‹$\varepsilon' : E' <_{CF.cone}$ *?II-II* : *?II* $\mapsto\mapsto_{C\alpha} \mathfrak{C}$›)
shows $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} f'$
$\langle proof \rangle$

lemma (**in** *is-cat-equalizer*) *cat-eq-unique*:
assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon' = \varepsilon \bullet_{NTCF}$ *ntcf-const* $(\Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F)\ \mathfrak{C}\ f'$
$\langle proof \rangle$

lemma (**in** *is-cat-equalizer*) *cat-eq-unique'*:
assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis \mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} f'$
$\langle proof \rangle$

lemma (**in** *is-cat-coequalizer*) *cat-coeq-unique-cocone*:
  assumes ε' :
    $\Uparrow\rightarrow\Uparrow_{CF}\ \mathfrak{C}\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F\ \mathfrak{b}\ \mathfrak{a}\ F' >_{CF.cocone} E'$:
    $\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    (**is** ‹ε' : *?II-II* $>_{CF.cocone} E'$: *?II* $\mapsto\mapsto_{C\alpha} \mathfrak{C}$›)
  shows $\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis = f' \circ_{A\mathfrak{C}} \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*) *cat-coeq-unique*:
  assumes $\varepsilon' : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E' : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'$.
    $f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon' =$ *ntcf-const* $(\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F)\ \mathfrak{C}\ f' \bullet_{NTCF} \varepsilon$
  ⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*) *cat-coeq-unique'*:
  assumes $\varepsilon' : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E' : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis = f' \circ_{A\mathfrak{C}} \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis$
⟨*proof*⟩

lemma *cat-equalizer-ex-is-iso-arr*:
  assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$
    and $\varepsilon' = \varepsilon \bullet_{NTCF}$ *ntcf-const* $(\Uparrow_C\ (\mathfrak{a}_{PL}\ F)\ (\mathfrak{b}_{PL}\ F)\ F)\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-equalizer-ex-is-iso-arr'*:
  assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},F,F') : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$
    and $\varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} f$
    and $\varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{PL}\ F\rrparenthesis = \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{PL}\ F\rrparenthesis \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-coequalizer-ex-is-iso-arr*:
  assumes $\varepsilon : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\varepsilon' : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E' : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$
    and $\varepsilon' =$ *ntcf-const* $(\Uparrow_C\ (\mathfrak{b}_{PL}\ F)\ (\mathfrak{a}_{PL}\ F)\ F)\ \mathfrak{C}\ f \bullet_{NTCF} \varepsilon$
⟨*proof*⟩

lemma *cat-coequalizer-ex-is-iso-arr'*:
  assumes $\varepsilon : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\varepsilon' : (\mathfrak{a},\mathfrak{b},F,F') >_{CF.coeq} E' : \Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$
    and $\varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis = f \circ_{A\mathfrak{C}} \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis$
    and $\varepsilon'\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{PL}\ F\rrparenthesis = f \circ_{A\mathfrak{C}} \varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{b}_{PL}\ F\rrparenthesis$
⟨*proof*⟩

7.1.3 Further properties

lemma (**in** *is-cat-equalizer*) *cat-eq-is-monic-arr*:
  — See subsection 3.3 in [3].
  $\varepsilon\llparenthesis NTMap\rrparenthesis\llparenthesis\mathfrak{a}_{PL}\ F\rrparenthesis : E \mapsto_{mon\mathfrak{C}} \mathfrak{a}$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer*) *cat-coeq-is-epic-arr*:

$\varepsilon(\|NTMap\|)(\|\mathfrak{a}_{PL}\ F\|) : \mathfrak{a} \mapsto_{epi\mathfrak{C}} E$
⟨proof⟩

7.2 Equalizer and coequalizer for two arrows

7.2.1 Definition and elementary properties

See [2][7].

locale *is-cat-equalizer-2* =
is-cat-limit α ‹↑↑$_C$ $\mathfrak{a}_{PL2}$ $\mathfrak{b}_{PL2}$ $\mathfrak{g}_{PL}$ $\mathfrak{f}_{PL}$› $\mathfrak{C}$ ‹↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}_{PL2}$ $\mathfrak{b}_{PL2}$ $\mathfrak{g}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$› E ε
for α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ E ε +
assumes *cat-eq-*$\mathfrak{g}$[*cat-lim-cs-intros*]: $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
and *cat-eq-*$\mathfrak{f}$[*cat-lim-cs-intros*]: $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$

syntax *-is-cat-equalizer-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - <$_{CF.eq}$ '(-,-,-,-') :/ ↑↑$_C$ ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-equalizer-2* ⇌ *is-cat-equalizer-2*
translations ε : E <$_{CF.eq}$ ($\mathfrak{a}$,$\mathfrak{b}$,$\mathfrak{g}$,$\mathfrak{f}$) : ↑↑$_C$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
CONST is-cat-equalizer-2 α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ E ε

locale *is-cat-coequalizer-2* =
is-cat-colimit
α ‹↑↑$_C$ $\mathfrak{b}_{PL2}$ $\mathfrak{a}_{PL2}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$› $\mathfrak{C}$ ‹↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{b}_{PL2}$ $\mathfrak{a}_{PL2}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$› E ε
for α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ E ε +
assumes *cat-coeq-*$\mathfrak{g}$[*cat-lim-cs-intros*]: $\mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and *cat-coeq-*$\mathfrak{f}$[*cat-lim-cs-intros*]: $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$

syntax *-is-cat-coequalizer-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ '(-,-,-,-') >$_{CF.coeq}$ - :/ ↑↑$_C$ ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-coequalizer-2* ⇌ *is-cat-coequalizer-2*
translations ε : ($\mathfrak{a}$,$\mathfrak{b}$,$\mathfrak{g}$,$\mathfrak{f}$) >$_{CF.coeq}$ E : ↑↑$_C$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
CONST is-cat-coequalizer-2 α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ E ε

Rules.

lemma (**in** *is-cat-equalizer-2*) *is-cat-equalizer-2-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$
and $E' = E$
and $\mathfrak{a}' = \mathfrak{a}$
and $\mathfrak{b}' = \mathfrak{b}$
and $\mathfrak{g}' = \mathfrak{g}$
and $\mathfrak{f}' = \mathfrak{f}$
and $\mathfrak{C}' = \mathfrak{C}$
shows ε : E' <$_{CF.eq}$ ($\mathfrak{a}'$,$\mathfrak{b}'$,$\mathfrak{g}'$,$\mathfrak{f}'$) : ↑↑$_C$ ↦↦$_{C\alpha'}$ $\mathfrak{C}'$
⟨proof⟩

mk-ide rf *is-cat-equalizer-2-def*[*unfolded is-cat-equalizer-2-axioms-def*]
|*intro is-cat-equalizer-2I*|
|*dest is-cat-equalizer-2D*[*dest*]|
|*elim is-cat-equalizer-2E*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-equalizer-2D*(*1*)

lemma (**in** *is-cat-coequalizer-2*) *is-cat-coequalizer-2-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$
and $E' = E$
and $\mathfrak{a}' = \mathfrak{a}$

[7] https://en.wikipedia.org/wiki/Equaliser_(mathematics)

and $\mathfrak{b}' = \mathfrak{b}$
and $\mathfrak{g}' = \mathfrak{g}$
and $\mathfrak{f}' = \mathfrak{f}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : (\mathfrak{a}',\mathfrak{b}',\mathfrak{g}',\mathfrak{f}') >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *is-cat-coequalizer-2-def*[*unfolded is-cat-coequalizer-2-axioms-def*]
|*intro is-cat-coequalizer-2I*|
|*dest is-cat-coequalizer-2D*[*dest*]|
|*elim is-cat-coequalizer-2E*[*elim*]|

lemmas [*cat-lim-cs-intros*] = *is-cat-coequalizer-2D*(*1*)

Helper lemmas.

lemma *cat-eq-F′-helper*:
$(\lambda f \in_\circ set\ \{\mathfrak{f}_{PL}, \mathfrak{g}_{PL}\}.\ (f = \mathfrak{g}_{PL}\ ?\ \mathfrak{g} : \mathfrak{f})) =$
$(\lambda f \in_\circ set\ \{\mathfrak{f}_{PL}, \mathfrak{g}_{PL}\}.\ (f = \mathfrak{f}_{PL}\ ?\ \mathfrak{f} : \mathfrak{g}))$
⟨*proof*⟩

Elementary properties.

sublocale *is-cat-equalizer-2* ⊆ *cf-parallel-2* $\alpha\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{C}$
⟨*proof*⟩

sublocale *is-cat-coequalizer-2* ⊆ *cf-parallel-2* $\alpha\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer-2*) *cat-equalizer-2-is-cat-equalizer*:
ε :
$E <_{CF.eq} (\mathfrak{a},\mathfrak{b},set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\},(\lambda f \in_\circ set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\}.\ (f = \mathfrak{f}_{PL}\ ?\ \mathfrak{f} : \mathfrak{g}))) :$
$\Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coequalizer-2-is-cat-coequalizer*:
ε :
$(\mathfrak{a},\mathfrak{b},set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\},(\lambda f \in_\circ set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\}.\ (f = \mathfrak{f}_{PL}\ ?\ \mathfrak{f} : \mathfrak{g}))) >_{CF.coeq} E :$
$\Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-equalizer-is-cat-equalizer-2*:
assumes ε :
$E <_{CF.eq} (\mathfrak{a},\mathfrak{b},set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\},(\lambda f \in_\circ set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\}.\ (f = \mathfrak{f}_{PL}\ ?\ \mathfrak{f} : \mathfrak{g}))) :$
$\Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-coequalizer-is-cat-coequalizer-2*:
assumes ε :
$(\mathfrak{a},\mathfrak{b},set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\},(\lambda f \in_\circ set\ \{\mathfrak{g}_{PL}, \mathfrak{f}_{PL}\}.\ (f = \mathfrak{f}_{PL}\ ?\ \mathfrak{f} : \mathfrak{g}))) >_{CF.coeq} E :$
$\Uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

Duality.

lemma (**in** *is-cat-equalizer-2*) *is-cat-coequalizer-2-op*:
op-ntcf $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer-2*) *is-cat-coequalizer-2-op'*[*cat-op-intros*]:
  assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
  shows $op\text{-}ntcf\ \varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
  ⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-equalizer-2.is-cat-coequalizer-2-op'*

lemma (**in** *is-cat-coequalizer-2*) *is-cat-equalizer-2-op*:
  $op\text{-}ntcf\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} op\text{-}cat\ \mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *is-cat-equalizer-2-op'*[*cat-op-intros*]:
  assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
  shows $op\text{-}ntcf\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
  ⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-coequalizer-2.is-cat-equalizer-2-op'*

Further properties.

lemma (**in** *category*) *cat-cf-parallel-2-cat-equalizer*:
  assumes $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$ **and** $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
  shows $cf\text{-}parallel\text{-}2\ \alpha\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *category*) *cat-cf-parallel-2-cat-coequalizer*:
  assumes $\mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$ **and** $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
  shows $cf\text{-}parallel\text{-}2\ \alpha\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}\ \mathfrak{C}$
  ⟨*proof*⟩

lemma *cat-cone-cf-par-2-eps-NTMap-app*:
  assumes ε :
    $E <_{CF.cone} \uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f} : \uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
    and $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
  shows
    $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \mathfrak{g} \circ_{A\mathfrak{C}} \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈$
    $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \mathfrak{f} \circ_{A\mathfrak{C}} \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈$
⟨*proof*⟩

lemma *cat-cocone-cf-par-2-eps-NTMap-app*:
  assumes ε :
    $\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g} >_{CF.cocone} E$:
    $\uparrow\uparrow_C\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
    and $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
  shows
    $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈ \circ_{A\mathfrak{C}} \mathfrak{g}$
    $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈ \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

lemma (**in** *is-cat-equalizer-2*) *cat-eq-2-eps-NTMap-app*:
  $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \mathfrak{g} \circ_{A\mathfrak{C}} \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈$
  $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \mathfrak{f} \circ_{A\mathfrak{C}} \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coeq-2-eps-NTMap-app*:
  $\varepsilon⦇NTMap⦈⦇\mathfrak{b}_{PL2}⦈ = \varepsilon⦇NTMap⦈⦇\mathfrak{a}_{PL2}⦈ \circ_{A\mathfrak{C}} \mathfrak{g}$

$\varepsilon(|NTMap|)(|\mathfrak{b}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-Comp-eq*:
  $\mathfrak{g} \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) = \mathfrak{f} \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
  $\mathfrak{f} \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) = \mathfrak{g} \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
  ⟨*proof*⟩

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-Comp-eq*:
  $\varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} \mathfrak{g} = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} \mathfrak{f}$
  $\varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} \mathfrak{f} = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} \mathfrak{g}$
  ⟨*proof*⟩

7.2.2 Universal property

lemma *is-cat-equalizer-2I′*:
  assumes ε :
    $E <_{CF.cone} \uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f} : \uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
    and $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
    and $\bigwedge\varepsilon'\ E'.\ \varepsilon'$:
      $E' <_{CF.cone} \uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$:
      $\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
      $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} f'$
  shows $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *is-cat-coequalizer-2I′*:
  assumes ε :
    $\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g} >_{CF.cocone} E$:
    $\uparrow\uparrow_C\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
    and $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
    and $\bigwedge\varepsilon'\ E'.\ \varepsilon'$:
      $\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g} >_{CF.cocone} E'$:
      $\uparrow\uparrow_C\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C} \Longrightarrow$
      $\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = f' \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
  shows $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique-cone*:
  assumes ε' :
    $E' <_{CF.cone} \uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$:
    $\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} f'$
  ⟨*proof*⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique*:
  assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows
    $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon' = \varepsilon \bullet_{NTCF}$ *ntcf-const* $(\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL})\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique′*:
  assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\exists!f'.\ f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coeq-2-unique-cocone*:
assumes ε' :
$\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{b}_{PL2}$ $\mathfrak{a}_{PL2}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$ $>_{CF.cocone}$ E' :
$\uparrow\uparrow_C$ $\mathfrak{b}_{PL2}$ $\mathfrak{a}_{PL2}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
shows $\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = f' \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coeq-2-unique*:
assumes $\varepsilon' : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.$
$f' : E \mapsto_{\mathfrak{C}} E' \wedge$
$\varepsilon' = ntcf\text{-}const\ (\uparrow\uparrow_C\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL})\ \mathfrak{C}\ f' \bullet_{NTCF} \varepsilon$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coeq-2-unique'*:
assumes $\varepsilon' : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\exists!f'.\ f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = f' \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
⟨*proof*⟩

lemma *cat-equalizer-2-ex-is-iso-arr*:
assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$
and $\varepsilon' = \varepsilon \bullet_{NTCF}\ ntcf\text{-}const\ (\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL})\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-equalizer-2-ex-is-iso-arr'*:
assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$
and $\varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) \circ_{A\mathfrak{C}} f$
and $\varepsilon'(|NTMap|)(|\mathfrak{b}_{PL2}|) = \varepsilon(|NTMap|)(|\mathfrak{b}_{PL2}|) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-coequalizer-2-ex-is-iso-arr*:
assumes $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon' : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$
and $\varepsilon' = ntcf\text{-}const\ (\uparrow\uparrow_C\ \mathfrak{b}_{PL2}\ \mathfrak{a}_{PL2}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL})\ \mathfrak{C}\ f \bullet_{NTCF} \varepsilon$
⟨*proof*⟩

lemma *cat-coequalizer-2-ex-is-iso-arr'*:
assumes $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon' : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$
and $\varepsilon'(|NTMap|)(|\mathfrak{a}_{PL2}|) = f \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|)$
and $\varepsilon'(|NTMap|)(|\mathfrak{b}_{PL2}|) = f \circ_{A\mathfrak{C}} \varepsilon(|NTMap|)(|\mathfrak{b}_{PL2}|)$
⟨*proof*⟩

7.2.3 Further properties

lemma (**in** *is-cat-equalizer-2*) *cat-eq-2-is-monic-arr*:
$\varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) : E \mapsto_{mon\mathfrak{C}} \mathfrak{a}$
⟨*proof*⟩

lemma (**in** *is-cat-coequalizer-2*) *cat-coeq-2-is-epic-arr*:
$\varepsilon(|NTMap|)(|\mathfrak{a}_{PL2}|) : \mathfrak{a} \mapsto_{epi\mathfrak{C}} E$
⟨*proof*⟩

7.3 Equalizer cone

7.3.1 Definition and elementary properties

definition *ntcf-equalizer-base* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
 where *ntcf-equalizer-base* $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E e =
  [
   $(\lambda x \in_\circ \uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}(\!|Obj|\!).\ e\ x)$,
   cf-const $(\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL})\ \mathfrak{C}\ E$,
   $\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$,
   $\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}$,
   $\mathfrak{C}$
  $]_\circ$

Components.

lemma *ntcf-equalizer-base-components*:
 shows *ntcf-equalizer-base* $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E $e(\!|NTMap|\!)$ =
  $(\lambda x \in_\circ \uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}(\!|Obj|\!).\ e\ x)$
  and [*cat-lim-cs-simps*]: *ntcf-equalizer-base* $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E $e(\!|NTDom|\!)$ =
   cf-const $(\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL})\ \mathfrak{C}\ E$
  and [*cat-lim-cs-simps*]: *ntcf-equalizer-base* $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E $e(\!|NTCod|\!)$ =
   $\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$
  and [*cat-lim-cs-simps*]:
   ntcf-equalizer-base $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E $e(\!|NTDGDom|\!) = \uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}$
  and [*cat-lim-cs-simps*]:
   ntcf-equalizer-base $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E $e(\!|NTDGCod|\!) = \mathfrak{C}$
 ⟨*proof*⟩

7.3.2 Natural transformation map

mk-VLambda *ntcf-equalizer-base-components*(*1*)
 |*vsv ntcf-equalizer-base-NTMap-vsv*[*cat-lim-cs-intros*]|
 |*vdomain ntcf-equalizer-base-NTMap-vdomain*[*cat-lim-cs-simps*]|
 |*app ntcf-equalizer-base-NTMap-app*[*cat-lim-cs-simps*]|

7.3.3 Equalizer cone is a cone

lemma (**in** *category*) *cat-ntcf-equalizer-base-is-cat-cone*:
 assumes $e\ \mathfrak{a}_{PL2} : E \mapsto_{\mathfrak{C}} \mathfrak{a}$
  and $e\ \mathfrak{b}_{PL2} : E \mapsto_{\mathfrak{C}} \mathfrak{b}$
  and $e\ \mathfrak{b}_{PL2} = \mathfrak{g} \circ_{A\mathfrak{C}} e\ \mathfrak{a}_{PL2}$
  and $e\ \mathfrak{b}_{PL2} = \mathfrak{f} \circ_{A\mathfrak{C}} e\ \mathfrak{a}_{PL2}$
  and $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
  and $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
 shows *ntcf-equalizer-base* $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ E e :
  $E <_{CF.cone} \uparrow\uparrow\rightarrow\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$:
  $\uparrow\uparrow_C\ \mathfrak{a}_{PL2}\ \mathfrak{b}_{PL2}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8 Pointed arrows and natural transformations

8.1 Pointed arrow

The terminology that is used in this section deviates from convention: a pointed arrow is merely an arrow in *Set* from a singleton set to another set.

8.1.1 Definition and elementary properties

See Chapter III-2 in [9].

definition *ntcf-paa* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-paa* $\mathfrak{a}$ B b = $[(\lambda a \in_\circ set\ \{\mathfrak{a}\}.\ b),\ set\ \{\mathfrak{a}\},\ B]_\circ$

Components.

lemma *ntcf-paa-components*:
  shows *ntcf-paa* $\mathfrak{a}$ B $b(|ArrVal|) = (\lambda a \in_\circ set\ \{\mathfrak{a}\}.\ b)$
    and [*cat-cs-simps*]: *ntcf-paa* $\mathfrak{a}$ B $b(|ArrDom|) = set\ \{\mathfrak{a}\}$
    and [*cat-cs-simps*]: *ntcf-paa* $\mathfrak{a}$ B $b(|ArrCod|) = B$
  ⟨*proof*⟩

8.1.2 Arrow value

mk-VLambda *ntcf-paa-components*(*1*)
  |*vsv ntcf-paa-ArrVal-vsv*[*cat-cs-intros*]|
  |*vdomain ntcf-paa-ArrVal-vdomain*[*cat-cs-simps*]|
  |*app ntcf-paa-ArrVal-app*[*unfolded vsingleton-iff*, *cat-cs-simps*]|

8.1.3 Pointed arrow is an arrow in *Set*

lemma (**in** $\mathcal{Z}$) *ntcf-paa-is-arr*:
  assumes $\mathfrak{a} \in_\circ$ *cat-Set* $\alpha(|Obj|)$ **and** $A \in_\circ$ *cat-Set* $\alpha(|Obj|)$ **and** $a \in_\circ A$
  shows *ntcf-paa* $\mathfrak{a}$ A a : $set\ \{\mathfrak{a}\} \mapsto_{cat\text{-}Set\ \alpha} A$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *ntcf-paa-is-arr′*[*cat-cs-intros*]:
  assumes $\mathfrak{a} \in_\circ$ *cat-Set* $\alpha(|Obj|)$
    and $A \in_\circ$ *cat-Set* $\alpha(|Obj|)$
    and $a \in_\circ A$
    and $A' = set\ \{\mathfrak{a}\}$
    and $B' = A$
    and $\mathfrak{C}' =$ *cat-Set* α
  shows *ntcf-paa* $\mathfrak{a}$ A a : $A' \mapsto_{\mathfrak{C}'} B'$
  ⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-paa-is-arr′*

8.1.4 Further properties

lemma *ntcf-paa-injective*[*cat-cs-simps*]:
  ntcf-paa $\mathfrak{a}$ A b = *ntcf-paa* $\mathfrak{a}$ A c $\longleftrightarrow b = c$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *ntcf-paa-ArrVal*:
  assumes F : $set\ \{\mathfrak{a}\} \mapsto_{cat\text{-}Set\ \alpha} X$
  shows *ntcf-paa* $\mathfrak{a}$ X $(F(|ArrVal|)(|\mathfrak{a}|)) = F$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *ntcf-paa-ArrVal'*:
  assumes F : *set* $\{\mathfrak{a}\} \mapsto_{cat\text{-}Set\ \alpha} X$ **and** $a = \mathfrak{a}$
  shows *ntcf-paa* $\mathfrak{a}$ X $(F(\!|ArrVal|\!)(\!|a|\!)) = F$
  ⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *ntcf-paa-Comp-right*[*cat-cs-simps*]:
  assumes $F : A \mapsto_{cat\text{-}Set\ \alpha} B$
    and $\mathfrak{a} \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
    and $a \in_\circ A$
  shows $F \circ_{A\, cat\text{-}Set\ \alpha}$ *ntcf-paa* $\mathfrak{a}$ A a = *ntcf-paa* $\mathfrak{a}$ B $(F(\!|ArrVal|\!)(\!|a|\!))$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = $\mathcal{Z}$.*ntcf-paa-Comp-right*

8.2 Pointed natural transformation

8.2.1 Definition and elementary properties

See Chapter III-2 in [9].

definition *ntcf-pointed* :: $V \Rightarrow V \Rightarrow V$
  where *ntcf-pointed* α $\mathfrak{a}$ =
    [
      (
        $\lambda x \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$.
          [
            $(\lambda f \in_\circ$ *Hom* (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) x. $f(\!|ArrVal|\!)(\!|\mathfrak{a}|\!))$,
            Hom (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) x,
            x
          ]$_\circ$
      ),
      $Hom_{O.C\alpha}$*cat-Set* α(*set* $\{\mathfrak{a}\}$,–),
      cf-id (*cat-Set* α),
      cat-Set α,
      cat-Set α
    ]$_\circ$

Components.

lemma *ntcf-pointed-components*:
  shows *ntcf-pointed* α $\mathfrak{a}(\!|NTMap|\!)$ =
      (
        $\lambda x \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$.
          [
            $(\lambda f \in_\circ$ *Hom* (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) x. $f(\!|ArrVal|\!)(\!|\mathfrak{a}|\!))$,
            Hom (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) x,
            x
          ]$_\circ$
)
    and [*cat-cs-simps*]: *ntcf-pointed* α $\mathfrak{a}(\!|NTDom|\!)$ = $Hom_{O.C\alpha}$*cat-Set* α(*set* $\{\mathfrak{a}\}$,–)
    and [*cat-cs-simps*]: *ntcf-pointed* α $\mathfrak{a}(\!|NTCod|\!)$ = *cf-id* (*cat-Set* α)
    and [*cat-cs-simps*]: *ntcf-pointed* α $\mathfrak{a}(\!|NTDGDom|\!)$ = *cat-Set* α
    and [*cat-cs-simps*]: *ntcf-pointed* α $\mathfrak{a}(\!|NTDGCod|\!)$ = *cat-Set* α
  ⟨*proof*⟩

8.2.2 Natural transformation map

mk-VLambda *ntcf-pointed-components*(*1*)
  |*vsv ntcf-pointed-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain ntcf-pointed-NTMap-vdomain*[*cat-cs-simps*]|

|*app ntcf-pointed-NTMap-app'*|

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-NTMap-app-ArrVal-app*[*cat-cs-simps*]:
assumes $X \in_\circ$ *cat-Set* α(|*Obj*|) **and** F : *set* $\{\mathfrak{a}\} \mapsto_{cat\text{-}Set\ \alpha} X$
shows *ntcf-pointed* $\alpha\ \mathfrak{a}$(|*NTMap*|)(|X|)(|*ArrVal*|)(|F|) = F(|*ArrVal*|)(|$\mathfrak{a}$|)
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-NTMap-app-is-iso-arr*:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|) **and** $X \in_\circ$ *cat-Set* α(|*Obj*|)
shows *ntcf-pointed* $\alpha\ \mathfrak{a}$(|*NTMap*|)(|X|) :
Hom (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) $X \mapsto_{iso_{cat\text{-}Set\ \alpha}} X$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-NTMap-app-is-iso-arr'*[*cat-cs-intros*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
and $X \in_\circ$ *cat-Set* α(|*Obj*|)
and A' = *Hom* (*cat-Set* α) (*set* $\{\mathfrak{a}\}$) X
and $B' = X$
and $\mathfrak{C}'$ = *cat-Set* α
shows *ntcf-pointed* $\alpha\ \mathfrak{a}$(|*NTMap*|)(|X|) : $A' \mapsto_{iso_{\mathfrak{C}'}} B'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-NTMap-app-is-iso-arr'*

lemmas (in $\mathcal{Z}$**)** *ntcf-pointed-NTMap-app-is-arr'*[*cat-cs-intros*] =
is-iso-arrD(*1*)[*OF* $\mathcal{Z}$.*ntcf-pointed-NTMap-app-is-iso-arr'*]

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-NTMap-app-is-arr'*

8.2.3 Pointed natural transformation is a natural isomorphism

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-is-iso-ntcf*:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
shows *ntcf-pointed* $\alpha\ \mathfrak{a}$:
$Hom_{O.C\alpha}$*cat-Set* α(*set* $\{\mathfrak{a}\}$,–) $\mapsto_{CF.iso}$ *cf-id* (*cat-Set* α) :
cat-Set $\alpha \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-is-iso-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
and $\mathfrak{F}'$ = $Hom_{O.C\alpha}$*cat-Set* α(*set* $\{\mathfrak{a}\}$,–)
and $\mathfrak{G}'$ = *cf-id* (*cat-Set* α)
and $\mathfrak{A}'$ = *cat-Set* α
and $\mathfrak{B}'$ = *cat-Set* α
and $\alpha' = \alpha$
shows *ntcf-pointed* $\alpha\ \mathfrak{a}$: $\mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-is-iso-ntcf'*

8.3 Inverse pointed natural transformation

8.3.1 Definition and elementary properties

See Chapter III-2 in [9].

definition *ntcf-pointed-inv* :: $V \Rightarrow V \Rightarrow V$
where *ntcf-pointed-inv* $\alpha\ \mathfrak{a}$ =
[

(
  $\lambda X \in_\circ$ *cat-Set* α(|*Obj*|).
    [($\lambda x \in_\circ X$. *ntcf-paa* $\mathfrak{a}$ *X x*), *X*, *Hom* (*cat-Set* α) (*set* {$\mathfrak{a}$}) *X*]$_\circ$
),
cf-id (*cat-Set* α),
Hom$_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−),
cat-Set α,
cat-Set α
]$_\circ$

Components.

lemma *ntcf-pointed-inv-components*:
  shows *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTMap*|) =
    (
      $\lambda X \in_\circ$ *cat-Set* α(|*Obj*|).
        [($\lambda x \in_\circ X$. *ntcf-paa* $\mathfrak{a}$ *X x*), *X*, *Hom* (*cat-Set* α) (*set* {$\mathfrak{a}$}) *X*]$_\circ$
)
    and [*cat-cs-simps*]: *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTDom*|) = *cf-id* (*cat-Set* α)
    and [*cat-cs-simps*]:
      ntcf-pointed-inv α $\mathfrak{a}$(|*NTCod*|) = *Hom*$_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−)
    and [*cat-cs-simps*]: *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTDGDom*|) = *cat-Set* α
    and [*cat-cs-simps*]: *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTDGCod*|) = *cat-Set* α
  ⟨*proof*⟩

8.3.2 Natural transformation map

mk-VLambda *ntcf-pointed-inv-components*(*1*)
  |*vsv ntcf-pointed-inv-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain ntcf-pointed-inv-NTMap-vdomain*[*cat-cs-simps*]|
  |*app ntcf-pointed-inv-NTMap-app′*|

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-inv-NTMap-app-ArrVal-app*[*cat-cs-simps*]:
  assumes $X \in_\circ$ *cat-Set* α(|*Obj*|) **and** $x \in_\circ X$
  shows *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTMap*|)(|*X*|)(|*ArrVal*|)(|*x*|) = *ntcf-paa* $\mathfrak{a}$ *X x*
  ⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-inv-NTMap-app-is-arr*:
  assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|) **and** $X \in_\circ$ *cat-Set* α(|*Obj*|)
  shows *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTMap*|)(|*X*|) :
    $X \mapsto_{cat\text{-}Set\ \alpha}$ *Hom* (*cat-Set* α) (*set* {$\mathfrak{a}$}) *X*
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ntcf-pointed-inv-NTMap-app-is-arr′*[*cat-cs-intros*]:
  assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
    and $X \in_\circ$ *cat-Set* α(|*Obj*|)
    and $A' = X$
    and B' = *Hom* (*cat-Set* α) (*set* {$\mathfrak{a}$}) *X*
    and $\mathfrak{C}'$ = *cat-Set* α
  shows *ntcf-pointed-inv* α $\mathfrak{a}$(|*NTMap*|)(|*X*|) : $A' \mapsto_{\mathfrak{C}'} B'$
  ⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-inv-NTMap-app-is-arr′*

lemma (in $\mathcal{Z}$**)** *is-inverse-ntcf-pointed-inv-NTMap-app*:
  assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|) **and** $X \in_\circ$ *cat-Set* α(|*Obj*|)
  shows
    is-inverse
      (*cat-Set* α)

(*ntcf-pointed-inv* α $\mathfrak{a}$(|*NTMap*|)(|*X*|))
(*ntcf-pointed* α $\mathfrak{a}$(|*NTMap*|)(|*X*|))
(is ‹*is-inverse* (*cat-Set* α) *?bwd ?fwd*›)
⟨*proof*⟩

8.3.3 Inverse pointed natural transformation is a natural isomorphism

lemma (in $\mathcal{Z}$) *ntcf-pointed-inv-is-ntcf*:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
shows *ntcf-pointed-inv* α $\mathfrak{a}$:
cf-id (*cat-Set* α) $\mapsto_{CF}$ $Hom_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−) :
cat-Set α $\mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *ntcf-pointed-inv-is-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
and $\mathfrak{F}'$ = *cf-id* (*cat-Set* α)
and $\mathfrak{G}'$ = $Hom_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−)
and $\mathfrak{A}'$ = *cat-Set* α
and $\mathfrak{B}'$ = *cat-Set* α
and α' = α
shows *ntcf-pointed-inv* α $\mathfrak{a}$: $\mathfrak{F}'$ $\mapsto_{CF}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\mapsto_{C_{\alpha'}}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-inv-is-ntcf'*

lemma (in $\mathcal{Z}$) *inv-ntcf-ntcf-pointed*[*cat-cs-simps*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
shows *inv-ntcf* (*ntcf-pointed* α $\mathfrak{a}$) = *ntcf-pointed-inv* α $\mathfrak{a}$
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *inv-ntcf-ntcf-pointed-inv*[*cat-cs-simps*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
shows *inv-ntcf* (*ntcf-pointed-inv* α $\mathfrak{a}$) = *ntcf-pointed* α $\mathfrak{a}$
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *ntcf-pointed-inv-is-iso-ntcf*:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
shows *ntcf-pointed-inv* α $\mathfrak{a}$:
cf-id (*cat-Set* α) $\mapsto_{CF.iso}$ $Hom_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−) :
cat-Set α $\mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *ntcf-pointed-inv-is-iso-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{a} \in_\circ$ *cat-Set* α(|*Obj*|)
and $\mathfrak{F}'$ = *cf-id* (*cat-Set* α)
and $\mathfrak{G}'$ = $Hom_{O.C\alpha}$*cat-Set* α(*set* {$\mathfrak{a}$},−)
and $\mathfrak{A}'$ = *cat-Set* α
and $\mathfrak{B}'$ = *cat-Set* α
and α' = α
shows *ntcf-pointed-inv* α $\mathfrak{a}$: $\mathfrak{F}'$ $\mapsto_{CF.iso}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\mapsto_{C_{\alpha'}}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*ntcf-pointed-inv-is-iso-ntcf'*

9 Representable and corepresentable functors

9.1 Representable and corepresentable functors

9.1.1 Definitions and elementary properties

See Chapter III-2 in [9] or Section 2.1 in [14].

definition *cat-representation* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *cat-representation* α $\mathfrak{F}$ c ψ $\longleftrightarrow$
    $c \in_\circ \mathfrak{F}(|HomDom|)(|Obj|) \wedge$
    $\psi : Hom_{O.C\alpha}\mathfrak{F}(|HomDom|)(c,-) \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{F}(|HomDom|) \mapsto\mapsto_{C\alpha}$ *cat-Set* α

definition *cat-corepresentation* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *cat-corepresentation* α $\mathfrak{F}$ c ψ $\longleftrightarrow$
    $c \in_\circ \mathfrak{F}(|HomDom|)(|Obj|) \wedge$
    $\psi : Hom_{O.C\alpha}$*op-cat* $(\mathfrak{F}(|HomDom|))(-,c) \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{F}(|HomDom|) \mapsto\mapsto_{C\alpha}$ *cat-Set* α

Rules.

context
  fixes α $\mathfrak{C}$ $\mathfrak{F}$
  assumes $\mathfrak{F}$: $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
begin

interpretation $\mathfrak{F}$: *is-functor* α $\mathfrak{C}$ ‹*cat-Set* α› $\mathfrak{F}$ ⟨*proof*⟩

mk-ide rf *cat-representation-def*[**where** α=α **and** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cat-cs-simps*]
  |*intro cat-representationI*|
  |*dest cat-representationD'*|
  |*elim cat-representationE'*|

end

lemmas *cat-representationD*[*dest*] = *cat-representationD'*[*rotated*]
  and *cat-representationE*[*elim*] = *cat-representationE'*[*rotated*]

lemma *cat-corepresentationI*:
  assumes *category* α $\mathfrak{C}$
    and $\mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
    and $c \in_\circ \mathfrak{C}(|Obj|)$
    and $\psi : Hom_{O.C\alpha}\mathfrak{C}(-,c) \mapsto_{CF.iso} \mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  shows *cat-corepresentation* α $\mathfrak{F}$ c ψ
⟨*proof*⟩

lemma *cat-corepresentationD*:
  assumes *cat-corepresentation* α $\mathfrak{F}$ c ψ
    and *category* α $\mathfrak{C}$
    and $\mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  shows $c \in_\circ \mathfrak{C}(|Obj|)$
    and $\psi : Hom_{O.C\alpha}\mathfrak{C}(-,c) \mapsto_{CF.iso} \mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma *cat-corepresentationE*:
  assumes *cat-corepresentation* α $\mathfrak{F}$ c ψ
    and *category* α $\mathfrak{C}$
    and $\mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  obtains $c \in_\circ \mathfrak{C}(|Obj|)$
    and $\psi : Hom_{O.C\alpha}\mathfrak{C}(-,c) \mapsto_{CF.iso} \mathfrak{F}$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  ⟨*proof*⟩

9.1.2 Representable functors and universal arrows

lemma *universal-arrow-of-if-cat-representation*:
— See Proposition 2 in Chapter III-2 in [9].
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathit{cat\text{-}Set}\ \alpha$
and $\mathit{cat\text{-}representation}\ \alpha\ \mathfrak{K}\ r\ \psi$
and $\mathfrak{a} \in_\circ \mathit{cat\text{-}Set}\ \alpha(\!|\mathit{Obj}|\!)$
shows *universal-arrow-of*
$\mathfrak{K}\ (\mathit{set}\ \{\mathfrak{a}\})\ r\ (\mathit{ntcf\text{-}paa}\ \mathfrak{a}\ (\mathfrak{K}(\!|\mathit{ObjMap}|\!)(\!|r|\!))\ (\psi(\!|\mathit{NTMap}|\!)(\!|r|\!)(\!|\mathit{ArrVal}|\!)(\!|\mathfrak{C}(\!|\mathit{CId}|\!)(\!|r|\!)|\!)))$
⟨*proof*⟩

lemma *universal-arrow-of-if-cat-corepresentation*:
— See Proposition 2 in Chapter III-2 in [9].
assumes $\mathit{category}\ \alpha\ \mathfrak{C}$
and $\mathfrak{K} : \mathit{op\text{-}cat}\ \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathit{cat\text{-}Set}\ \alpha$
and $\mathit{cat\text{-}corepresentation}\ \alpha\ \mathfrak{K}\ r\ \psi$
and $\mathfrak{a} \in_\circ \mathit{cat\text{-}Set}\ \alpha(\!|\mathit{Obj}|\!)$
shows *universal-arrow-of*
$\mathfrak{K}\ (\mathit{set}\ \{\mathfrak{a}\})\ r\ (\mathit{ntcf\text{-}paa}\ \mathfrak{a}\ (\mathfrak{K}(\!|\mathit{ObjMap}|\!)(\!|r|\!))\ (\psi(\!|\mathit{NTMap}|\!)(\!|r|\!)(\!|\mathit{ArrVal}|\!)(\!|\mathfrak{C}(\!|\mathit{CId}|\!)(\!|r|\!)|\!)))$
⟨*proof*⟩

lemma *cat-representation-if-universal-arrow-of*:
— See Proposition 2 in Chapter III-2 in [9].
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathit{cat\text{-}Set}\ \alpha$
and $\mathfrak{a} \in_\circ \mathit{cat\text{-}Set}\ \alpha(\!|\mathit{Obj}|\!)$
and $\mathit{universal\text{-}arrow\text{-}of}\ \mathfrak{K}\ (\mathit{set}\ \{\mathfrak{a}\})\ r\ u$
shows $\mathit{cat\text{-}representation}\ \alpha\ \mathfrak{K}\ r\ (\mathit{Yoneda\text{-}arrow}\ \alpha\ \mathfrak{K}\ r\ (u(\!|\mathit{ArrVal}|\!)(\!|\mathfrak{a}|\!)))$
⟨*proof*⟩

lemma *cat-corepresentation-if-universal-arrow-of*:
— See Proposition 2 in Chapter III-2 in [9].
assumes $\mathit{category}\ \alpha\ \mathfrak{C}$
and $\mathfrak{K} : \mathit{op\text{-}cat}\ \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathit{cat\text{-}Set}\ \alpha$
and $\mathfrak{a} \in_\circ \mathit{cat\text{-}Set}\ \alpha(\!|\mathit{Obj}|\!)$
and $\mathit{universal\text{-}arrow\text{-}of}\ \mathfrak{K}\ (\mathit{set}\ \{\mathfrak{a}\})\ r\ u$
shows $\mathit{cat\text{-}corepresentation}\ \alpha\ \mathfrak{K}\ r\ (\mathit{Yoneda\text{-}arrow}\ \alpha\ \mathfrak{K}\ r\ (u(\!|\mathit{ArrVal}|\!)(\!|\mathfrak{a}|\!)))$
⟨*proof*⟩

9.2 Limits and colimits as universal cones

lemma *is-tm-cat-limit-if-cat-corepresentation*:
— See Definition 3.1.5 in Section 3.1 in [14].
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathit{cat\text{-}corepresentation}\ \alpha\ (\mathit{tm\text{-}cf\text{-}Cone}\ \alpha\ \mathfrak{F})\ r\ \psi$
(**is** ‹$\mathit{cat\text{-}corepresentation}\ \alpha\ ?\mathit{Cone}\ r\ \psi$›)
shows $\mathit{ntcf\text{-}of\text{-}ntcf\text{-}arrow}\ \mathfrak{J}\ \mathfrak{C}\ (\psi(\!|\mathit{NTMap}|\!)(\!|r|\!)(\!|\mathit{ArrVal}|\!)(\!|\mathfrak{C}(\!|\mathit{CId}|\!)(\!|r|\!)|\!)) :$
$r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
(**is** ‹$\mathit{ntcf\text{-}of\text{-}ntcf\text{-}arrow}\ \mathfrak{J}\ \mathfrak{C}\ ?\psi r1r : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$›)
⟨*proof*⟩

lemma *cat-corepresentation-if-is-tm-cat-limit*:
— See Definition 3.1.5 in Section 3.1 in [14].
assumes $\psi : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows *cat-corepresentation*
$\alpha\ (\mathit{tm\text{-}cf\text{-}Cone}\ \alpha\ \mathfrak{F})\ r\ (\mathit{Yoneda\text{-}arrow}\ \alpha\ (\mathit{tm\text{-}cf\text{-}Cone}\ \alpha\ \mathfrak{F})\ r\ (\mathit{ntcf\text{-}arrow}\ \psi))$
(**is** ‹$\mathit{cat\text{-}corepresentation}\ \alpha\ ?\mathit{Cone}\ r\ ?Y\psi$›)
⟨*proof*⟩

10 Completeness and cocompleteness

10.1 Limits by products and equalizers

lemma *cat-limit-of-cat-prod-obj-and-cat-equalizer*:
— See Theorem 1 in Chapter V-2 in [9].
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b};\ \mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A.$ *tm-cf-discrete* $\alpha\ (\mathfrak{J}(\!|Obj|\!))\ A\ \mathfrak{C} \Longrightarrow$
$\exists P\ \pi.\ \pi : P <_{CF.\prod} A : \mathfrak{J}(\!|Obj|\!) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A.$ *tm-cf-discrete* $\alpha\ (\mathfrak{J}(\!|Arr|\!))\ A\ \mathfrak{C} \Longrightarrow$
$\exists P\ \pi.\ \pi : P <_{CF.\prod} A : \mathfrak{J}(\!|Arr|\!) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $r\ u$ **where** $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-colimit-of-cat-prod-obj-and-cat-coequalizer*:
— See Theorem 1 in Chapter V-2 in [9].
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a};\ \mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A.$ *tm-cf-discrete* $\alpha\ (\mathfrak{J}(\!|Obj|\!))\ A\ \mathfrak{C} \Longrightarrow$
$\exists P\ \pi.\ \pi : A >_{CF.\coprod} P : \mathfrak{J}(\!|Obj|\!) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A.$ *tm-cf-discrete* $\alpha\ (\mathfrak{J}(\!|Arr|\!))\ A\ \mathfrak{C} \Longrightarrow$
$\exists P\ \pi.\ \pi : A >_{CF.\coprod} P : \mathfrak{J}(\!|Arr|\!) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $r\ u$ **where** $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

10.2 Small-complete and small-cocomplete category

10.2.1 Definition and elementary properties

locale *cat-small-complete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ \mathfrak{C}$ +
assumes *cat-small-complete*:
$\bigwedge \mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \Longrightarrow \exists u\ r.\ u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

locale *cat-small-cocomplete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ \mathfrak{C}$ +
assumes *cat-small-cocomplete*:
$\bigwedge \mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \Longrightarrow \exists u\ r.\ u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

Rules.

mk-ide rf *cat-small-complete-def*[*unfolded cat-small-complete-axioms-def*]
|*intro cat-small-completeI*|
|*dest cat-small-completeD*[*dest*]|
|*elim cat-small-completeE*[*elim*]|

lemma *cat-small-completeE'*[*elim*]:
assumes *cat-small-complete* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
obtains $u\ r$ **where** $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cat-small-cocomplete-def*[*unfolded cat-small-cocomplete-axioms-def*]
|*intro cat-small-cocompleteI*|
|*dest cat-small-cocompleteD*[*dest*]|
|*elim cat-small-cocompleteE*[*elim*]|

lemma *cat-small-cocompleteE'*[*elim*]:
assumes *cat-small-cocomplete* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
obtains $u\ r$ **where** $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

⟨*proof*⟩

10.2.2 Duality

lemma (**in** *cat-small-complete*) *cat-small-cocomplete-op*[*cat-op-intros*]:
cat-small-cocomplete α (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cat-small-complete.cat-small-cocomplete-op*

lemma (**in** *cat-small-cocomplete*) *cat-small-complete-op*[*cat-op-intros*]:
cat-small-complete α (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cat-small-cocomplete.cat-small-complete-op*

10.2.3 A category with equalizers and small products is small-complete

lemma (**in** *category*) *cat-small-complete-if-eq-and-obj-prod*:
— See Corollary 2 in Chapter V-2 in [9]
assumes $\bigwedge\mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b};\ \mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A\ I.$ *tm-cf-discrete* $\alpha\ I\ A\ \mathfrak{C} \Longrightarrow \exists P\ \pi.\ \pi : P <_{CF.\prod} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cat-small-complete* α $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-small-cocomplete-if-eq-and-obj-prod*:
assumes $\bigwedge\mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a};\ \mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A\ I.$ *tm-cf-discrete* $\alpha\ I\ A\ \mathfrak{C} \Longrightarrow \exists P\ \pi.\ \pi : A >_{CF.\coprod} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cat-small-cocomplete* α $\mathfrak{C}$
⟨*proof*⟩

10.2.4 Existence of the initial and terminal objects in small-complete and small-cocomplete categories

lemma (**in** *cat-small-complete*) *cat-sc-ex-obj-initial*:
— See Theorem 1 in Chapter V-6 in [9].
assumes $A \subseteq_\circ \mathfrak{C}(\!|Obj|\!)$
and $A \in_\circ$ *Vset* α
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \exists f\ a.\ a \in_\circ A \wedge f : a \mapsto_{\mathfrak{C}} c$
obtains z **where** *obj-initial* $\mathfrak{C}$ z
⟨*proof*⟩

lemma (**in** *cat-small-cocomplete*) *cat-sc-ex-obj-terminal*:
— See Theorem 1 in Chapter V-6 in [9].
assumes $A \subseteq_\circ \mathfrak{C}(\!|Obj|\!)$
and $A \in_\circ$ *Vset* α
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \exists f\ a.\ a \in_\circ A \wedge f : c \mapsto_{\mathfrak{C}} a$
obtains z **where** *obj-terminal* $\mathfrak{C}$ z
⟨*proof*⟩

10.2.5 Creation of limits, continuity and completeness

lemma
— See Theorem 2 in Chapter V-4 in [9].
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *cat-small-complete* α $\mathfrak{B}$
and $\bigwedge\mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{A} \Longrightarrow \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$

and $\bigwedge\mathfrak{F}\ \mathfrak{J}.\ \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{A} \Longrightarrow$ *cf-creates-limits* $\alpha\ \mathfrak{G}\ \mathfrak{F}$
shows *is-tm-cf-continuous-if-cf-creates-limits*: *is-tm-cf-continuous* $\alpha\ \mathfrak{G}$
and *cat-small-complete-if-cf-creates-limits*: *cat-small-complete* $\alpha\ \mathfrak{A}$
⟨*proof*⟩

10.3 Finite-complete and finite-cocomplete category

locale *cat-finite-complete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ \mathfrak{C}$ +
assumes *cat-finite-complete*:
$\bigwedge\mathfrak{F}\ \mathfrak{J}.$ [[*finite-category* $\alpha\ \mathfrak{J}$; $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$]] $\Longrightarrow$
$\exists\, u\ r.\ u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

locale *cat-finite-cocomplete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ \mathfrak{C}$ +
assumes *cat-finite-cocomplete*:
$\bigwedge\mathfrak{F}\ \mathfrak{J}.$ [[*finite-category* $\alpha\ \mathfrak{J}$; $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$]] $\Longrightarrow$
$\exists\, u\ r.\ u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

Rules.

mk-ide rf *cat-finite-complete-def*[*unfolded cat-finite-complete-axioms-def*]
|*intro cat-finite-completeI*|
|*dest cat-finite-completeD*[*dest*]|
|*elim cat-finite-completeE*[*elim*]|

lemma *cat-finite-completeE'*[*elim*]:
assumes *cat-finite-complete* $\alpha\ \mathfrak{C}$
and *finite-category* $\alpha\ \mathfrak{J}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $u\ r$ **where** $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cat-finite-cocomplete-def*[*unfolded cat-finite-cocomplete-axioms-def*]
|*intro cat-finite-cocompleteI*|
|*dest cat-finite-cocompleteD*[*dest*]|
|*elim cat-finite-cocompleteE*[*elim*]|

lemma *cat-finite-cocompleteE'*[*elim*]:
assumes *cat-finite-cocomplete* $\alpha\ \mathfrak{C}$
and *finite-category* $\alpha\ \mathfrak{J}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $u\ r$ **where** $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

Elementary properties.

sublocale *cat-small-complete* ⊆ *cat-finite-complete*
⟨*proof*⟩

sublocale *cat-small-cocomplete* ⊆ *cat-finite-cocomplete*
⟨*proof*⟩

11 Comma categories and universal constructions

11.1 Relationship between the universal arrows, initial objects and terminal objects

lemma (**in** *is-functor*) *universal-arrow-of-if-obj-initial*:
— See Chapter III-1 in [9].
assumes $c \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** *obj-initial* $(c \downarrow_{CF} \mathfrak{F})$ $[0, r, u]_\circ$
shows *universal-arrow-of* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *obj-initial-if-universal-arrow-of*:
— See Chapter III-1 in [9].
assumes *universal-arrow-of* $\mathfrak{F}$ c r u
shows *obj-initial* $(c \downarrow_{CF} \mathfrak{F})$ $[0, r, u]_\circ$
⟨*proof*⟩

lemma (**in** *is-functor*) *universal-arrow-fo-if-obj-terminal*:
— See Chapter III-1 in [9].
assumes $c \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** *obj-terminal* $(\mathfrak{F} {}_{CF}\!\downarrow c)$ $[r, 0, u]_\circ$
shows *universal-arrow-fo* $\mathfrak{F}$ c r u
⟨*proof*⟩

lemma (**in** *is-functor*) *obj-terminal-if-universal-arrow-fo*:
— See Chapter III-1 in [9].
assumes *universal-arrow-fo* $\mathfrak{F}$ c r u
shows *obj-terminal* $(\mathfrak{F} {}_{CF}\!\downarrow c)$ $[r, 0, u]_\circ$
⟨*proof*⟩

11.2 A projection for a comma category constructed from a functor and an object creates small limits

See Chapter V-6 in [9].

lemma *cf-obj-cf-comma-proj-creates-limits*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{X}$
and *is-tm-cf-continuous* α $\mathfrak{G}$
and $x \in_\circ \mathfrak{X}(\!|Obj|\!)$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} x \downarrow_{CF} \mathfrak{G}$
shows *cf-creates-limits* α $(x \ {}_O\!\sqcap_{CF} \mathfrak{G})$ $\mathfrak{F}$
⟨*proof*⟩

12 Category *Set* and universal constructions

12.1 Discrete functor with tiny maps to the category *Set*

lemma (in $\mathcal{Z}$**)** *tm-cf-discrete-cat-Set-if-VLambda-in-Vset*:
  assumes *VLambda I F* $\in_\circ$ *Vset* α
  shows *tm-cf-discrete* α *I F* (*cat-Set* α)
⟨*proof*⟩

12.2 Product cone and coproduct cocone for the category *Set*

12.2.1 Definition and elementary properties

definition *ntcf-Set-obj-prod* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  where *ntcf-Set-obj-prod* α *I F* = *ntcf-obj-prod-base*
    (*cat-Set* α) *I F* ($\prod_\circ i\in_\circ I.\ F\ i$) ($\lambda i.$ *vprojection-arrow I F i*)

definition *ntcf-Set-obj-coprod* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  where *ntcf-Set-obj-coprod* α *I F* = *ntcf-obj-coprod-base*
    (*cat-Set* α) *I F* ($\coprod_\circ i\in_\circ I.\ F\ i$) ($\lambda i.$ *vcinjection-arrow I F i*)

Components.

lemma *ntcf-Set-obj-prod-components*:
  shows *ntcf-Set-obj-prod* α *I F*(|*NTMap*|) =
    ($\lambda i\in_\circ :_C$ *I*(|*Obj*|). *vprojection-arrow I F i*)
    and *ntcf-Set-obj-prod* α *I F*(|*NTDom*|) =
    cf-const ($:_C$ *I*) (*cat-Set* α) ($\prod_\circ i\in_\circ I.\ F\ i$)
    and *ntcf-Set-obj-prod* α *I F*(|*NTCod*|) = :→: *I F* (*cat-Set* α)
    and *ntcf-Set-obj-prod* α *I F*(|*NTDGDom*|) = $:_C$ *I*
    and *ntcf-Set-obj-prod* α *I F*(|*NTDGCod*|) = *cat-Set* α
  ⟨*proof*⟩

lemma *ntcf-Set-obj-coprod-components*:
  shows *ntcf-Set-obj-coprod* α *I F*(|*NTMap*|) =
      ($\lambda i\in_\circ :_C$ *I*(|*Obj*|). *vcinjection-arrow I F i*)
    and *ntcf-Set-obj-coprod* α *I F*(|*NTDom*|) = :→: *I F* (*cat-Set* α)
    and *ntcf-Set-obj-coprod* α *I F*(|*NTCod*|) =
      cf-const ($:_C$ *I*) (*cat-Set* α) ($\coprod_\circ i\in_\circ I.\ F\ i$)
    and *ntcf-Set-obj-coprod* α *I F*(|*NTDGDom*|) = $:_C$ *I*
    and *ntcf-Set-obj-coprod* α *I F*(|*NTDGCod*|) = *cat-Set* α
  ⟨*proof*⟩

12.2.2 Natural transformation map

mk-VLambda *ntcf-Set-obj-prod-components*(*1*)
  |*vsv ntcf-Set-obj-prod-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain ntcf-Set-obj-prod-NTMap-vdomain*[*cat-cs-simps*]|
  |*app ntcf-Set-obj-prod-NTMap-app*[*cat-cs-simps*]|

mk-VLambda *ntcf-Set-obj-coprod-components*(*1*)
  |*vsv ntcf-Set-obj-coprod-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain ntcf-Set-obj-coprod-NTMap-vdomain*[*cat-cs-simps*]|
  |*app ntcf-Set-obj-coprod-NTMap-app*[*cat-cs-simps*]|

12.2.3 Product cone for the category *Set* is a universal cone and product cocone for the category *Set* is a universal cocone

lemma (in $\mathcal{Z}$**)** *tm-cf-discrete-ntcf-obj-prod-base-is-cat-obj-prod*:
  — See Theorem 5.2 in Chapter Introduction in [6].

assumes *VLambda I F* $\in_\circ$ *Vset* α
shows *ntcf-Set-obj-prod* α *I F* :
$(\prod_\circ i\in_\circ I.\ F\ i) <_{CF.\prod} F : I \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma **(in** $\mathcal{Z}$**)** *tm-cf-discrete-ntcf-obj-prod-base-is-tm-cat-obj-prod*:
— See Theorem 5.2 in Chapter Introduction in [6].
assumes *VLambda I F* $\in_\circ$ *Vset* α
shows *ntcf-Set-obj-prod* α *I F* :
$(\prod_\circ i\in_\circ I.\ F\ i) <_{CF.tm.\prod} F : I \mapsto\mapsto_{C.tm\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma **(in** $\mathcal{Z}$**)** *tm-cf-discrete-ntcf-obj-coprod-base-is-cat-obj-coprod*:
— See Theorem 5.2 in Chapter Introduction in [6].
assumes *VLambda I F* $\in_\circ$ *Vset* α
shows *ntcf-Set-obj-coprod* α *I F* :
$F >_{CF.\coprod} (\coprod_\circ i\in_\circ I.\ F\ i) : I \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma **(in** $\mathcal{Z}$**)** *ntcf-Set-obj-coprod-is-tm-cat-obj-coprod*:
— See Theorem 5.2 in Chapter Introduction in [6].
assumes *VLambda I F* $\in_\circ$ *Vset* α
shows *ntcf-Set-obj-coprod* α *I F* :
$F >_{CF.tm.\coprod} (\coprod_\circ i\in_\circ I.\ F\ i) : I \mapsto\mapsto_{C.tm\alpha}$ *cat-Set* α
⟨*proof*⟩

12.3 Equalizer for the category *Set*

12.3.1 Definition and elementary properties

abbreviation *ntcf-Set-equalizer-map* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-Set-equalizer-map* α *a g f i* ≡
(
$i = \mathfrak{a}_{PL2}$ *?*
incl-Set (*vequalizer a g f*) *a* :
$g \circ_{A\,cat\text{-}Set\ \alpha}$ *incl-Set* (*vequalizer a g f*) *a*
)

definition *ntcf-Set-equalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-Set-equalizer* α *a b g f* = *ntcf-equalizer-base*
(*cat-Set* α) *a b g f* (*vequalizer a g f*) (*ntcf-Set-equalizer-map* α *a g f*)

Components.

context
fixes *a g f* α :: *V*
begin

lemmas *ntcf-Set-equalizer-components* =
ntcf-equalizer-base-components[
where $\mathfrak{C}$=‹*cat-Set* α›
and *e*=‹*ntcf-Set-equalizer-map* α *a g f*›
and *E*=‹*vequalizer a g f*›
and $\mathfrak{a}$=*a* **and** $\mathfrak{g}$=*g* **and** $\mathfrak{f}$=*f*,
folded ntcf-Set-equalizer-def
]

end

12.3.2 Natural transformation map

mk-VLambda *ntcf-Set-equalizer-components*(*1*)
|*vsv ntcf-Set-equalizer-NTMap-vsv*[*cat-Set-cs-intros*]|
|*vdomain ntcf-Set-equalizer-NTMap-vdomain*[*cat-Set-cs-simps*]|
|*app ntcf-Set-equalizer-NTMap-app*|

lemma *ntcf-Set-equalizer-2-NTMap-app-*𝔞[*cat-Set-cs-simps*]:
assumes $x = \mathfrak{a}_{PL2}$
shows
ntcf-Set-equalizer α *a b g f*⦇*NTMap*⦈⦇x⦈ =
incl-Set (*vequalizer a g f*) *a*
⟨*proof*⟩

lemma *ntcf-Set-equalizer-2-NTMap-app-*𝔟[*cat-Set-cs-simps*]:
assumes $x = \mathfrak{b}_{PL2}$
shows
ntcf-Set-equalizer α *a b g f*⦇*NTMap*⦈⦇x⦈ =
$g \circ_{A\,cat\text{-}Set\ \alpha}$ *incl-Set* (*vequalizer a g f*) *a*
⟨*proof*⟩

12.3.3 Equalizer for the category *Set* is an equalizer

lemma (**in** *𝒵*) *ntcf-Set-equalizer-2-is-cat-equalizer-2*:
assumes $\mathfrak{g} : \mathfrak{a} \mapsto_{cat\text{-}Set\ \alpha} \mathfrak{b}$ **and** $\mathfrak{f} : \mathfrak{a} \mapsto_{cat\text{-}Set\ \alpha} \mathfrak{b}$
shows *ntcf-Set-equalizer* α 𝔞 𝔟 𝔤 𝔣 :
vequalizer 𝔞 𝔤 𝔣 $<_{CF.eq}$ (𝔞,𝔟,𝔤,𝔣) : $\uparrow\uparrow_C \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

12.4 The category *Set* is small-complete

lemma (**in** *𝒵*) *cat-small-complete-cat-Set*: *cat-small-complete* α (*cat-Set* α)
— This lemma appears as a remark on page 113 in [9].
⟨*proof*⟩

13 Adjoints

13.1 Background

named-theorems *adj-cs-simps*
named-theorems *adj-cs-intros*
named-theorems *adj-field-simps*

definition *AdjLeft* :: *V* **where** [*adj-field-simps*]: *AdjLeft* = *0*
definition *AdjRight* :: *V* **where** [*adj-field-simps*]: *AdjRight* = $1_{\mathbb{N}}$
definition *AdjNT* :: *V* **where** [*adj-field-simps*]: *AdjNT* = $2_{\mathbb{N}}$

13.2 Definition and elementary properties

See subsection 2.1 in [4] or Chapter IV-1 in [9].

locale *is-cf-adjunction* =
  $\mathcal{Z}$ α +
  vfsequence Φ +
  L: *category* α $\mathfrak{C}$ +
  R: *category* α $\mathfrak{D}$ +
  LR: *is-functor* α $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ +
  RL: *is-functor* α $\mathfrak{D}$ $\mathfrak{C}$ $\mathfrak{G}$ +
  NT: *is-iso-ntcf*
    α
    ‹*op-cat* $\mathfrak{C}$ $\times_C$ $\mathfrak{D}$›
    ‹*cat-Set* α›
    ‹$Hom_{O.C\alpha}\mathfrak{D}(\mathfrak{F}-,-)$›
    ‹$Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$›
    ‹$\Phi(|AdjNT|)$›
    for α $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ +
  assumes *cf-adj-length*[*adj-cs-simps*]: *vcard* Φ = $3_{\mathbb{N}}$
    and *cf-adj-AdjLeft*[*adj-cs-simps*]: $\Phi(|AdjLeft|)$ = $\mathfrak{F}$
    and *cf-adj-AdjRight*[*adj-cs-simps*]: $\Phi(|AdjRight|)$ = $\mathfrak{G}$

syntax *-is-cf-adjunction* :: *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *bool*
  (‹(- : - $\rightleftharpoons_{CF}$ - : - $\rightleftharpoons\rightleftharpoons_{C^1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cf-adjunction* ⇌ *is-cf-adjunction*
translations Φ : $\mathfrak{F}$ $\rightleftharpoons_{CF}$ $\mathfrak{G}$: $\mathfrak{C}$ $\rightleftharpoons\rightleftharpoons_{C\alpha}$ $\mathfrak{D}$ ⇌
  CONST is-cf-adjunction α $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ

lemmas [*adj-cs-simps*] =
  is-cf-adjunction.cf-adj-length
  is-cf-adjunction.cf-adj-AdjLeft
  is-cf-adjunction.cf-adj-AdjRight

Components.

lemma *cf-adjunction-components*[*adj-cs-simps*]:
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(|AdjLeft|) = \mathfrak{F}$
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(|AdjRight|) = \mathfrak{G}$
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(|AdjNT|) = \varphi$
  ⟨*proof*⟩

Rules.

lemma (**in** *is-cf-adjunction*) *is-cf-adjunction-axioms'*[*adj-cs-intros*]:
  assumes α' = α **and** $\mathfrak{C}'$ = $\mathfrak{C}$ **and** $\mathfrak{D}'$ = $\mathfrak{D}$ **and** $\mathfrak{F}'$ = $\mathfrak{F}$ **and** $\mathfrak{G}'$ = $\mathfrak{G}$
  shows Φ : $\mathfrak{F}'$ $\rightleftharpoons_{CF}$ $\mathfrak{G}'$: $\mathfrak{C}'$ $\rightleftharpoons\rightleftharpoons_{C_{\alpha'}}$ $\mathfrak{D}'$
  ⟨*proof*⟩

lemmas (**in** *is-cf-adjunction*) [*adj-cs-intros*] = *is-cf-adjunction-axioms*

mk-ide rf *is-cf-adjunction-def*[*unfolded is-cf-adjunction-axioms-def*]
  |*intro is-cf-adjunctionI*|
  |*dest is-cf-adjunctionD*[*dest*]|
  |*elim is-cf-adjunctionE*[*elim*]|

lemmas [*adj-cs-intros*] = *is-cf-adjunctionD*(*3–6*)

lemma (**in** *is-cf-adjunction*) *cf-adj-is-iso-ntcf'*:
  assumes $\mathfrak{F}' = Hom_{O.C\alpha}\mathfrak{D}(\mathfrak{F}-,-)$
    and $\mathfrak{G}' = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$
    and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{D}$
    and $\mathfrak{B}' = cat\text{-}Set\ \alpha$
  shows $\Phi(|AdjNT|) : \mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*adj-cs-intros*] = *is-cf-adjunction.cf-adj-is-iso-ntcf'*

lemma *cf-adj-eqI*:
  assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
    and $\Phi' : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G}' : \mathfrak{C}' \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}'$
    and $\mathfrak{C} = \mathfrak{C}'$
    and $\mathfrak{D} = \mathfrak{D}'$
    and $\mathfrak{F} = \mathfrak{F}'$
    and $\mathfrak{G} = \mathfrak{G}'$
    and $\Phi(|AdjNT|) = \Phi'(|AdjNT|)$
  shows $\Phi = \Phi'$
⟨*proof*⟩

13.3 Opposite adjunction

13.3.1 Definition and elementary properties

See [7] for further information.

abbreviation *op-cf-adj-nt* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *op-cf-adj-nt* $\mathfrak{C}\ \mathfrak{D}\ \varphi \equiv$ *inv-ntcf* (*bnt-flip* (*op-cat* $\mathfrak{C}$) $\mathfrak{D}\ \varphi$)

definition *op-cf-adj* :: $V \Rightarrow V$
  where *op-cf-adj* Φ =
    [
      op-cf ($\Phi(|AdjRight|)$),
      op-cf ($\Phi(|AdjLeft|)$),
      op-cf-adj-nt ($\Phi(|AdjLeft|)(|HomDom|)$) ($\Phi(|AdjLeft|)(|HomCod|)$) ($\Phi(|AdjNT|)$)
    ]∘

lemma *op-cf-adj-components*:
  shows *op-cf-adj* $\Phi(|AdjLeft|)$ = *op-cf* ($\Phi(|AdjRight|)$)
    and *op-cf-adj* $\Phi(|AdjRight|)$ = *op-cf* ($\Phi(|AdjLeft|)$)
    and *op-cf-adj* $\Phi(|AdjNT|)$ =
      op-cf-adj-nt ($\Phi(|AdjLeft|)(|HomDom|)$) ($\Phi(|AdjLeft|)(|HomCod|)$) ($\Phi(|AdjNT|)$)
  ⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *op-cf-adj-components*:
  shows *op-cf-adj* $\Phi(|AdjLeft|)$ = *op-cf* $\mathfrak{G}$
    and *op-cf-adj* $\Phi(|AdjRight|)$ = *op-cf* $\mathfrak{F}$
    and *op-cf-adj* $\Phi(|AdjNT|)$ = *inv-ntcf* (*bnt-flip* (*op-cat* $\mathfrak{C}$) $\mathfrak{D}$ ($\Phi(|AdjNT|)$))

⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.op-cf-adj-components*

The opposite adjunction is an adjunction.

lemma (**in** *is-cf-adjunction*) *is-cf-adjunction-op*:
— See comments in subsection 2.1 in [4].
op-cf-adj Φ : *op-cf* $\mathfrak{G} \rightleftharpoons_{CF}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{D} \rightleftharpoons\rightleftharpoons_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemmas *is-cf-adjunction-op* =
is-cf-adjunction.is-cf-adjunction-op

lemma (**in** *is-cf-adjunction*) *is-cf-adjunction-op'*[*cat-op-intros*]:
assumes $\mathfrak{G}'$ = *op-cf* $\mathfrak{G}$
and $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$
and $\mathfrak{D}'$ = *op-cat* $\mathfrak{D}$
and $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-cf-adj* Φ : $\mathfrak{G}' \rightleftharpoons_{CF} \mathfrak{F}'$: $\mathfrak{D}' \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cf-adjunction.is-cf-adjunction-op'*

The operation of taking the opposite adjunction is an involution.

lemma (**in** *is-cf-adjunction*) *cf-adjunction-op-cf-adj-op-cf-adj*[*cat-op-simps*]:
op-cf-adj (*op-cf-adj* Φ) = Φ
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.cf-adjunction-op-cf-adj-op-cf-adj*

13.3.2 Alternative form of the naturality condition

The lemmas in this subsection are based on the comments on page 81 in [9].

lemma (**in** *is-cf-adjunction*) *cf-adj-Comp-commute-RL*:
assumes $x \in_\circ \mathfrak{C}(|Obj|)$
and $f : \mathfrak{F}(|ObjMap|)(|x|) \mapsto_{\mathfrak{D}} a$
and $k : a \mapsto_{\mathfrak{D}} a'$
shows
$\mathfrak{G}(|ArrMap|)(|k|) \circ_{A\mathfrak{C}} (\Phi(|AdjNT|)(|NTMap|)(|x, a|)_\bullet)(|ArrVal|)(|f|) =$
$(\Phi(|AdjNT|)(|NTMap|)(|x, a'|)_\bullet)(|ArrVal|)(|k \circ_{A\mathfrak{D}} f|)$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adj-Comp-commute-LR*:
assumes $x \in_\circ \mathfrak{C}(|Obj|)$
and $f : \mathfrak{F}(|ObjMap|)(|x|) \mapsto_{\mathfrak{D}} a$
and $h : x' \mapsto_{\mathfrak{C}} x$
shows
$(\Phi(|AdjNT|)(|NTMap|)(|x, a|)_\bullet)(|ArrVal|)(|f|) \circ_{A\mathfrak{C}} h =$
$(\Phi(|AdjNT|)(|NTMap|)(|x', a|)_\bullet)(|ArrVal|)(|f \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|h|)|)$
⟨*proof*⟩

13.4 Unit

13.4.1 Definition and elementary properties

See Chapter IV-1 in [9].

definition *cf-adjunction-unit* :: $V \Rightarrow V$ (‹η_C›)

where η_C Φ =
  [
    (
      $\lambda x \in_\circ \Phi(|AdjLeft|)(|HomDom|)(|Obj|)$.
        $(\Phi(|AdjNT|)(|NTMap|)(|x, \Phi(|AdjLeft|)(|ObjMap|)(|x|)|)_\bullet)(|ArrVal|)($
          $\Phi(|AdjLeft|)(|HomCod|)(|CId|)(|\Phi(|AdjLeft|)(|ObjMap|)(|x|)|)$
        $|)$
    ),
    cf-id $(\Phi(|AdjLeft|)(|HomDom|))$,
    $(\Phi(|AdjRight|)) \circ_{CF} (\Phi(|AdjLeft|))$,
    $\Phi(|AdjLeft|)(|HomDom|)$,
    $\Phi(|AdjLeft|)(|HomDom|)$
  $]_\circ$

Components.

lemma *cf-adjunction-unit-components*:
  shows η_C $\Phi(|NTMap|)$ =
    (
      $\lambda x \in_\circ \Phi(|AdjLeft|)(|HomDom|)(|Obj|)$.
        $(\Phi(|AdjNT|)(|NTMap|)(|x, \Phi(|AdjLeft|)(|ObjMap|)(|x|)|)_\bullet)(|ArrVal|)($
          $\Phi(|AdjLeft|)(|HomCod|)(|CId|)(|\Phi(|AdjLeft|)(|ObjMap|)(|x|)|)$
        $|)$
)
    and η_C $\Phi(|NTDom|)$ = *cf-id* $(\Phi(|AdjLeft|)(|HomDom|))$
    and η_C $\Phi(|NTCod|)$ = $(\Phi(|AdjRight|)) \circ_{CF} (\Phi(|AdjLeft|))$
    and η_C $\Phi(|NTDGDom|)$ = $\Phi(|AdjLeft|)(|HomDom|)$
    and η_C $\Phi(|NTDGCod|)$ = $\Phi(|AdjLeft|)(|HomDom|)$
  ⟨*proof*⟩

context *is-cf-adjunction*
begin

lemma *cf-adjunction-unit-components'*:
  shows η_C $\Phi(|NTMap|)$ =
    $(\lambda x \in_\circ \mathfrak{C}(|Obj|). (\Phi(|AdjNT|)(|NTMap|)(|x, \mathfrak{F}(|ObjMap|)(|x|)|)_\bullet)(|ArrVal|)(|\mathfrak{D}(|CId|)(|\mathfrak{F}(|ObjMap|)(|x|)|)|))$
    and η_C $\Phi(|NTDom|)$ = *cf-id* $\mathfrak{C}$
    and η_C $\Phi(|NTCod|)$ = $\mathfrak{G} \circ_{CF} \mathfrak{F}$
    and η_C $\Phi(|NTDGDom|)$ = $\mathfrak{C}$
    and η_C $\Phi(|NTDGCod|)$ = $\mathfrak{C}$
  ⟨*proof*⟩

mk-VLambda *cf-adjunction-unit-components'*(*1*)
  |*vdomain cf-adjunction-unit-NTMap-vdomain*[*adj-cs-simps*]|
  |*app cf-adjunction-unit-NTMap-app*[*adj-cs-simps*]|

end

mk-VLambda *cf-adjunction-unit-components*(*1*)
  |*vsv cf-adjunction-unit-NTMap-vsv*[*adj-cs-intros*]|

lemmas [*adj-cs-simps*] =
  is-cf-adjunction.cf-adjunction-unit-NTMap-vdomain
  is-cf-adjunction.cf-adjunction-unit-NTMap-app

13.4.2 Natural transformation map

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-NTMap-is-arr*:
  assumes $x \in_\circ \mathfrak{C}(|Obj|)$

shows $\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!) : x \mapsto_{\mathfrak{C}} \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)|\!)$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-NTMap-is-arr′*:
assumes $x \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $a = x$
and $b = \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)|\!)$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!) : x \mapsto_{\mathfrak{C}'} b$
⟨*proof*⟩

lemmas [*adj-cs-intros*] = *is-cf-adjunction.cf-adjunction-unit-NTMap-is-arr′*

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-NTMap-vrange*:
$\mathcal{R}_\circ\ (\eta_C\ \Phi(\!|NTMap|\!)) \subseteq_\circ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

13.4.3 Unit is a natural transformation

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-is-ntcf*:
$\eta_C\ \Phi : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-is-ntcf′*:
assumes $\mathfrak{S} = \textit{cf-id}\ \mathfrak{C}$
and $\mathfrak{S}' = \mathfrak{G} \circ_{CF} \mathfrak{F}$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{C}$
shows $\eta_C\ \Phi : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*adj-cs-intros*] = *is-cf-adjunction.cf-adjunction-unit-is-ntcf′*

13.4.4 Every component of a unit is a universal arrow

The lemmas in this subsection are based on elements of the statement of Theorem 1 in Chapter IV-1 in [9].

lemma (**in** *is-cf-adjunction*) *cf-adj-umap-of-unit*:
assumes $x \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $a \in_\circ \mathfrak{D}(\!|Obj|\!)$
shows $\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x, a|\!)_\bullet = \textit{umap-of}\ \mathfrak{G}\ x\ (\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))\ (\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!))\ a$
(**is** ‹$\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x, a|\!)_\bullet = \textit{?uof-a}$›)
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adj-umap-of-unit′*:
assumes $x \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $a \in_\circ \mathfrak{D}(\!|Obj|\!)$
and $\eta = \eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!)$
and $\mathfrak{F}x = \mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)$
shows $\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x, a|\!)_\bullet = \textit{umap-of}\ \mathfrak{G}\ x\ \mathfrak{F}x\ \eta\ a$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-component-is-ua-of*:
assumes $x \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $\textit{universal-arrow-of}\ \mathfrak{G}\ x\ (\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))\ (\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!))$
(**is** ‹$\textit{universal-arrow-of}\ \mathfrak{G}\ x\ (\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))\ \textit{?}\eta x$›)
⟨*proof*⟩

13.5 Counit

13.5.1 Definition and elementary properties

definition *cf-adjunction-counit* :: $V \Rightarrow V$ (‹ε_C›)
  where ε_C Φ =
    [
      (
        λx∈∘Φ⦇*AdjLeft*⦈⦇*HomCod*⦈⦇*Obj*⦈.
          (Φ⦇*AdjNT*⦈⦇*NTMap*⦈⦇Φ⦇*AdjRight*⦈⦇*ObjMap*⦈⦇x⦈, x⦈∙)$^{-1}$ $_{Set}$⦇*ArrVal*⦈⦇
            Φ⦇*AdjLeft*⦈⦇*HomDom*⦈⦇*CId*⦈⦇Φ⦇*AdjRight*⦈⦇*ObjMap*⦈⦇x⦈⦈
            ⦈
      ),
      (Φ⦇*AdjLeft*⦈) $\circ_{CF}$ (Φ⦇*AdjRight*⦈),
      cf-id (Φ⦇*AdjLeft*⦈⦇*HomCod*⦈),
      Φ⦇*AdjLeft*⦈⦇*HomCod*⦈,
      Φ⦇*AdjLeft*⦈⦇*HomCod*⦈
    ]∘

Components.

lemma *cf-adjunction-counit-components*:
  shows ε_C Φ⦇*NTMap*⦈ =
    (
      λx∈∘Φ⦇*AdjLeft*⦈⦇*HomCod*⦈⦇*Obj*⦈.
        (Φ⦇*AdjNT*⦈⦇*NTMap*⦈⦇Φ⦇*AdjRight*⦈⦇*ObjMap*⦈⦇x⦈, x⦈∙)$^{-1}$ $_{Set}$⦇*ArrVal*⦈⦇
          Φ⦇*AdjLeft*⦈⦇*HomDom*⦈⦇*CId*⦈⦇Φ⦇*AdjRight*⦈⦇*ObjMap*⦈⦇x⦈⦈
          ⦈
)
    and ε_C Φ⦇*NTDom*⦈ = (Φ⦇*AdjLeft*⦈) $\circ_{CF}$ (Φ⦇*AdjRight*⦈)
    and ε_C Φ⦇*NTCod*⦈ = *cf-id* (Φ⦇*AdjLeft*⦈⦇*HomCod*⦈)
    and ε_C Φ⦇*NTDGDom*⦈ = Φ⦇*AdjLeft*⦈⦇*HomCod*⦈
    and ε_C Φ⦇*NTDGCod*⦈ = Φ⦇*AdjLeft*⦈⦇*HomCod*⦈
  ⟨*proof*⟩

context *is-cf-adjunction*
begin

lemma *cf-adjunction-counit-components'*:
  shows ε_C Φ⦇*NTMap*⦈ =
    (
      λx∈∘𝔇⦇*Obj*⦈.
        (Φ⦇*AdjNT*⦈⦇*NTMap*⦈⦇𝔊⦇*ObjMap*⦈⦇x⦈, x⦈∙)$^{-1}$ $_{Set}$⦇*ArrVal*⦈⦇ℭ⦇*CId*⦈⦇𝔊⦇*ObjMap*⦈⦇x⦈⦈⦈
)
    and ε_C Φ⦇*NTDom*⦈ = 𝔉 $\circ_{CF}$ 𝔊
    and ε_C Φ⦇*NTCod*⦈ = *cf-id* 𝔇
    and ε_C Φ⦇*NTDGDom*⦈ = 𝔇
    and ε_C Φ⦇*NTDGCod*⦈ = 𝔇
  ⟨*proof*⟩

mk-VLambda *cf-adjunction-counit-components'*(*1*)
  |*vdomain cf-adjunction-counit-NTMap-vdomain*[*adj-cs-simps*]|
  |*app cf-adjunction-counit-NTMap-app*[*adj-cs-simps*]|

end

mk-VLambda *cf-adjunction-counit-components*(*1*)
  |*vsv cf-adjunction-counit-NTMap-vsv*[*adj-cs-intros*]|

lemmas [*adj-cs-simps*] =

is-cf-adjunction.cf-adjunction-counit-NTMap-vdomain
is-cf-adjunction.cf-adjunction-counit-NTMap-app

13.5.2 Duality for the unit and counit

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-NTMap-op*:
η_C (*op-cf-adj* Φ)(|*NTMap*|) = ε_C Φ(|*NTMap*|)
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.cf-adjunction-unit-NTMap-op*

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-NTMap-op*:
ε_C (*op-cf-adj* Φ)(|*NTMap*|) = η_C Φ(|*NTMap*|)
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.cf-adjunction-counit-NTMap-op*

lemma (**in** *is-cf-adjunction*) *op-ntcf-cf-adjunction-counit*:
op-ntcf (ε_C Φ) = η_C (*op-cf-adj* Φ)
(**is** ‹?ε = ?η›)
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.op-ntcf-cf-adjunction-counit*

lemma (**in** *is-cf-adjunction*) *op-ntcf-cf-adjunction-unit*:
op-ntcf (η_C Φ) = ε_C (*op-cf-adj* Φ)
(**is** ‹?η = ?ε›)
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-cf-adjunction.op-ntcf-cf-adjunction-unit*

13.5.3 Natural transformation map

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-NTMap-is-arr*:
assumes $x \in_\circ \mathfrak{D}$(|*Obj*|)
shows ε_C Φ(|*NTMap*|)(|x|) : $\mathfrak{F}$(|*ObjMap*|)(|$\mathfrak{G}$(|*ObjMap*|)(|x|)|) $\mapsto_{\mathfrak{D}}$ x
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-NTMap-is-arr'*:
assumes $x \in_\circ \mathfrak{D}$(|*Obj*|)
and a = $\mathfrak{F}$(|*ObjMap*|)(|$\mathfrak{G}$(|*ObjMap*|)(|x|)|)
and $b = x$
and $\mathfrak{D}' = \mathfrak{D}$
shows ε_C Φ(|*NTMap*|)(|x|) : $a \mapsto_{\mathfrak{D}'} b$
⟨*proof*⟩

lemmas [*adj-cs-intros*] = *is-cf-adjunction.cf-adjunction-counit-NTMap-is-arr'*

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-NTMap-vrange*:
$\mathcal{R}_\circ$ (ε_C Φ(|*NTMap*|)) $\subseteq_\circ$ $\mathfrak{D}$(|*Arr*|)
⟨*proof*⟩

13.5.4 Counit is a natural transformation

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-is-ntcf*:
ε_C Φ : $\mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF}$ *cf-id* $\mathfrak{D}$: $\mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-is-ntcf'*:
assumes $\mathfrak{S} = \mathfrak{F} \circ_{CF} \mathfrak{G}$
and $\mathfrak{S}' = \textit{cf-id}\ \mathfrak{D}$
and $\mathfrak{A} = \mathfrak{D}$
and $\mathfrak{B} = \mathfrak{D}$
shows $\varepsilon_C\ \Phi : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*adj-cs-intros*] = *is-cf-adjunction.cf-adjunction-counit-is-ntcf'*

13.5.5 Every component of a counit is a universal arrow

The lemmas in this subsection are based on elements of the statement of Theorem 1 in Chapter IV-1 in [9].

lemma (**in** *is-cf-adjunction*) *cf-adj-umap-fo-counit*:
assumes $x \in_\circ \mathfrak{D}(|Obj|)$ **and** $a \in_\circ \mathfrak{C}(|Obj|)$
shows $\textit{op-cf-adj}\ \Phi(|AdjNT|)(|NTMap|)(|x,\ a|)_\bullet =$
$\textit{umap-fo}\ \mathfrak{F}\ x\ (\mathfrak{G}(|ObjMap|)(|x|))\ (\varepsilon_C\ \Phi(|NTMap|)(|x|))\ a$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-component-is-ua-fo*:
assumes $x \in_\circ \mathfrak{D}(|Obj|)$
shows $\textit{universal-arrow-fo}\ \mathfrak{F}\ x\ (\mathfrak{G}(|ObjMap|)(|x|))\ (\varepsilon_C\ \Phi(|NTMap|)(|x|))$
⟨*proof*⟩

13.5.6 Further properties

lemma (**in** *is-cf-adjunction*) *cf-adj-AdjNT-cf-adjunction-unit*:
— See Chapter IV-1 in [9].
assumes $x \in_\circ \mathfrak{C}(|Obj|)$ **and** $f : \mathfrak{F}(|ObjMap|)(|x|) \mapsto_{\mathfrak{D}} a$
shows
$\mathfrak{G}(|ArrMap|)(|f|) \circ_{A\mathfrak{C}} \eta_C\ \Phi(|NTMap|)(|x|) =$
$(\Phi(|AdjNT|)(|NTMap|)(|x,\ a|)_\bullet)(|ArrVal|)(|f|)$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adj-AdjNT-cf-adjunction-counit*:
— See Chapter IV-1 in [9].
assumes $x \in_\circ \mathfrak{D}(|Obj|)$ **and** $g : a \mapsto_{\mathfrak{C}} \mathfrak{G}(|ObjMap|)(|x|)$
shows
$\varepsilon_C\ \Phi(|NTMap|)(|x|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|g|) =$
$(\Phi(|AdjNT|)(|NTMap|)(|a,\ x|)_\bullet)^{-1}{}_{C\,\textit{cat-Set}\ \alpha}(|ArrVal|)(|g|)$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adj-counit-unit-app*[*adj-cs-simps*]:
— See Chapter IV-1 in [9].
assumes $x \in_\circ \mathfrak{D}(|Obj|)$ **and** $g : a \mapsto_{\mathfrak{C}} \mathfrak{G}(|ObjMap|)(|x|)$
shows $\mathfrak{G}(|ArrMap|)(|\varepsilon_C\ \Phi(|NTMap|)(|x|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|g|)|) \circ_{A\mathfrak{C}} \eta_C\ \Phi(|NTMap|)(|a|) = g$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-cf-adjunction.cf-adj-counit-unit-app*

lemma (**in** *is-cf-adjunction*) *cf-adj-unit-counit-app*[*adj-cs-simps*]:
— See Chapter IV-1 in [9].
assumes $x \in_\circ \mathfrak{C}(|Obj|)$ **and** $f : \mathfrak{F}(|ObjMap|)(|x|) \mapsto_{\mathfrak{D}} a$
shows $\varepsilon_C\ \Phi(|NTMap|)(|a|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|\mathfrak{G}(|ArrMap|)(|f|) \circ_{A\mathfrak{C}} \eta_C\ \Phi(|NTMap|)(|x|)|) = f$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-cf-adjunction.cf-adj-unit-counit-app*

13.6 Counit-unit equations

The following equations appear as part of the statement of Theorem 1 in Chapter IV-1 in [9]. These equations also appear in [2], where they are named *counit−unit equations*.

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-unit*:
$(\mathfrak{G} \circ_{CF-NTCF} \varepsilon_C \Phi) \bullet_{NTCF} (\eta_C \Phi \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$
(**is** $\langle(\mathfrak{G} \circ_{CF-NTCF} ?\varepsilon) \bullet_{NTCF} (?\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}\rangle$)
⟨*proof*⟩

lemmas [*adj-cs-simps*] = *is-cf-adjunction.cf-adjunction-counit-unit*

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-counit*:
$(\varepsilon_C \Phi \circ_{NTCF-CF} \mathfrak{F}) \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta_C \Phi) = \textit{ntcf-id}\ \mathfrak{F}$
(**is** $\langle(?\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} ?\eta) = \textit{ntcf-id}\ \mathfrak{F}\rangle$)
⟨*proof*⟩

lemmas [*adj-cs-simps*] = *is-cf-adjunction.cf-adjunction-unit-counit*

13.7 Construction of an adjunction from universal morphisms from objects to functors

The subsection presents the construction of an adjunction given a structured collection of universal morphisms from objects to functors. The content of this subsection follows the statement and the proof of Theorem 2-i in Chapter IV-1 in [9].

13.7.1 The natural transformation associated with the adjunction constructed from universal morphisms from objects to functors

definition *cf-adjunction-AdjNT-of-unit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta$ =
[
$(\lambda cd \in_\circ (\textit{op-cat}\ (\mathfrak{F}(|HomDom|)) \times_C \mathfrak{F}(|HomCod|))(|Obj|).$
$\textit{umap-of}\ \mathfrak{G}\ (cd(|0|))\ (\mathfrak{F}(|ObjMap|)(|cd(|0|)|))\ (\eta(|NTMap|)(|cd(|0|)|))\ (cd(|1_{\mathbb{N}}|))),$
$Hom_{O.C\alpha}\mathfrak{F}(|HomCod|)(\mathfrak{F}-,-),$
$Hom_{O.C\alpha}\mathfrak{F}(|HomDom|)(-,\mathfrak{G}-),$
$\textit{op-cat}\ (\mathfrak{F}(|HomDom|)) \times_C (\mathfrak{F}(|HomCod|)),$
$\textit{cat-Set}\ \alpha$
$]_\circ$

Components.

lemma *cf-adjunction-AdjNT-of-unit-components*:
shows *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(|NTMap|)$ =
(
$\lambda cd \in_\circ (\textit{op-cat}\ (\mathfrak{F}(|HomDom|)) \times_C \mathfrak{F}(|HomCod|))(|Obj|).$
$\textit{umap-of}\ \mathfrak{G}\ (cd(|0|))\ (\mathfrak{F}(|ObjMap|)(|cd(|0|)|))\ (\eta(|NTMap|)(|cd(|0|)|))\ \ (cd(|1_{\mathbb{N}}|))$
)
and *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(|NTDom|) = Hom_{O.C\alpha}\mathfrak{F}(|HomCod|)(\mathfrak{F}-,-)$
and *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(|NTCod|) = Hom_{O.C\alpha}\mathfrak{F}(|HomDom|)(-,\mathfrak{G}-)$
and *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(|NTDGDom|)$ =
$\textit{op-cat}\ (\mathfrak{F}(|HomDom|)) \times_C (\mathfrak{F}(|HomCod|))$
and *cf-adjunction-AdjNT-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(|NTDGCod|) = \textit{cat-Set}\ \alpha$
⟨*proof*⟩

13.7.2 Natural transformation map

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vsv*[*adj-cs-intros*]:

vsv (*cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|NTMap|)$)
⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vdomain*[*adj-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{D}_\circ$ (*cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|NTMap|)$) = (*op-cat* $\mathfrak{C} \times_C \mathfrak{D}$)$(|Obj|)$
⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-NTMap-app*[*adj-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $c \in_\circ \mathfrak{C}(|Obj|)$ **and** $d \in_\circ \mathfrak{D}(|Obj|)$
shows
cf-adjunction-AdjNT-of-unit α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|NTMap|)(|c, d|)_\bullet$ =
umap-of $\mathfrak{G}$ c $(\mathfrak{F}(|ObjMap|)(|c|))$ $(\eta(|NTMap|)(|c|))$ d
⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vrange*:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and η : *cf-id* $\mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ$ (*cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|NTMap|)$) $\subseteq_\circ$ *cat-Set* $\alpha(|Arr|)$
⟨*proof*⟩

13.7.3 Adjunction constructed from universal morphisms from objects to functors is an adjunction

lemma *cf-adjunction-AdjNT-of-unit-is-ntcf*:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and η : *cf-id* $\mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ η :
$Hom_{O.C\alpha}\mathfrak{D}(\mathfrak{F}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$:
op-cat $\mathfrak{C} \times_C \mathfrak{D} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-is-ntcf'*[*adj-cs-intros*]:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and η : *cf-id* $\mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} = Hom_{O.C\alpha}\mathfrak{D}(\mathfrak{F}-,-)$
and $\mathfrak{S}' = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$
and $\mathfrak{A}$ = *op-cat* $\mathfrak{C} \times_C \mathfrak{D}$
and $\mathfrak{B}$ = *cat-Set* α
shows *cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ η : $\mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

13.7.4 Adjunction constructed from universal morphisms from objects to functors

definition *cf-adjunction-of-unit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ η =
$[\mathfrak{F}, \mathfrak{G},$ *cf-adjunction-AdjNT-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta]_\circ$

Components.

lemma *cf-adjunction-of-unit-components*:
shows [*adj-cs-simps*]: *cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|AdjLeft|)$ = $\mathfrak{F}$
and [*adj-cs-simps*]: *cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|AdjRight|)$ = $\mathfrak{G}$
and *cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|AdjNT|)$ =
cf-adjunction-AdjNT-of-unit α $\mathfrak{F}$ $\mathfrak{G}$ η
⟨*proof*⟩

Natural transformation map.

lemma *cf-adjunction-of-unit-AdjNT-NTMap-vdomain*[*adj-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{D}_\circ$ (*cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|AdjNT|)(|NTMap|)$) =
(*op-cat* $\mathfrak{C} \times_C \mathfrak{D})(|Obj|)$
⟨*proof*⟩

lemma *cf-adjunction-of-unit-AdjNT-NTMap-app*[*adj-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $c \in_\circ \mathfrak{C}(|Obj|)$ **and** $d \in_\circ \mathfrak{D}(|Obj|)$
shows
cf-adjunction-of-unit α $\mathfrak{F}$ $\mathfrak{G}$ $\eta(|AdjNT|)(|NTMap|)(|c, d|)_\bullet$ =
umap-of $\mathfrak{G}$ c $(\mathfrak{F}(|ObjMap|)(|c|))$ $(\eta(|NTMap|)(|c|))$ d
⟨*proof*⟩

The adjunction constructed from universal morphisms from objects to functors is an adjunction.

lemma *cf-adjunction-of-unit-is-cf-adjunction*:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and η : *cf-id* $\mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge x.\ x \in_\circ \mathfrak{C}(|Obj|) \Longrightarrow$ *universal-arrow-of* $\mathfrak{G}$ x $(\mathfrak{F}(|ObjMap|)(|x|))$ $(\eta(|NTMap|)(|x|))$
shows *cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ $\eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and η_C (*cf-adjunction-of-unit* α $\mathfrak{F}$ $\mathfrak{G}$ η) = η
⟨*proof*⟩

13.8 Construction of an adjunction from a functor and universal morphisms from objects to functors

The subsection presents the construction of an adjunction given a functor and a structured collection of universal morphisms from objects to functors. The content of this subsection follows the statement and the proof of Theorem 2-ii in Chapter IV-1 in [9].

13.8.1 Left adjoint

definition *cf-la-of-ra* :: $(V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-la-of-ra* F $\mathfrak{G}$ η =
[
$(\lambda x \in_\circ \mathfrak{G}(|HomCod|)(|Obj|).\ F\ x)$,
(
$\lambda h \in_\circ \mathfrak{G}(|HomCod|)(|Arr|).\ THE\ f'.$
$f' : F\ (\mathfrak{G}(|HomCod|)(|Dom|)(|h|)) \mapsto_{\mathfrak{G}(|HomDom|)} F\ (\mathfrak{G}(|HomCod|)(|Cod|)(|h|)) \wedge$
$\eta(|\mathfrak{G}(|HomCod|)(|Cod|)(|h|)|) \circ_{A\mathfrak{G}(|HomCod|)} h$ =
(
umap-of
$\mathfrak{G}$
$(\mathfrak{G}(|HomCod|)(|Dom|)(|h|))$
$(F\ (\mathfrak{G}(|HomCod|)(|Dom|)(|h|)))$
$(\eta(|\mathfrak{G}(|HomCod|)(|Dom|)(|h|)|))$

(F (𝔊⦇HomCod⦈⦇Cod⦈⦇h⦈))
)⦇ArrVal⦈⦇f'⦈
),
𝔊⦇HomCod⦈,
𝔊⦇HomDom⦈
]∘

Components.

lemma *cf-la-of-ra-components*:
shows *cf-la-of-ra F 𝔊 η*⦇*ObjMap*⦈ = (λ*x*∈∘𝔊⦇*HomCod*⦈⦇*Obj*⦈. *F x*)
and *cf-la-of-ra F 𝔊 η*⦇*ArrMap*⦈ =
(
λ*h*∈∘𝔊⦇*HomCod*⦈⦇*Arr*⦈. *THE f'*.
f' : *F* (𝔊⦇*HomCod*⦈⦇*Dom*⦈⦇*h*⦈) ↦𝔊⦇*HomDom*⦈ *F* (𝔊⦇*HomCod*⦈⦇*Cod*⦈⦇*h*⦈) ∧
η⦇𝔊⦇*HomCod*⦈⦇*Cod*⦈⦇*h*⦈⦈ ∘*A*𝔊⦇*HomCod*⦈ *h* =
(
umap-of
𝔊
(𝔊⦇*HomCod*⦈⦇*Dom*⦈⦇*h*⦈)
(*F* (𝔊⦇*HomCod*⦈⦇*Dom*⦈⦇*h*⦈))
(η⦇𝔊⦇*HomCod*⦈⦇*Dom*⦈⦇*h*⦈⦈)
(*F* (𝔊⦇*HomCod*⦈⦇*Cod*⦈⦇*h*⦈))
)⦇*ArrVal*⦈⦇*f'*⦈
)
and *cf-la-of-ra F 𝔊 η*⦇*HomDom*⦈ = 𝔊⦇*HomCod*⦈
and *cf-la-of-ra F 𝔊 η*⦇*HomCod*⦈ = 𝔊⦇*HomDom*⦈
⟨*proof*⟩

13.8.2 Object map

mk-VLambda *cf-la-of-ra-components*(*1*)
|*vsv cf-la-of-ra-ObjMap-vsv*[*adj-cs-intros*]|

mk-VLambda (in *is-functor*)
cf-la-of-ra-components(*1*)[**where** *?𝔊=𝔉*, *unfolded cf-HomCod*]
|*vdomain cf-la-of-ra-ObjMap-vdomain*[*adj-cs-simps*]|
|*app cf-la-of-ra-ObjMap-app*[*adj-cs-simps*]|

lemmas [*adj-cs-simps*] =
is-functor.cf-la-of-ra-ObjMap-vdomain
is-functor.cf-la-of-ra-ObjMap-app

13.8.3 Arrow map

mk-VLambda *cf-la-of-ra-components*(*2*)
|*vsv cf-la-of-ra-ArrMap-vsv*[*adj-cs-intros*]|

mk-VLambda (in *is-functor*)
cf-la-of-ra-components(*2*)[**where** *?𝔊=𝔉*, *unfolded cf-HomCod cf-HomDom*]
|*vdomain cf-la-of-ra-ArrMap-vdomain*[*adj-cs-simps*]|
|*app cf-la-of-ra-ArrMap-app*|

lemmas [*adj-cs-simps*] = *is-functor.cf-la-of-ra-ArrMap-vdomain*

lemma (in *is-functor*) *cf-la-of-ra-ArrMap-app'*:
assumes *h* : *a* ↦𝔅 *b*
shows
cf-la-of-ra F 𝔉 η⦇*ArrMap*⦈⦇*h*⦈ =

(
 THE f'.
 $f' : F\ a \mapsto_{\mathfrak{A}} F\ b\ \wedge$
 $\eta(\!|b|\!) \circ_{A\mathfrak{B}} h = \textit{umap-of}\ \mathfrak{F}\ a\ (F\ a)\ (\eta(\!|a|\!))\ (F\ b)(\!|\textit{ArrVal}|\!)(\!|f'|\!)$
)
⟨*proof*⟩

lemma *cf-la-of-ra-ArrMap-app-unique*:
 assumes $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
 and $f : a \mapsto_{\mathfrak{C}} b$
 and $\textit{universal-arrow-of}\ \mathfrak{G}\ a\ (\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ObjMap}|\!)(\!|a|\!))\ (\eta(\!|a|\!))$
 and $\textit{universal-arrow-of}\ \mathfrak{G}\ b\ (\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ObjMap}|\!)(\!|b|\!))\ (\eta(\!|b|\!))$
 shows $\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ArrMap}|\!)(\!|f|\!) : F\ a \mapsto_{\mathfrak{D}} F\ b$
 and $\eta(\!|b|\!) \circ_{A\mathfrak{C}} f =$
 $\textit{umap-of}\ \mathfrak{G}\ a\ (F\ a)\ (\eta(\!|a|\!))\ (F\ b)(\!|\textit{ArrVal}|\!)(\!|\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ArrMap}|\!)(\!|f|\!)|\!)$
 and $\bigwedge f'$.
 $[\![$
 $f' : F\ a \mapsto_{\mathfrak{D}} F\ b;$
 $\eta(\!|b|\!) \circ_{A\mathfrak{C}} f = \textit{umap-of}\ \mathfrak{G}\ a\ (F\ a)\ (\eta(\!|a|\!))\ (F\ b)(\!|\textit{ArrVal}|\!)(\!|f'|\!)$
 $]\!] \Longrightarrow \textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ArrMap}|\!)(\!|f|\!) = f'$
⟨*proof*⟩

lemma *cf-la-of-ra-ArrMap-app-is-arr*[*adj-cs-intros*]:
 assumes $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
 and $f : a \mapsto_{\mathfrak{C}} b$
 and $\textit{universal-arrow-of}\ \mathfrak{G}\ a\ (\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ObjMap}|\!)(\!|a|\!))\ (\eta(\!|a|\!))$
 and $\textit{universal-arrow-of}\ \mathfrak{G}\ b\ (\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ObjMap}|\!)(\!|b|\!))\ (\eta(\!|b|\!))$
 and $Fa = F\ a$
 and $Fb = F\ b$
 shows $\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ArrMap}|\!)(\!|f|\!) : Fa \mapsto_{\mathfrak{D}} Fb$
 ⟨*proof*⟩

13.8.4 An adjunction constructed from a functor and universal morphisms from objects to functors is an adjunction

lemma *cf-la-of-ra-is-functor*:
 assumes $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
 and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|\textit{Obj}|\!) \Longrightarrow F\ c \in_\circ \mathfrak{D}(\!|\textit{Obj}|\!)$
 and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|\textit{Obj}|\!) \Longrightarrow$
 $\textit{universal-arrow-of}\ \mathfrak{G}\ c\ (\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ObjMap}|\!)(\!|c|\!))\ (\eta(\!|c|\!))$
 and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
 $\mathfrak{G}(\!|\textit{ArrMap}|\!)(\!|\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta(\!|\textit{ArrMap}|\!)(\!|h|\!)|\!) \circ_{A\mathfrak{C}} \eta(\!|c|\!) = \eta(\!|c'|\!) \circ_{A\mathfrak{C}} h$
 shows $\textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ (**is** ‹$?\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$›)
⟨*proof*⟩

lemma *cf-la-of-ra-is-ntcf*:
 fixes $F\ \mathfrak{C}\ \mathfrak{F}\ \mathfrak{G}\ \eta_m\ \eta$
 defines $\mathfrak{F} \equiv \textit{cf-la-of-ra}\ F\ \mathfrak{G}\ \eta_m$
 and $\eta \equiv [\eta_m,\ \textit{cf-id}\ \mathfrak{C},\ \mathfrak{G} \circ_{CF} \mathfrak{F},\ \mathfrak{C},\ \mathfrak{C}]_\circ$
 assumes $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
 and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|\textit{Obj}|\!) \Longrightarrow F\ c \in_\circ \mathfrak{D}(\!|\textit{Obj}|\!)$
 and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|\textit{Obj}|\!) \Longrightarrow \textit{universal-arrow-of}\ \mathfrak{G}\ c\ (\mathfrak{F}(\!|\textit{ObjMap}|\!)(\!|c|\!))\ (\eta(\!|\textit{NTMap}|\!)(\!|c|\!))$
 and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
 $\mathfrak{G}(\!|\textit{ArrMap}|\!)(\!|\mathfrak{F}(\!|\textit{ArrMap}|\!)(\!|h|\!)|\!) \circ_{A\mathfrak{C}} (\eta(\!|\textit{NTMap}|\!)(\!|c|\!)) = (\eta(\!|\textit{NTMap}|\!)(\!|c'|\!)) \circ_{A\mathfrak{C}} h$
 and $\textit{vsv}\ (\eta(\!|\textit{NTMap}|\!))$
 and $\mathcal{D}_\circ\ (\eta(\!|\textit{NTMap}|\!)) = \mathfrak{C}(\!|\textit{Obj}|\!)$
 shows $\eta : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-la-of-ra-is-unit*:
  fixes $F\ \mathfrak{C}\ \mathfrak{F}\ \mathfrak{G}\ \eta_m\ \eta$
  defines $\mathfrak{F} \equiv$ *cf-la-of-ra* $F\ \mathfrak{G}\ \eta_m$
    and $\eta \equiv [\eta_m,$ *cf-id* $\mathfrak{C}, \mathfrak{G} \circ_{CF} \mathfrak{F}, \mathfrak{C}, \mathfrak{C}]_\circ$
  assumes *category* $\alpha\ \mathfrak{C}$
    and *category* $\alpha\ \mathfrak{D}$
    and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\llparenthesis Obj \rrparenthesis) \Longrightarrow F\ c \in_\circ \mathfrak{D}(\llparenthesis Obj \rrparenthesis)$
    and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\llparenthesis Obj \rrparenthesis) \Longrightarrow$
      universal-arrow-of $\mathfrak{G}\ c\ (\mathfrak{F}(\llparenthesis ObjMap \rrparenthesis)(\llparenthesis c \rrparenthesis))\ (\eta(\llparenthesis NTMap \rrparenthesis)(\llparenthesis c \rrparenthesis))$
    and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
      $\mathfrak{G}(\llparenthesis ArrMap \rrparenthesis)(\llparenthesis \mathfrak{F}(\llparenthesis ArrMap \rrparenthesis)(\llparenthesis h \rrparenthesis) \rrparenthesis) \circ_{A\mathfrak{C}} (\eta(\llparenthesis NTMap \rrparenthesis)(\llparenthesis c \rrparenthesis)) = (\eta(\llparenthesis NTMap \rrparenthesis)(\llparenthesis c' \rrparenthesis)) \circ_{A\mathfrak{C}} h$
    and *vsv* $(\eta(\llparenthesis NTMap \rrparenthesis))$
    and $\mathcal{D}_\circ\ (\eta(\llparenthesis NTMap \rrparenthesis)) = \mathfrak{C}(\llparenthesis Obj \rrparenthesis)$
  shows *cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
    and η_C (*cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \eta$
⟨*proof*⟩

13.9 Construction of an adjunction from universal morphisms from functors to objects

13.9.1 Definition and elementary properties

The subsection presents the construction of an adjunction given a structured collection of universal morphisms from functors to objects. The content of this subsection follows the statement and the proof of Theorem 2-iii in Chapter IV-1 in [9].

definition *cf-adjunction-of-counit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon =$
    op-cf-adj (*cf-adjunction-of-unit* α (*op-cf* $\mathfrak{G}$) (*op-cf* $\mathfrak{F}$) (*op-ntcf* ε))

Components.

lemma *cf-adjunction-of-counit-components*:
  shows *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(\llparenthesis AdjLeft \rrparenthesis) =$ *op-cf* (*op-cf* $\mathfrak{F}$)
    and *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(\llparenthesis AdjRight \rrparenthesis) =$ *op-cf* (*op-cf* $\mathfrak{G}$)
    and *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(\llparenthesis AdjNT \rrparenthesis) =$ *op-cf-adj-nt*
      (*op-cf* $\mathfrak{G}(\llparenthesis HomDom \rrparenthesis)$)
      (*op-cf* $\mathfrak{G}(\llparenthesis HomCod \rrparenthesis)$)
      (*cf-adjunction-AdjNT-of-unit* α (*op-cf* $\mathfrak{G}$) (*op-cf* $\mathfrak{F}$) (*op-ntcf* ε))
  ⟨*proof*⟩

13.9.2 Natural transformation map

lemma *cf-adjunction-of-counit-NTMap-vsv*:
  vsv (*cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(\llparenthesis AdjNT \rrparenthesis)(\llparenthesis NTMap \rrparenthesis)$)
  ⟨*proof*⟩

13.9.3 An adjunction constructed from universal morphisms from functors to objects is an adjunction

lemma *cf-adjunction-of-counit-is-cf-adjunction*:
  assumes *category* $\alpha\ \mathfrak{C}$
    and *category* $\alpha\ \mathfrak{D}$
    and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF}$ *cf-id* $\mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $\bigwedge x.\ x \in_\circ \mathfrak{D}(\llparenthesis Obj \rrparenthesis) \Longrightarrow$ *universal-arrow-fo* $\mathfrak{F}\ x\ (\mathfrak{G}(\llparenthesis ObjMap \rrparenthesis)(\llparenthesis x \rrparenthesis))\ (\varepsilon(\llparenthesis NTMap \rrparenthesis)(\llparenthesis x \rrparenthesis))$

shows $\mathit{cf\text{-}adjunction\text{-}of\text{-}counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\varepsilon_C\ (\mathit{cf\text{-}adjunction\text{-}of\text{-}counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \varepsilon$
and $\mathcal{D}_\circ\ (\mathit{cf\text{-}adjunction\text{-}of\text{-}counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(|\mathit{AdjNT}|)(|\mathit{NTMap}|)) =$
$(\mathit{op\text{-}cat}\ \mathfrak{C} \times_C \mathfrak{D})(|\mathit{Obj}|)$
and $\bigwedge c\ d.\ [\![\ c \in_\circ \mathfrak{C}(|\mathit{Obj}|);\ d \in_\circ \mathfrak{D}(|\mathit{Obj}|)\ ]\!] \Longrightarrow$
$\mathit{cf\text{-}adjunction\text{-}of\text{-}counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon(|\mathit{AdjNT}|)(|\mathit{NTMap}|)(|c,\ d|)_\bullet =$
$(\mathit{umap\text{-}fo}\ \mathfrak{F}\ d\ (\mathfrak{G}(|\mathit{ObjMap}|)(|d|))\ (\varepsilon(|\mathit{NTMap}|)(|d|))\ c)^{-1}{}_{Set}$
⟨*proof*⟩

13.10 Construction of an adjunction from a functor and universal morphisms from functors to objects

The subsection presents the construction of an adjunction given a functor and a structured collection of universal morphisms from functors to objects. The content of this subsection follows the statement and the proof of Theorem 2-iv in Chapter IV-1 in [9].

13.10.1 Definition and elementary properties

definition $\mathit{cf\text{-}ra\text{-}of\text{-}la} :: (V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
where $\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon = \mathit{op\text{-}cf}\ (\mathit{cf\text{-}la\text{-}of\text{-}ra}\ F\ (\mathit{op\text{-}cf}\ \mathfrak{F})\ \varepsilon)$

13.10.2 Object map

lemma *cf-ra-of-la-ObjMap-vsv*[*adj-cs-intros*]: $\mathit{vsv}\ (\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ObjMap}|))$
⟨*proof*⟩

lemma (in *is-functor***)** *cf-ra-of-la-ObjMap-vdomain*:
$\mathcal{D}_\circ\ (\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ObjMap}|)) = \mathfrak{B}(|\mathit{Obj}|)$
⟨*proof*⟩

lemmas [*adj-cs-simps*] = *is-functor.cf-ra-of-la-ObjMap-vdomain*

lemma (in *is-functor***)** *cf-ra-of-la-ObjMap-app*:
assumes $d \in_\circ \mathfrak{B}(|\mathit{Obj}|)$
shows $\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ObjMap}|)(|d|) = F\ d$
⟨*proof*⟩

lemmas [*adj-cs-simps*] = *is-functor.cf-ra-of-la-ObjMap-app*

13.10.3 Arrow map

lemma *cf-ra-of-la-ArrMap-app-unique*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} b$
and $\mathit{universal\text{-}arrow\text{-}fo}\ \mathfrak{F}\ a\ (\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ObjMap}|)(|a|))\ (\varepsilon(|a|))$
and $\mathit{universal\text{-}arrow\text{-}fo}\ \mathfrak{F}\ b\ (\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ObjMap}|)(|b|))\ (\varepsilon(|b|))$
shows $\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ArrMap}|)(|f|) : F\ a \mapsto_{\mathfrak{C}} F\ b$
and $f \circ_{A\mathfrak{D}} \varepsilon(|a|) =$
$\mathit{umap\text{-}fo}\ \mathfrak{F}\ b\ (F\ b)\ (\varepsilon(|b|))\ (F\ a)(|\mathit{ArrVal}|)(|\mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ArrMap}|)(|f|)|)$
and $\bigwedge f'.$
$[\![$
$f' : F\ a \mapsto_{\mathfrak{C}} F\ b;$
$f \circ_{A\mathfrak{D}} \varepsilon(|a|) = \mathit{umap\text{-}fo}\ \mathfrak{F}\ b\ (F\ b)\ (\varepsilon(|b|))\ (F\ a)(|\mathit{ArrVal}|)(|f'|)$
$]\!] \Longrightarrow \mathit{cf\text{-}ra\text{-}of\text{-}la}\ F\ \mathfrak{F}\ \varepsilon(|\mathit{ArrMap}|)(|f|) = f'$
⟨*proof*⟩

lemma *cf-ra-of-la-ArrMap-app-is-arr*[*adj-cs-intros*]:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$

and $f : a \mapsto_{\mathfrak{D}} b$
and *universal-arrow-fo* $\mathfrak{F}$ a (*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ObjMap|)(|a|)$) ($\varepsilon(|a|)$)
and *universal-arrow-fo* $\mathfrak{F}$ b (*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ObjMap|)(|b|)$) ($\varepsilon(|b|)$)
and $Fa = F\ a$
and $Fb = F\ b$
shows *cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ArrMap|)(|f|) : Fa \mapsto_{\mathfrak{C}} Fb$
⟨*proof*⟩

13.10.4 An adjunction constructed from a functor and universal morphisms from functors to objects is an adjunction

lemma *op-cf-cf-la-of-ra-op*[*cat-op-simps*]:
op-cf (*cf-la-of-ra* F (*op-cf* $\mathfrak{F}$) ε) = *cf-ra-of-la* F $\mathfrak{F}$ ε
⟨*proof*⟩

lemma *cf-ra-of-la-commute-op*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}$ d (*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ObjMap|)(|d|)$) ($\varepsilon(|d|)$)
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(|d'|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|$*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ArrMap|)(|h|)|) =$
$h \circ_{A\mathfrak{D}} \varepsilon(|d|)$
and $h : c' \mapsto_{\mathfrak{D}} c$
shows $\mathfrak{F}(|ArrMap|)(|$*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ArrMap|)(|h|)|) \circ_{A\,op\text{-}cat\ \mathfrak{D}} \varepsilon(|c|) =$
$\varepsilon(|c'|) \circ_{A\,op\text{-}cat\ \mathfrak{D}} h$
⟨*proof*⟩

lemma
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow F\ d \in_\circ \mathfrak{C}(|Obj|)$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}$ d (*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ObjMap|)(|d|)$) ($\varepsilon(|d|)$)
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(|d'|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|$*cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon(|ArrMap|)(|h|)|) =$
$h \circ_{A\mathfrak{D}} \varepsilon(|d|)$
shows *cf-ra-of-la-is-functor*: *cf-ra-of-la* F $\mathfrak{F}$ $\varepsilon : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *cf-la-of-ra-op-is-functor*:
cf-la-of-ra F (*op-cf* $\mathfrak{F}$) ε : *op-cat* $\mathfrak{D} \mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemma *cf-ra-of-la-is-ntcf*:
fixes F $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ ε_m ε
defines $\mathfrak{G} \equiv$ *cf-ra-of-la* F $\mathfrak{F}$ ε_m
and $\varepsilon \equiv [\varepsilon_m, \mathfrak{F} \circ_{CF} \mathfrak{G},$ *cf-id* $\mathfrak{D}, \mathfrak{D}, \mathfrak{D}]_\circ$
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow F\ d \in_\circ \mathfrak{C}(|Obj|)$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}$ d ($\mathfrak{G}(|ObjMap|)(|d|)$) ($\varepsilon(|NTMap|)(|d|)$)
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(|NTMap|)(|d'|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|\mathfrak{G}(|ArrMap|)(|h|)|) = h \circ_{A\mathfrak{D}} \varepsilon(|NTMap|)(|d|)$
and *vsv* ($\varepsilon(|NTMap|)$)
and $\mathcal{D}_\circ$ ($\varepsilon(|NTMap|)$) = $\mathfrak{D}(|Obj|)$
shows $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF}$ *cf-id* $\mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-ra-of-la-is-counit*:
fixes F $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ ε_m ε
defines $\mathfrak{G} \equiv$ *cf-ra-of-la* F $\mathfrak{F}$ ε_m

and $\varepsilon \equiv [\varepsilon_m, \mathfrak{F} \circ_{CF} \mathfrak{G}, \textit{cf-id}\ \mathfrak{D}, \mathfrak{D}, \mathfrak{D}]_\circ$
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow F\ d \in_\circ \mathfrak{C}(|Obj|)$
and $\bigwedge d.\ d \in_\circ \mathfrak{D}(|Obj|) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}\ d\ (\mathfrak{G}(|ObjMap|)(|d|))\ (\varepsilon(|NTMap|)(|d|))$
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(|NTMap|)(|d'|) \circ_{A\mathfrak{D}} \mathfrak{F}(|ArrMap|)(|\mathfrak{G}(|ArrMap|)(|h|)|) = h \circ_{A\mathfrak{D}} \varepsilon(|NTMap|)(|d|)$
and *vfsequence* ε
and *vsv* $(\varepsilon(|NTMap|))$
and $\mathcal{D}_\circ\ (\varepsilon(|NTMap|)) = \mathfrak{D}(|Obj|)$
shows *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and ε_C (*cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \varepsilon$
⟨*proof*⟩

13.11 Construction of an adjunction from the counit-unit equations

The subsection presents the construction of an adjunction given two natural transformations satisfying counit-unit equations. The content of this subsection follows the statement and the proof of Theorem 2-v in Chapter IV-1 in [9].

lemma *counit-unit-is-cf-adjunction*:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\eta : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \textit{cf-id}\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \bullet_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = \textit{ntcf-id}\ \mathfrak{F}$
shows *cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and η_C (*cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \eta$
and ε_C (*cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \varepsilon$
⟨*proof*⟩

lemma *counit-unit-cf-adjunction-of-counit-is-cf-adjunction*:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\eta : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \textit{cf-id}\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \bullet_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = \textit{ntcf-id}\ \mathfrak{F}$
shows *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and η_C (*cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \eta$
and ε_C (*cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \varepsilon$
⟨*proof*⟩

13.12 Adjoints are unique up to isomorphism

The content of the following subsection is based predominantly on the statement and the proof of Corollary 1 in Chapter IV-1 in [9]. However, similar results can also be found in section 4 in [14] and in subsection 2.1 in [4].

13.12.1 Definitions and elementary properties

definition *cf-adj-LR-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ Ψ =
[
(
$\lambda x{\in_\circ}\mathfrak{C}(\!|Obj|\!)$. *THE* f'.
let
$\eta = \eta_C\ \Phi$;
$\eta' = \eta_C\ \Psi$;
$\mathfrak{F}x = \mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)$;
$\mathfrak{F}'x = \mathfrak{F}'(\!|ObjMap|\!)(\!|x|\!)$
in
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
$\eta'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}of\ \mathfrak{G}\ x\ (\mathfrak{F}x)\ (\eta(\!|NTMap|\!)(\!|x|\!))\ (\mathfrak{F}'x)(\!|ArrVal|\!)(\!|f'|\!)$
),
$\mathfrak{F}$,
$\mathfrak{F}'$,
$\mathfrak{C}$,
$\mathfrak{D}$
$]_\circ$

definition *cf-adj-RL-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ Ψ =
[
(
$\lambda x{\in_\circ}\mathfrak{D}(\!|Obj|\!)$. *THE* f'.
let
$\varepsilon = \varepsilon_C\ \Phi$;
$\varepsilon' = \varepsilon_C\ \Psi$;
$\mathfrak{G}x = \mathfrak{G}(\!|ObjMap|\!)(\!|x|\!)$;
$\mathfrak{G}'x = \mathfrak{G}'(\!|ObjMap|\!)(\!|x|\!)$
in
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x \wedge$
$\varepsilon'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}fo\ \mathfrak{F}\ x\ \mathfrak{G}x\ (\varepsilon(\!|NTMap|\!)(\!|x|\!))\ \mathfrak{G}'x(\!|ArrVal|\!)(\!|f'|\!)$
),
$\mathfrak{G}'$,
$\mathfrak{G}$,
$\mathfrak{D}$,
$\mathfrak{C}$
$]_\circ$

Components.

lemma *cf-adj-LR-iso-components*:
shows *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(\!|NTMap|\!)$ =
(
$\lambda x{\in_\circ}\mathfrak{C}(\!|Obj|\!)$. *THE* f'.
let
$\eta = \eta_C\ \Phi$;
$\eta' = \eta_C\ \Psi$;
$\mathfrak{F}x = \mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)$;
$\mathfrak{F}'x = \mathfrak{F}'(\!|ObjMap|\!)(\!|x|\!)$
in
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
$\eta'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}of\ \mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(\!|NTMap|\!)(\!|x|\!))\ \mathfrak{F}'x(\!|ArrVal|\!)(\!|f'|\!)$
)
and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(\!|NTDom|\!) = \mathfrak{F}$
and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(\!|NTCod|\!) = \mathfrak{F}'$

and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(|NTDGDom|) = \mathfrak{C}$
and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(|NTDGCod|) = \mathfrak{D}$
⟨*proof*⟩

lemma *cf-adj-RL-iso-components*:
shows *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTMap|)$ =
(
$\lambda x \in_\circ \mathfrak{D}(|Obj|)$. *THE* f'.
let
$\varepsilon = \varepsilon_C\ \Phi$;
$\varepsilon' = \varepsilon_C\ \Psi$;
$\mathfrak{G}x = \mathfrak{G}(|ObjMap|)(|x|)$;
$\mathfrak{G}'x = \mathfrak{G}'(|ObjMap|)(|x|)$
in
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x\ \wedge$
$\varepsilon'(|NTMap|)(|x|)$ = *umap-fo* $\mathfrak{F}$ x $\mathfrak{G}x$ $(\varepsilon(|NTMap|)(|x|))$ $\mathfrak{G}'x(|ArrVal|)(|f'|)$
)
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTDom|) = \mathfrak{G}'$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTCod|) = \mathfrak{G}$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTDGDom|) = \mathfrak{D}$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTDGCod|) = \mathfrak{C}$
⟨*proof*⟩

13.12.2 Natural transformation map

lemma *cf-adj-LR-iso-vsv*[*adj-cs-intros*]:
vsv (*cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(|NTMap|)$)
⟨*proof*⟩

lemma *cf-adj-RL-iso-vsv*[*adj-cs-intros*]:
vsv (*cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTMap|)$)
⟨*proof*⟩

lemma *cf-adj-LR-iso-vdomain*[*adj-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(|NTMap|)$) = $\mathfrak{C}(|Obj|)$
⟨*proof*⟩

lemma *cf-adj-RL-iso-vdomain*[*adj-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTMap|)$) = $\mathfrak{D}(|Obj|)$
⟨*proof*⟩

lemma *cf-adj-LR-iso-app*:
fixes $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ Ψ
assumes $x \in_\circ \mathfrak{C}(|Obj|)$
defines $\mathfrak{F}x \equiv \mathfrak{F}(|ObjMap|)(|x|)$
and $\mathfrak{F}'x \equiv \mathfrak{F}'(|ObjMap|)(|x|)$
and $\eta \equiv \eta_C\ \Phi$
and $\eta' \equiv \eta_C\ \Psi$
shows *cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ $\Psi(|NTMap|)(|x|)$ =
(
THE f'.
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x\ \wedge$
$\eta'(|NTMap|)(|x|)$ = *umap-of* $\mathfrak{G}$ x $\mathfrak{F}x$ $(\eta(|NTMap|)(|x|))$ $\mathfrak{F}'x(|ArrVal|)(|f'|)$
)
⟨*proof*⟩

lemma *cf-adj-RL-iso-app*:
fixes $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ Ψ

assumes $x \in_\circ \mathfrak{D}(|Obj|)$
defines $\mathfrak{G}x \equiv \mathfrak{G}(|ObjMap|)(|x|)$
and $\mathfrak{G}'x \equiv \mathfrak{G}'(|ObjMap|)(|x|)$
and $\varepsilon \equiv \varepsilon_C\ \Phi$
and $\varepsilon' \equiv \varepsilon_C\ \Psi$
shows *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(|NTMap|)(|x|) =$
(
THE f'.
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x\ \wedge$
$\varepsilon'(|NTMap|)(|x|) =$ *umap-fo* $\mathfrak{F}\ x\ \mathfrak{G}x\ (\varepsilon(|NTMap|)(|x|))\ \mathfrak{G}'x(|ArrVal|)(|f'|)$
)
⟨*proof*⟩

lemma *cf-adj-LR-iso-app-unique*:
fixes $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi$
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $x \in_\circ \mathfrak{C}(|Obj|)$
defines $\mathfrak{F}x \equiv \mathfrak{F}(|ObjMap|)(|x|)$
and $\mathfrak{F}'x \equiv \mathfrak{F}'(|ObjMap|)(|x|)$
and $\eta \equiv \eta_C\ \Phi$
and $\eta' \equiv \eta_C\ \Psi$
and $f \equiv$ *cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi(|NTMap|)(|x|)$
shows
$\exists!f'$.
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x\ \wedge$
$\eta'(|NTMap|)(|x|) =$ *umap-of* $\mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(|NTMap|)(|x|))\ \mathfrak{F}'x(|ArrVal|)(|f'|)$
$f : \mathfrak{F}x \mapsto_{iso\mathfrak{D}} \mathfrak{F}'x$
$\eta'(|NTMap|)(|x|) =$ *umap-of* $\mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(|NTMap|)(|x|))\ \mathfrak{F}'x(|ArrVal|)(|f|)$
⟨*proof*⟩

13.12.3 Main results

lemma *cf-adj-LR-iso-is-iso-functor*:
— See Corollary 1 in Chapter IV-1 in [9].
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$ **and** $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
shows $\exists!\vartheta.\ \vartheta : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D} \wedge \eta_C\ \Psi = (\mathfrak{G} \circ_{CF-NTCF} \vartheta) \bullet_{NTCF} \eta_C\ \Phi$
and *cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\eta_C\ \Psi = (\mathfrak{G} \circ_{CF-NTCF}$ *cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi) \bullet_{NTCF} \eta_C\ \Phi$
⟨*proof*⟩

lemma *op-ntcf-cf-adj-RL-iso*[*cat-op-simps*]:
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
defines *op*-$\mathfrak{D} \equiv$ *op-cat* $\mathfrak{D}$
and *op*-$\mathfrak{C} \equiv$ *op-cat* $\mathfrak{C}$
and *op*-$\mathfrak{F} \equiv$ *op-cf* $\mathfrak{F}$
and *op*-$\mathfrak{G} \equiv$ *op-cf* $\mathfrak{G}$
and *op*-$\Phi \equiv$ *op-cf-adj* Φ
and *op*-$\mathfrak{G}' \equiv$ *op-cf* $\mathfrak{G}'$
and *op*-$\Psi \equiv$ *op-cf-adj* Ψ
shows
op-ntcf (*cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi) =$
cf-adj-LR-iso *op*-$\mathfrak{D}$ *op*-$\mathfrak{C}$ *op*-$\mathfrak{F}$ *op*-$\mathfrak{G}$ *op*-Φ *op*-$\mathfrak{G}'$ *op*-Ψ
⟨*proof*⟩

lemma *op-ntcf-cf-adj-LR-iso*[*cat-op-simps*]:
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$ **and** $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$

defines *op-*$\mathfrak{D}$ ≡ *op-cat* $\mathfrak{D}$
and *op-*$\mathfrak{C}$ ≡ *op-cat* $\mathfrak{C}$
and *op-*$\mathfrak{F}$ ≡ *op-cf* $\mathfrak{F}$
and *op-*$\mathfrak{G}$ ≡ *op-cf* $\mathfrak{G}$
and *op-*Φ ≡ *op-cf-adj* Φ
and *op-*$\mathfrak{F}'$ ≡ *op-cf* $\mathfrak{F}'$
and *op-*Ψ ≡ *op-cf-adj* Ψ
shows
op-ntcf (*cf-adj-LR-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{G}$ $\mathfrak{F}$ Φ $\mathfrak{F}'$ Ψ) =
cf-adj-RL-iso *op-*$\mathfrak{D}$ *op-*$\mathfrak{C}$ *op-*$\mathfrak{G}$ *op-*$\mathfrak{F}$ *op-*Φ *op-*$\mathfrak{F}'$ *op-*Ψ
⟨*proof*⟩

lemma *cf-adj-RL-iso-app-unique*:
fixes $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ Ψ
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and $x \in_\circ \mathfrak{D}(|Obj|)$
defines $\mathfrak{G}x \equiv \mathfrak{G}(|ObjMap|)(|x|)$
and $\mathfrak{G}'x \equiv \mathfrak{G}'(|ObjMap|)(|x|)$
and $\varepsilon \equiv \varepsilon_C\ \Phi$
and $\varepsilon' \equiv \varepsilon_C\ \Psi$
and $f \equiv$ *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi(|NTMap|)(|x|)$
shows
$\exists!f'.$
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x\ \wedge$
$\varepsilon'(|NTMap|)(|x|)$ = *umap-fo* $\mathfrak{F}$ x $\mathfrak{G}x$ $(\varepsilon(|NTMap|)(|x|))$ $\mathfrak{G}'x(|ArrVal|)(|f'|)$
$f : \mathfrak{G}'x \mapsto_{iso\mathfrak{C}} \mathfrak{G}x$
$\varepsilon'(|NTMap|)(|x|)$ = *umap-fo* $\mathfrak{F}$ x $\mathfrak{G}x$ $(\varepsilon(|NTMap|)(|x|))$ $\mathfrak{G}'x(|ArrVal|)(|f|)$
⟨*proof*⟩

lemma *cf-adj-RL-iso-is-iso-functor*:
— See Corollary 1 in Chapter IV-1 in [9].
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$ **and** $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
shows $\exists!\vartheta.$
$\vartheta : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}\ \wedge$
$\varepsilon_C\ \Psi = \varepsilon_C\ \Phi \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \vartheta)$
and *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ Ψ $: \mathfrak{G}' \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon_C\ \Psi$ =
$\varepsilon_C\ \Phi \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF}$ *cf-adj-RL-iso* $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ Φ $\mathfrak{G}'$ $\Psi)$
⟨*proof*⟩

13.13 Further properties of the adjoint functors

lemma (in *is-cf-adjunction***)** *cf-adj-exp-cf-cat*:
— See Proposition 4.4.6 in [14].
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$ **and** *category* α $\mathfrak{J}$
shows
cf-adjunction-of-unit
β
(*exp-cf-cat* α $\mathfrak{F}$ $\mathfrak{J}$)
(*exp-cf-cat* α $\mathfrak{G}$ $\mathfrak{J}$)
(*exp-ntcf-cat* α $(\eta_C\ \Phi)$ $\mathfrak{J}$) :
exp-cf-cat α $\mathfrak{F}$ $\mathfrak{J}$ $\rightleftharpoons_{CF}$ *exp-cf-cat* α $\mathfrak{G}$ $\mathfrak{J}$:
cat-FUNCT α $\mathfrak{J}$ $\mathfrak{C}$ $\rightleftharpoons\rightleftharpoons_{C\beta}$ *cat-FUNCT* α $\mathfrak{J}$ $\mathfrak{D}$
⟨*proof*⟩

lemma (in *is-cf-adjunction***)** *cf-adj-exp-cf-cat-exp-cf-cat*:
— See Proposition 4.4.6 in [14].

assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$ **and** *category* $\alpha\ \mathfrak{A}$
shows
cf-adjunction-of-unit
β
(*exp-cat-cf* $\alpha\ \mathfrak{A}\ \mathfrak{G}$)
(*exp-cat-cf* $\alpha\ \mathfrak{A}\ \mathfrak{F}$)
(*exp-cat-ntcf* $\alpha\ \mathfrak{A}\ (\eta_C\ \Phi)$) :
exp-cat-cf $\alpha\ \mathfrak{A}\ \mathfrak{G} \rightleftharpoons_{CF}$ *exp-cat-cf* $\alpha\ \mathfrak{A}\ \mathfrak{F}$:
cat-FUNCT $\alpha\ \mathfrak{C}\ \mathfrak{A} \rightleftharpoons\rightleftharpoons_{C\beta}$ *cat-FUNCT* $\alpha\ \mathfrak{D}\ \mathfrak{A}$
⟨*proof*⟩

13.14 Adjoints on limits

lemma *cf-AdjRight-preserves-limits*:
— See Chapter V-5 in [9].
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{X} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{A}$
shows *is-cf-continuous* $\alpha\ \mathfrak{G}$
⟨*proof*⟩

14 Simple Kan extensions

14.1 Background

named-theorems *cat-Kan-cs-simps*
named-theorems *cat-Kan-cs-intros*

14.2 Kan extension

14.2.1 Definition and elementary properties

See Chapter X-3 in [9].

locale *is-cat-rKe* =
  AG: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ +
  Ran: *is-functor* α $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{G}$ +
  ntcf-rKe: *is-ntcf* α $\mathfrak{B}$ $\mathfrak{A}$ ‹$\mathfrak{G} \circ_{CF} \mathfrak{K}$› $\mathfrak{T}$ ε
  for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε +
  assumes *cat-rKe-ua-fo*:
    universal-arrow-fo
      (*exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}$)
      (*cf-map* $\mathfrak{T}$)
      (*cf-map* $\mathfrak{G}$)
      (*ntcf-arrow* ε)

syntax *-is-cat-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe}$1 - :/ - $\mapsto_C$ - $\mapsto_C$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-rKe* ⇌ *is-cat-rKe*
translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$ ⇌
  CONST is-cat-rKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε

locale *is-cat-lKe* =
  AG: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ +
  Lan: *is-functor* α $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{F}$ +
  ntcf-lKe: *is-ntcf* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ‹$\mathfrak{F} \circ_{CF} \mathfrak{K}$› η
  for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η +
  assumes *cat-lKe-ua-fo*:
    universal-arrow-fo
      (*exp-cat-cf* α (*op-cat* $\mathfrak{A}$) (*op-cf* $\mathfrak{K}$))
      (*cf-map* $\mathfrak{T}$)
      (*cf-map* $\mathfrak{F}$)
      (*ntcf-arrow* (*op-ntcf* η))

syntax *-is-cat-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $\mapsto_{CF.lKe}$1 - $\circ_{CF}$ - :/ - $\mapsto_C$ - $\mapsto_C$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-lKe* ⇌ *is-cat-lKe*
translations $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$ ⇌
  CONST is-cat-lKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η

Rules.

lemma (**in** *is-cat-rKe*) *is-cat-rKe-axioms'*[*cat-Kan-cs-intros*]:
  assumes $\alpha' = \alpha$
    and $\mathfrak{G}' = \mathfrak{G}$
    and $\mathfrak{K}' = \mathfrak{K}$
    and $\mathfrak{T}' = \mathfrak{T}$
    and $\mathfrak{B}' = \mathfrak{B}$
    and $\mathfrak{A}' = \mathfrak{A}$
    and $\mathfrak{C}' = \mathfrak{C}$
  shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe_{\alpha'}} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$

⟨*proof*⟩

mk-ide rf *is-cat-rKe-def*[*unfolded is-cat-rKe-axioms-def*]
|*intro is-cat-rKeI*|
|*dest is-cat-rKeD*[*dest*]|
|*elim is-cat-rKeE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-rKeD*(*1–3*)

lemma (**in** *is-cat-lKe*) *is-cat-lKe-axioms'*[*cat-Kan-cs-intros*]:
assumes $\alpha' = \alpha$
and $\mathfrak{F}' = \mathfrak{F}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\eta : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{F}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
⟨*proof*⟩

mk-ide rf *is-cat-lKe-def*[*unfolded is-cat-lKe-axioms-def*]
|*intro is-cat-lKeI*|
|*dest is-cat-lKeD*[*dest*]|
|*elim is-cat-lKeE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-lKeD*(*1–3*)

Duality.

lemma (**in** *is-cat-rKe*) *is-cat-lKe-op*:
op-ntcf ε :
op-cf $\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *op-cf* $\mathfrak{G} \circ_{CF}$ *op-cf* $\mathfrak{K}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-rKe*) *is-cat-lKe-op'*[*cat-op-intros*]:
assumes $\mathfrak{T}'$ = *op-cf* $\mathfrak{T}$
and $\mathfrak{G}'$ = *op-cf* $\mathfrak{G}$
and $\mathfrak{K}'$ = *op-cf* $\mathfrak{K}$
and $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
and $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$
and $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\varepsilon : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-rKe.is-cat-lKe-op'*

lemma (**in** *is-cat-lKe*) *is-cat-rKe-op*:
op-ntcf η :
op-cf $\mathfrak{F} \circ_{CF}$ *op-cf* $\mathfrak{K} \mapsto_{CF.rKe\alpha}$ *op-cf* $\mathfrak{T}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-lKe*) *is-cat-lKe-op'*[*cat-op-intros*]:
assumes $\mathfrak{T}'$ = *op-cf* $\mathfrak{T}$
and $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$
and $\mathfrak{K}'$ = *op-cf* $\mathfrak{K}$
and $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
and $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$

and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
shows $op\text{-}ntcf\ \eta : \mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-lKe.is-cat-lKe-op'*

Elementary properties.

lemma (**in** *is-cat-rKe*) *cat-rKe-exp-cat-cf-cat-FUNCT-is-arr*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $exp\text{-}cat\text{-}cf\ \alpha\ \mathfrak{A}\ \mathfrak{K} : cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} cat\text{-}FUNCT\ \alpha\ \mathfrak{B}\ \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-lKe*) *cat-lKe-exp-cat-cf-cat-FUNCT-is-arr*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $exp\text{-}cat\text{-}cf\ \alpha\ \mathfrak{A}\ \mathfrak{K} : cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} cat\text{-}FUNCT\ \alpha\ \mathfrak{B}\ \mathfrak{A}$
⟨*proof*⟩

14.2.2 Universal property

See Chapter X-3 in [9] and [2][8].

lemma *is-cat-rKeI'*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge\mathfrak{G}'\ \varepsilon'.$
$[\![\ \mathfrak{G}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A};\ \varepsilon' : \mathfrak{G}' \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ ]\!] \Longrightarrow$
$\exists!\sigma.\ \sigma : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \bullet_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$
shows $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma *is-cat-lKeI'*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\eta : \mathfrak{T} \mapsto_{CF} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge\mathfrak{F}'\ \eta'.$
$[\![\ \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A};\ \eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ ]\!] \Longrightarrow$
$\exists!\sigma.\ \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \bullet_{NTCF} \eta$
shows $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-rKe*) *cat-rKe-unique*:
assumes $\mathfrak{G}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $\varepsilon' : \mathfrak{G}' \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\exists!\sigma.\ \sigma : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \bullet_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$
⟨*proof*⟩

lemma (**in** *is-cat-lKe*) *cat-lKe-unique*:
assumes $\mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\exists!\sigma.\ \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \bullet_{NTCF} \eta$
⟨*proof*⟩

14.2.3 Further properties

lemma (**in** *is-cat-rKe*) *cat-rKe-ntcf-ua-fo-is-iso-ntcf-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows

[8] https://en.wikipedia.org/wiki/Kan_extension

ntcf-ua-fo β (*exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}$) (*cf-map* $\mathfrak{T}$) (*cf-map* $\mathfrak{G}$) (*ntcf-arrow* ε) :
$Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{C}$ $\mathfrak{A}$(−,*cf-map* $\mathfrak{G}$) $\mapsto_{CF.iso}$
$Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{B}$ $\mathfrak{A}$(−,*cf-map* $\mathfrak{T}$) $\circ_{CF}$ *op-cf* (*exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}$) :
op-cat (*cat-FUNCT* α $\mathfrak{C}$ $\mathfrak{A}$) $\mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

lemma (**in** *is-cat-lKe*) *cat-lKe-ntcf-ua-fo-is-iso-ntcf-if-ge-Limit*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
defines $\mathfrak{A}\mathfrak{K}$ ≡ *exp-cat-cf* α (*op-cat* $\mathfrak{A}$) (*op-cf* $\mathfrak{K}$)
and $\mathfrak{A}\mathfrak{C}$ ≡ *cat-FUNCT* α (*op-cat* $\mathfrak{C}$) (*op-cat* $\mathfrak{A}$)
and $\mathfrak{A}\mathfrak{B}$ ≡ *cat-FUNCT* α (*op-cat* $\mathfrak{B}$) (*op-cat* $\mathfrak{A}$)
shows
ntcf-ua-fo β $\mathfrak{A}\mathfrak{K}$ (*cf-map* $\mathfrak{T}$) (*cf-map* $\mathfrak{F}$) (*ntcf-arrow* (*op-ntcf* η)) :
$Hom_{O.C\beta}\mathfrak{A}\mathfrak{C}$(−,*cf-map* $\mathfrak{F}$) $\mapsto_{CF.iso}$ $Hom_{O.C\beta}\mathfrak{A}\mathfrak{B}$(−,*cf-map* $\mathfrak{T}$) $\circ_{CF}$ *op-cf* $\mathfrak{A}\mathfrak{K}$:
op-cat $\mathfrak{A}\mathfrak{C}$ $\mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

14.3 Opposite universal arrow for Kan extensions

14.3.1 Definition and elementary properties

The following definition is merely a convenience utility for the exposition of dual results associated with the formula for the right Kan extension and the pointwise right Kan extension.

definition *op-ua* :: (V ⇒ V) ⇒ V ⇒ V ⇒ V
where *op-ua lim-Obj* $\mathfrak{K}$ c =
[
lim-Obj c⦇*UObj*⦈,
op-ntcf (*lim-Obj* c⦇*UArr*⦈) $\circ_{NTCF-CF}$ *inv-cf* (*op-cf-obj-comma* $\mathfrak{K}$ c)
]$_\circ$

Components.

lemma *op-ua-components*:
shows [*cat-op-simps*]: *op-ua lim-Obj* $\mathfrak{K}$ c⦇*UObj*⦈ = *lim-Obj* c⦇*UObj*⦈
and *op-ua lim-Obj* $\mathfrak{K}$ c⦇*UArr*⦈ =
op-ntcf (*lim-Obj* c⦇*UArr*⦈) $\circ_{NTCF-CF}$ *inv-cf* (*op-cf-obj-comma* $\mathfrak{K}$ c)
⟨*proof*⟩

14.3.2 Opposite universal arrow for Kan extensions is a limit

lemma *op-ua-UArr-is-cat-limit*:
assumes $\mathfrak{K}$: $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{A}$
and $c \in_\circ \mathfrak{C}$⦇*Obj*⦈
and u : $\mathfrak{T}$ $\circ_{CF}$ $\mathfrak{K}$ $_{CF}\sqcap_O$ c $>_{CF.colim}$ r : $\mathfrak{K}$ $_{CF}\downarrow$ c $\mapsto\mapsto_{C\alpha}$ $\mathfrak{A}$
shows *op-ntcf* u $\circ_{NTCF-CF}$ *inv-cf* (*op-cf-obj-comma* $\mathfrak{K}$ c) :
r $<_{CF.lim}$ *op-cf* $\mathfrak{T}$ $\circ_{CF}$ c $_O\sqcap_{CF}$ (*op-cf* $\mathfrak{K}$) : c $\downarrow_{CF}$ (*op-cf* $\mathfrak{K}$) $\mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

context
fixes *lim-Obj* :: V ⇒ V **and** c :: V
begin

lemmas *op-ua-UArr-is-cat-limit'* = *op-ua-UArr-is-cat-limit*
[
unfolded op-ua-components(*2*)[*symmetric*],
where *u*=‹*lim-Obj c*⦇*UArr*⦈› **and** *r*=‹*lim-Obj c*⦇*UObj*⦈› **and** *c*=*c*,
folded op-ua-components(*2*)[**where** *lim-Obj*=*lim-Obj* **and** *c*=*c*]

```
  ]
end
```

14.4 The Kan extension

The following subsection is based on the statement and proof of Theorem 1 in Chapter X-3 in [9].

14.4.1 Definition and elementary properties

```
definition the-cf-rKe :: V ⇒ V ⇒ V ⇒ (V ⇒ V) ⇒ V
  where the-cf-rKe α 𝔗 𝔎 lim-Obj =
    [
      (λc∈∘𝔎(|HomCod|)(|Obj|). lim-Obj c(|UObj|)),
      (
        λg∈∘𝔎(|HomCod|)(|Arr|). THE f.
          f :
            lim-Obj (𝔎(|HomCod|)(|Dom|)(|g|))(|UObj|) ↦𝔗(|HomCod|)
            lim-Obj (𝔎(|HomCod|)(|Cod|)(|g|))(|UObj|) ∧
          lim-Obj (𝔎(|HomCod|)(|Dom|)(|g|))(|UArr|) ∘NTCF-CF g A↓CF 𝔎 =
            lim-Obj (𝔎(|HomCod|)(|Cod|)(|g|))(|UArr|) •NTCF
            ntcf-const ((𝔎(|HomCod|)(|Cod|)(|g|)) ↓CF 𝔎) (𝔗(|HomCod|)) f
      ),
      𝔎(|HomCod|),
      𝔗(|HomCod|)
    ]∘

definition the-ntcf-rKe :: V ⇒ V ⇒ V ⇒ (V ⇒ V) ⇒ V
  where the-ntcf-rKe α 𝔗 𝔎 lim-Obj =
    [
      (
        λc∈∘𝔗(|HomDom|)(|Obj|).
          lim-Obj (𝔎(|ObjMap|)(|c|))(|UArr|)(|NTMap|)(|0, c, 𝔎(|HomCod|)(|CId|)(|𝔎(|ObjMap|)(|c|)|)|)•
      ),
      the-cf-rKe α 𝔗 𝔎 lim-Obj ∘CF 𝔎,
      𝔗,
      𝔗(|HomDom|),
      𝔗(|HomCod|)
    ]∘

definition the-cf-lKe :: V ⇒ V ⇒ V ⇒ (V ⇒ V) ⇒ V
  where the-cf-lKe α 𝔗 𝔎 lim-Obj =
    op-cf (the-cf-rKe α (op-cf 𝔗) (op-cf 𝔎) (op-ua lim-Obj 𝔎))

definition the-ntcf-lKe :: V ⇒ V ⇒ V ⇒ (V ⇒ V) ⇒ V
  where the-ntcf-lKe α 𝔗 𝔎 lim-Obj =
    op-ntcf (the-ntcf-rKe α (op-cf 𝔗) (op-cf 𝔎) (op-ua lim-Obj 𝔎))
```

Components.

```
lemma the-cf-rKe-components:
  shows the-cf-rKe α 𝔗 𝔎 lim-Obj(|ObjMap|) =
    (λc∈∘𝔎(|HomCod|)(|Obj|). lim-Obj c(|UObj|))
    and the-cf-rKe α 𝔗 𝔎 lim-Obj(|ArrMap|) =
    (
      λg∈∘𝔎(|HomCod|)(|Arr|). THE f.
        f :
```

$lim\text{-}Obj$ $(\mathfrak{K}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|UObj|\!) \mapsto_{\mathfrak{T}(\!|HomCod|\!)}$
$lim\text{-}Obj$ $(\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))(\!|UObj|\!) \wedge$
$lim\text{-}Obj$ $(\mathfrak{K}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|UArr|\!) \circ_{NTCF-CF}\ g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{K} =$
$lim\text{-}Obj$ $(\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))(\!|UArr|\!) \bullet_{NTCF}$
$ntcf\text{-}const$ $((\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))\downarrow_{CF} \mathfrak{K})$ $(\mathfrak{T}(\!|HomCod|\!))$ f
)
and *the-cf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|HomDom|\!) = \mathfrak{K}(\!|HomCod|\!)$
and *the-cf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|HomCod|\!) = \mathfrak{T}(\!|HomCod|\!)$
⟨*proof*⟩

lemma *the-ntcf-rKe-components*:
shows *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|NTMap|\!) =$
(
$\lambda c \in_{\circ} \mathfrak{T}(\!|HomDom|\!)(\!|Obj|\!)$.
$lim\text{-}Obj$ $(\mathfrak{K}(\!|ObjMap|\!)(\!|c|\!))(\!|UArr|\!)(\!|NTMap|\!)(\!|0,\ c,\ \mathfrak{K}(\!|HomCod|\!)(\!|CId|\!)(\!|\mathfrak{K}(\!|ObjMap|\!)(\!|c|\!)|\!)|\!)$•
)
and *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|NTDom|\!) =$ *the-cf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj \circ_{CF} \mathfrak{K}$
and *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|NTCod|\!) = \mathfrak{T}$
and *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|NTDGDom|\!) = \mathfrak{T}(\!|HomDom|\!)$
and *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ $lim\text{-}Obj(\!|NTDGCod|\!) = \mathfrak{T}(\!|HomCod|\!)$
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ $\mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *the-cf-rKe-components′* = *the-cf-rKe-components*[
where $\mathfrak{K}$=$\mathfrak{K}$ **and** $\mathfrak{T}$=$\mathfrak{T}$ **and** α=α, *unfolded* $\mathfrak{K}$.*cf-HomCod* $\mathfrak{T}$.*cf-HomCod*
]

lemmas [*cat-Kan-cs-simps*] = *the-cf-rKe-components′*(*3*,*4*)

lemmas *the-ntcf-rKe-components′* = *the-ntcf-rKe-components*[
where $\mathfrak{K}$=$\mathfrak{K}$ **and** $\mathfrak{T}$=$\mathfrak{T}$ **and** α=α, *unfolded* $\mathfrak{K}$.*cf-HomCod* $\mathfrak{T}$.*cf-HomCod* $\mathfrak{T}$.*cf-HomDom*
]

lemmas [*cat-Kan-cs-simps*] = *the-ntcf-rKe-components′*(*2*–*5*)

end

14.4.2 Functor: object map

mk-VLambda *the-cf-rKe-components*(*1*)
|*vsv the-cf-rKe-ObjMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ $\mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩

mk-VLambda *the-cf-rKe-components'(1)*[*OF* $\mathfrak{K}$ $\mathfrak{T}$]
|*vdomain the-cf-rKe-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app the-cf-rKe-ObjMap-impl-app*[*cat-Kan-cs-simps*]|

lemma *the-cf-rKe-ObjMap-vrange*:
assumes $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow lim\text{-}Obj\ c(\!|UObj|\!) \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathcal{R}_\circ$ (*the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|ObjMap|\!)) \subseteq_\circ \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

end

14.4.3 Functor: arrow map

mk-VLambda *the-cf-rKe-components(2)*
|*vsv the-cf-rKe-ArrMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\!\mapsto_{C\alpha} \mathfrak{C}$
begin

interpretation $\mathfrak{K}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}$ ⟨*proof*⟩

mk-VLambda *the-cf-rKe-components(2)*[**where** α=α **and** $\mathfrak{K}$=$\mathfrak{K}$, *unfolded* $\mathfrak{K}$*.cf-HomCod*]
|*vdomain the-cf-rKe-ArrMap-vdomain*[*cat-Kan-cs-simps*]|

context
fixes $\mathfrak{A}\ \mathfrak{T}\ c\ c'\ g$
assumes $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\!\mapsto_{C\alpha} \mathfrak{A}$
and g: $g : c \mapsto_{\mathfrak{C}} c'$
begin

interpretation $\mathfrak{T}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{A}\ \mathfrak{T}$ ⟨*proof*⟩

lemma g': $g \in_\circ \mathfrak{C}(\!|Arr|\!)$ ⟨*proof*⟩

mk-VLambda *the-cf-rKe-components(2)*[
where α=α **and** $\mathfrak{K}$=$\mathfrak{K}$ **and** $\mathfrak{T}$=$\mathfrak{T}$, *unfolded* $\mathfrak{K}$*.cf-HomCod* $\mathfrak{T}$*.cf-HomCod*
]
|*app the-cf-rKe-ArrMap-app-impl'*|

lemmas *the-cf-rKe-ArrMap-app'* = *the-cf-rKe-ArrMap-app-impl'*[
OF g', *unfolded* $\mathfrak{K}$*.HomCod.cat-is-arrD*[*OF* g]
]

end

end

lemma *the-cf-rKe-ArrMap-app-impl*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\!\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\!\mapsto_{C\alpha} \mathfrak{A}$
and $g : c \mapsto_{\mathfrak{C}} c'$
and $u : r <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\!\mapsto_{C\alpha} \mathfrak{A}$
and $u' : r' <_{CF.lim} \mathfrak{T} \circ_{CF} c'\ {}_O\sqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\!\mapsto_{C\alpha} \mathfrak{A}$
shows $\exists!f.$
$f : r \mapsto_{\mathfrak{A}} r' \wedge$
$u \circ_{NTCF-CF} g\ {}_A\!\downarrow_{CF} \mathfrak{K} = u' \cdot_{NTCF}$ *ntcf-const* $(c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ f$

⟨*proof*⟩

lemma *the-cf-rKe-ArrMap-app*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $g : c \mapsto_{\mathfrak{C}} c'$
and $\textit{lim-Obj}\ c(|\textit{UArr}|) :$
$\textit{lim-Obj}\ c(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ c\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\textit{lim-Obj}\ c'(|\textit{UArr}|) :$
$\textit{lim-Obj}\ c'(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ c'\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(|\textit{ArrMap}|)(|g|) :$
$\textit{lim-Obj}\ c(|\textit{UObj}|) \mapsto_{\mathfrak{A}} \textit{lim-Obj}\ c'(|\textit{UObj}|)$
and
$\textit{lim-Obj}\ c(|\textit{UArr}|) \circ_{NTCF-CF}\ g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{K} =$
$\textit{lim-Obj}\ c'(|\textit{UArr}|) \bullet_{NTCF}$
$\textit{ntcf-const}\ (c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ (\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(|\textit{ArrMap}|)(|g|))$
and
$[\![$
$f : \textit{lim-Obj}\ c(|\textit{UObj}|) \mapsto_{\mathfrak{A}} \textit{lim-Obj}\ c'(|\textit{UObj}|);$
$\textit{lim-Obj}\ c(|\textit{UArr}|) \circ_{NTCF-CF}\ g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{K} =$
$\textit{lim-Obj}\ c'(|\textit{UArr}|) \bullet_{NTCF}\ \textit{ntcf-const}\ (c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ f$
$]\!] \Longrightarrow f = \textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(|\textit{ArrMap}|)(|g|)$
⟨*proof*⟩

lemma *the-cf-rKe-ArrMap-is-arr'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $g : c \mapsto_{\mathfrak{C}} c'$
and $\textit{lim-Obj}\ c(|\textit{UArr}|) :$
$\textit{lim-Obj}\ c(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ c\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\textit{lim-Obj}\ c'(|\textit{UArr}|) :$
$\textit{lim-Obj}\ c'(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ c'\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a = \textit{lim-Obj}\ c(|\textit{UObj}|)$
and $b = \textit{lim-Obj}\ c'(|\textit{UObj}|)$
shows $\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(|\textit{ArrMap}|)(|g|) : a \mapsto_{\mathfrak{A}} b$
⟨*proof*⟩

lemma *lim-Obj-the-cf-rKe-commute*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\textit{lim-Obj}\ a(|\textit{UArr}|) :$
$\textit{lim-Obj}\ a(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ a\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : a \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\textit{lim-Obj}\ b(|\textit{UArr}|) :$
$\textit{lim-Obj}\ b(|\textit{UObj}|) <_{CF.lim} \mathfrak{T} \circ_{CF}\ b\ {}_{O}\sqcap_{CF}\ \mathfrak{K} : b \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $f : a \mapsto_{\mathfrak{C}} b$
and $[a', b', f']_{\circ} \in_{\circ} b \downarrow_{CF} \mathfrak{K}(|\textit{Obj}|)$
shows
$\textit{lim-Obj}\ a(|\textit{UArr}|)(|\textit{NTMap}|)(|a', b', f' \circ_{A\mathfrak{C}} f|)_{\bullet} =$
$\textit{lim-Obj}\ b(|\textit{UArr}|)(|\textit{NTMap}|)(|a', b', f'|)_{\bullet} \circ_{A\mathfrak{A}}$
$\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(|\textit{ArrMap}|)(|f|)$
⟨*proof*⟩

14.4.4 Natural transformation: natural transformation map

mk-VLambda *the-ntcf-rKe-components*(*1*)
|*vsv the-ntcf-rKe-NTMap-vsv*[*cat-Kan-cs-intros*]|

context

fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ $\mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

mk-VLambda *the-ntcf-rKe-components′(1)*[*OF* $\mathfrak{K}$ $\mathfrak{T}$]
|*vdomain the-ntcf-rKe-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app the-ntcf-rKe-ObjMap-impl-app*[*cat-Kan-cs-simps*]|

end

14.4.5 The Kan extension is a Kan extension

lemma *the-cf-rKe-is-functor*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow$ *lim-Obj* $c(\!|UArr|\!)$:
lim-Obj $c(\!|UObj|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-cf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* : $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *the-cf-lKe-is-functor*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow$ *lim-Obj* $c(\!|UArr|\!)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\sqcap_O\ c >_{CF.colim}$ *lim-Obj* $c(\!|UObj|\!) : \mathfrak{K}\ {}_{CF}\!\downarrow c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-cf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* : $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *the-ntcf-rKe-is-ntcf*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow$ *lim-Obj* $c(\!|UArr|\!)$:
lim-Obj $c(\!|UObj|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* :
the-cf-rKe α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* $\circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *the-ntcf-lKe-is-ntcf*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow$ *lim-Obj* $c(\!|UArr|\!)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\sqcap_O\ c >_{CF.colim}$ *lim-Obj* $c(\!|UObj|\!) : \mathfrak{K}\ {}_{CF}\!\downarrow c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* :
$\mathfrak{T} \mapsto_{CF}$ *the-cf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* $\circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *the-ntcf-rKe-is-cat-rKe*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow$ *lim-Obj* $c(\!|UArr|\!)$:
lim-Obj $c(\!|UObj|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* :
the-cf-rKe α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* $\circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma *the-ntcf-lKe-is-cat-lKe*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(|Obj|) \Longrightarrow$ *lim-Obj* $c(|UArr|)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ _{CF}\sqcap_O\ c >_{CF.colim}$ *lim-Obj* $c(|UObj|) : \mathfrak{K}\ _{CF}\downarrow\ c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* :
$\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *the-cf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* $\circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

14.5 Preservation of Kan extensions

The following definitions are similar to the definitions that can be found in [14] or [8].

locale *is-cat-rKe-preserves* =
is-cat-rKe $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ \varepsilon$ + *is-functor* $\alpha\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{H}$
for $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ \mathfrak{H}\ \varepsilon$ +
assumes *cat-rKe-preserves*:
$\mathfrak{H} \circ_{CF-NTCF} \varepsilon : (\mathfrak{H} \circ_{CF} \mathfrak{G}) \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{H} \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

syntax *-is-cat-rKe-preserves* ::
$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(
‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe}$ı - :/ - $\mapsto_C$ - $\mapsto_C$ - : - $\mapsto\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
syntax-consts *-is-cat-rKe-preserves* ⇌ *is-cat-rKe-preserves*
translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_C \mathfrak{D})$ ⇌
CONST is-cat-rKe-preserves $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ \mathfrak{H}\ \varepsilon$

locale *is-cat-lKe-preserves* =
is-cat-lKe $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ \eta$ + *is-functor* $\alpha\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{H}$
for $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ \mathfrak{H}\ \eta$ +
assumes *cat-lKe-preserves*:
$\mathfrak{H} \circ_{CF-NTCF} \eta : \mathfrak{H} \circ_{CF} \mathfrak{T} \mapsto_{CF.lKe\alpha} (\mathfrak{H} \circ_{CF} \mathfrak{F}) \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

syntax *-is-cat-lKe-preserves* ::
$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(
‹(- :/ - $\mapsto_{CF.lKe}$ı - $\circ_{CF}$ - :/ - $\mapsto_C$ - $\mapsto_C$ - : - $\mapsto\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
syntax-consts *-is-cat-lKe-preserves* ⇌ *is-cat-lKe-preserves*
translations $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_C \mathfrak{D})$ ⇌
CONST is-cat-lKe-preserves $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{D}\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ \mathfrak{H}\ \eta$

Rules.

lemma (**in** *is-cat-rKe-preserves*) *is-cat-rKe-preserves-axioms'*:
assumes $\alpha' = \alpha$
and $\mathfrak{G}' = \mathfrak{G}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{H}' = \mathfrak{H}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe_{\alpha'}} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$

⟨*proof*⟩

mk-ide rf *is-cat-rKe-preserves-def*[*unfolded is-cat-rKe-preserves-axioms-def*]
|*intro is-cat-rKe-preservesI*|
|*dest is-cat-rKe-preservesD*[*dest*]|
|*elim is-cat-rKe-preservesE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-rKeD*(*1–3*)

lemma (**in** *is-cat-lKe-preserves*) *is-cat-lKe-preserves-axioms'*:
assumes $\alpha' = \alpha$
and $\mathfrak{F}' = \mathfrak{F}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{H}' = \mathfrak{H}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows $\eta : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{F}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$
⟨*proof*⟩

mk-ide rf *is-cat-lKe-preserves-def*[*unfolded is-cat-lKe-preserves-axioms-def*]
|*intro is-cat-lKe-preservesI*|
|*dest is-cat-lKe-preservesD*[*dest*]|
|*elim is-cat-lKe-preservesE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-lKe-preservesD*(*1–3*)

Duality.

lemma (**in** *is-cat-rKe-preserves*) *is-cat-rKe-preserves-op*:
op-ntcf ε :
op-cf $\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *op-cf* $\mathfrak{G} \circ_{CF}$ *op-cf* $\mathfrak{K}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C$ (*op-cf* $\mathfrak{H}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_C$ *op-cat* $\mathfrak{D}$)
⟨*proof*⟩

lemma (**in** *is-cat-rKe-preserves*) *is-cat-lKe-preserves-op'*[*cat-op-intros*]:
assumes $\mathfrak{T}'$ = *op-cf* $\mathfrak{T}$
and $\mathfrak{G}'$ = *op-cf* $\mathfrak{G}$
and $\mathfrak{K}'$ = *op-cf* $\mathfrak{K}$
and $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
and $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$
and $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
and $\mathfrak{D}'$ = *op-cat* $\mathfrak{D}$
and $\mathfrak{H}'$ = *op-cf* $\mathfrak{H}$
shows *op-ntcf* ε :
$\mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-rKe-preserves.is-cat-lKe-preserves-op'*

lemma (**in** *is-cat-lKe-preserves*) *is-cat-rKe-preserves-op*:
op-ntcf η :
op-cf $\mathfrak{F} \circ_{CF}$ *op-cf* $\mathfrak{K} \mapsto_{CF.rKe\alpha}$ *op-cf* $\mathfrak{T}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C$ (*op-cf* $\mathfrak{H}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_C$ *op-cat* $\mathfrak{D}$)
⟨*proof*⟩

lemma (**in** *is-cat-lKe-preserves*) *is-cat-rKe-preserves-op'*[*cat-op-intros*]:

assumes $\mathfrak{T}' = $ *op-cf* $\mathfrak{T}$
and $\mathfrak{F}' = $ *op-cf* $\mathfrak{F}$
and $\mathfrak{K}' = $ *op-cf* $\mathfrak{K}$
and $\mathfrak{H}' = $ *op-cf* $\mathfrak{H}$
and $\mathfrak{B}' = $ *op-cat* $\mathfrak{B}$
and $\mathfrak{A}' = $ *op-cat* $\mathfrak{A}$
and $\mathfrak{C}' = $ *op-cat* $\mathfrak{C}$
and $\mathfrak{D}' = $ *op-cat* $\mathfrak{D}$
shows *op-ntcf* η :
$\mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$
⟨*proof*⟩

14.6 All concepts are Kan extensions

Background information for this subsection is provided in Chapter X-7 in [9] and subsection 6.5 in [14]. It should be noted that only the connections between the Kan extensions, limits and adjunctions are exposed (an alternative proof of the Yoneda lemma using Kan extensions is not provided in the context of this work).

14.6.1 Limits and colimits

lemma *cat-rKe-is-cat-limit*:
— The statement of the theorem is similar to the statement of a part of Theorem 1 in Chapter X-7 in [9] or Proposition 6.5.1 in [14].
assumes $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\varepsilon : \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{a}|\!) <_{CF.lim} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cat-lKe-is-cat-colimit*:
assumes $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\eta : \mathfrak{T} >_{CF.colim} \mathfrak{F}(\!|ObjMap|\!)(\!|\mathfrak{a}|\!) : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cat-limit-is-rKe*:
— The statement of the theorem is similar to the statement of a part of Theorem 1 in Chapter X-7 in [9] or Proposition 6.5.1 in [14].
assumes $\varepsilon : \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{a}|\!) <_{CF.lim} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-1* $\mathfrak{a}\ \mathfrak{f}$
and $\mathfrak{G} :$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma *cat-colimit-is-lKe*:
assumes $\eta : \mathfrak{T} >_{CF.colim} \mathfrak{F}(\!|ObjMap|\!)(\!|\mathfrak{a}|\!) : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-1* $\mathfrak{a}\ \mathfrak{f}$
and $\mathfrak{F} :$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C$ *cat-1* $\mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

14.6.2 Adjoints

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-is-rKe*:
— The statement of the theorem is similar to the statement of a part of Theorem 2 in Chapter X-7 in [9] or Proposition 6.5.2 in [14]. The proof follows (approximately) the proof in [14].
shows $\varepsilon_C\ \Phi : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF.rKe\alpha}$ *cf-id* $\mathfrak{D} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

$\langle proof \rangle$

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-is-lKe*:
shows $\eta_C\ \Phi$: *cf-id* $\mathfrak{C} \mapsto_{CF.lKe\alpha} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_C \mathfrak{D} \mapsto_C \mathfrak{C}$
$\langle proof \rangle$

lemma *cf-adjunction-if-lKe-preserves*:
— The statement of the theorem is similar to the statement of a part of Theorem 2 in Chapter X-7 in [9] or Proposition 6.5.2 in [14].
assumes η : *cf-id* $\mathfrak{D} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{G} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_C \mathfrak{C})$
shows *cf-adjunction-of-unit* $\alpha\ \mathfrak{G}\ \mathfrak{F}\ \eta : \mathfrak{G} \rightleftharpoons_{CF} \mathfrak{F} : \mathfrak{D} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{C}$
$\langle proof \rangle$

lemma *cf-adjunction-if-rKe-preserves*:
assumes $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF.rKe\alpha}$ *cf-id* $\mathfrak{D} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_C \mathfrak{C})$
shows *cf-adjunction-of-counit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
$\langle proof \rangle$

15 Pointwise Kan extensions

15.1 Pointwise Kan extensions

The following subsection is based on elements of the content of section 6.3 in [14] and Chapter X-5 in [9].

locale *is-cat-pw-rKe* = *is-cat-rKe* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε
for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε +
assumes *cat-pw-rKe-preserved*: $a \in_\circ \mathfrak{A}(\!|Obj|\!) \Longrightarrow$
ε :
$\mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T}$:
$\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (Hom_{O.C\alpha}\mathfrak{A}(a,-) : \mathfrak{A} \mapsto\mapsto_C \textit{cat-Set}\ \alpha)$

syntax *-is-cat-pw-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(
‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe.pw}$1 - :/ - $\mapsto_C$ - $\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
syntax-consts *-is-cat-pw-rKe* ⇌ *is-cat-pw-rKe*
translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe.pw\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$ ⇌
CONST is-cat-pw-rKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε

locale *is-cat-pw-lKe* = *is-cat-lKe* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η
for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η +
assumes *cat-pw-lKe-preserved*: $a \in_\circ$ *op-cat* $\mathfrak{A}(\!|Obj|\!) \Longrightarrow$
op-ntcf η :
op-cf $\mathfrak{F} \circ_{CF}$ *op-cf* $\mathfrak{K} \mapsto_{CF.rKe\alpha}$ *op-cf* $\mathfrak{T}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C (Hom_{O.C\alpha}\mathfrak{A}(-,a)$: *op-cat* $\mathfrak{A} \mapsto\mapsto_C$ *cat-Set* $\alpha)$

syntax *-is-cat-pw-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(
‹(- :/ - $\mapsto_{CF.lKe.pw}$1 - $\circ_{CF}$ - :/ - $\mapsto_C$ - $\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
syntax-consts *-is-cat-pw-lKe* ⇌ *is-cat-pw-lKe*
translations $\eta : \mathfrak{T} \mapsto_{CF.lKe.pw\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$ ⇌
CONST is-cat-pw-lKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η

lemma (**in** *is-cat-pw-rKe*) *cat-pw-rKe-preserved'*[*cat-Kan-cs-intros*]:
assumes $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{H}' = Hom_{O.C\alpha}\mathfrak{A}(a,-)$
and $\mathfrak{E}'$ = *cat-Set* α
shows $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{E}')$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = *is-cat-pw-rKe.cat-pw-rKe-preserved'*

lemma (**in** *is-cat-pw-lKe*) *cat-pw-lKe-preserved'*[*cat-Kan-cs-intros*]:
assumes $a \in_\circ$ *op-cat* $\mathfrak{A}(\!|Obj|\!)$
and $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$
and $\mathfrak{K}'$ = *op-cf* $\mathfrak{K}$
and $\mathfrak{T}'$ = *op-cf* $\mathfrak{T}$
and $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
and $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
and $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$
and $\mathfrak{H}' = Hom_{O.C\alpha}\mathfrak{A}(-,a)$
and $\mathfrak{E}'$ = *cat-Set* α

shows *op-ntcf* η :
$\mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{E}')$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = *is-cat-pw-lKe.cat-pw-lKe-preserved'*

Rules.

lemma (**in** *is-cat-pw-rKe*) *is-cat-pw-rKe-axioms'*[*cat-Kan-cs-intros*]:
assumes $\alpha' = \alpha$
and $\mathfrak{G}' = \mathfrak{G}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe.pw_{\alpha'}} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
⟨*proof*⟩

mk-ide rf *is-cat-pw-rKe-def*[*unfolded is-cat-pw-rKe-axioms-def*]
|*intro is-cat-pw-rKeI*|
|*dest is-cat-pw-rKeD*[*dest*]|
|*elim is-cat-pw-rKeE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-pw-rKeD*(*1*)

lemma (**in** *is-cat-pw-lKe*) *is-cat-pw-lKe-axioms'*[*cat-Kan-cs-intros*]:
assumes $\alpha' = \alpha$
and $\mathfrak{F}' = \mathfrak{F}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\eta : \mathfrak{T}' \mapsto_{CF.lKe.pw_{\alpha'}} \mathfrak{F}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
⟨*proof*⟩

mk-ide rf *is-cat-pw-lKe-def*[*unfolded is-cat-pw-lKe-axioms-def*]
|*intro is-cat-pw-lKeI*|
|*dest is-cat-pw-lKeD*[*dest*]|
|*elim is-cat-pw-lKeE*[*elim*]|

lemmas [*cat-Kan-cs-intros*] = *is-cat-pw-lKeD*(*1*)

Duality.

lemma (**in** *is-cat-pw-rKe*) *is-cat-pw-lKe-op*:
op-ntcf ε :
op-cf $\mathfrak{T} \mapsto_{CF.lKe.pw\alpha}$ *op-cf* $\mathfrak{G} \circ_{CF}$ *op-cf* $\mathfrak{K}$:
op-cat $\mathfrak{B} \mapsto_C$ *op-cat* $\mathfrak{C} \mapsto_C$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-pw-rKe*) *is-cat-pw-lKe-op'*[*cat-op-intros*]:
assumes $\mathfrak{T}' =$ *op-cf* $\mathfrak{T}$
and $\mathfrak{G}' =$ *op-cf* $\mathfrak{G}$
and $\mathfrak{K}' =$ *op-cf* $\mathfrak{K}$
and $\mathfrak{B}' =$ *op-cat* $\mathfrak{B}$
and $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$
and $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$
shows *op-ntcf* $\varepsilon : \mathfrak{T}' \mapsto_{CF.lKe.pw\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$

⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-pw-rKe.is-cat-pw-lKe-op′*

lemma (**in** *is-cat-pw-lKe*) *is-cat-pw-rKe-op*:
op-ntcf η :
op-cf $\mathfrak{F}$ $\circ_{CF}$ *op-cf* $\mathfrak{K}$ $\mapsto_{CF.rKe.pw\alpha}$ *op-cf* $\mathfrak{T}$:
op-cat $\mathfrak{B}$ $\mapsto_C$ *op-cat* $\mathfrak{C}$ $\mapsto_C$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-pw-lKe*) *is-cat-pw-lKe-op′*[*cat-op-intros*]:
assumes $\mathfrak{T}'$ = *op-cf* $\mathfrak{T}$
and $\mathfrak{F}'$ = *op-cf* $\mathfrak{F}$
and $\mathfrak{K}'$ = *op-cf* $\mathfrak{K}$
and $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
and $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$
and $\mathfrak{C}'$ = *op-cat* $\mathfrak{C}$
shows *op-ntcf* η : $\mathfrak{F}'$ $\circ_{CF}$ $\mathfrak{K}'$ $\mapsto_{CF.rKe.pw\alpha}$ $\mathfrak{T}'$: $\mathfrak{B}'$ $\mapsto_C$ $\mathfrak{C}'$ $\mapsto_C$ $\mathfrak{A}'$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *is-cat-pw-lKe.is-cat-pw-lKe-op′*

15.2 Lemma X.5: *L-10-5-N*

This subsection and several further subsections (15.2-15.8) expose definitions that are used in the proof of the technical lemma that was used in the proof of Theorem 3 from Chapter X-5 in [9].

definition *L-10-5-N* :: *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V*
where *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ *c* =
[
(
$\lambda a \in_\circ \mathfrak{T}(|HomCod|)(|Obj|)$.
cf-nt α β $\mathfrak{K}(|ObjMap|)(|$*cf-map* ($Hom_{O.C\alpha}\mathfrak{T}(|HomCod|)(a,-)$ $\circ_{CF}$ $\mathfrak{T}$), *c*$|)_\bullet$
),
(
$\lambda f \in_\circ \mathfrak{T}(|HomCod|)(|Arr|)$.
cf-nt α β $\mathfrak{K}(|ArrMap|)(|$
ntcf-arrow ($Hom_{A.C\alpha}\mathfrak{T}(|HomCod|)(f,-)$ $\circ_{NTCF-CF}$ $\mathfrak{T}$), $\mathfrak{K}(|HomCod|)(|CId|)(|c|)$
$|)_\bullet$
),
op-cat ($\mathfrak{T}(|HomCod|)$),
cat-Set β
$]_\circ$

Components.

lemma *L-10-5-N-components*:
shows *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ $c(|ObjMap|)$ =
(
$\lambda a \in_\circ \mathfrak{T}(|HomCod|)(|Obj|)$.
cf-nt α β $\mathfrak{K}(|ObjMap|)(|$*cf-map* ($Hom_{O.C\alpha}\mathfrak{T}(|HomCod|)(a,-)$ $\circ_{CF}$ $\mathfrak{T}$), *c*$|)_\bullet$
)
and *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ $c(|ArrMap|)$ =
(
$\lambda f \in_\circ \mathfrak{T}(|HomCod|)(|Arr|)$.
cf-nt α β $\mathfrak{K}(|ArrMap|)(|$
ntcf-arrow ($Hom_{A.C\alpha}\mathfrak{T}(|HomCod|)(f,-)$ $\circ_{NTCF-CF}$ $\mathfrak{T}$), $\mathfrak{K}(|HomCod|)(|CId|)(|c|)$
$|)_\bullet$

)
and *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ $c(|HomDom|)$ = *op-cat* ($\mathfrak{T}(|HomCod|)$)
and *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ $c(|HomCod|)$ = *cat-Set* β
⟨*proof*⟩

context
fixes α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *L-10-5-N-components′* = *L-10-5-N-components*[
where $\mathfrak{T}$=$\mathfrak{T}$ **and** $\mathfrak{K}$=$\mathfrak{K}$, *unfolded cat-cs-simps*
]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-N-components′*(*3*,*4*)

end

15.2.1 Object map

mk-VLambda *L-10-5-N-components*(*1*)
|*vsv L-10-5-N-ObjMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ c
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

mk-VLambda *L-10-5-N-components′*(*1*)[*OF* $\mathfrak{K}$ $\mathfrak{T}$]
|*vdomain L-10-5-N-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app L-10-5-N-ObjMap-app*[*cat-Kan-cs-simps*]|

end

15.2.2 Arrow map

mk-VLambda *L-10-5-N-components*(*2*)
|*vsv L-10-5-N-ArrMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ c
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

mk-VLambda *L-10-5-N-components′*(*2*)[*OF* $\mathfrak{K}$ $\mathfrak{T}$]
|*vdomain L-10-5-N-ArrMap-vdomain*[*cat-Kan-cs-simps*]|
|*app L-10-5-N-ArrMap-app*[*cat-Kan-cs-simps*]|

end

15.2.3 *L-10-5-N* is a functor

lemma *L-10-5-N-is-functor*:

assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ c : *op-cat* $\mathfrak{A} \mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

lemma *L-10-5-N-is-functor*′[*cat-Kan-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$
and $\mathfrak{B}' =$ *cat-Set* β
and $\beta' = \beta$
shows *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ c : $\mathfrak{A}' \mapsto\mapsto_{C\beta'} \mathfrak{B}'$
⟨*proof*⟩

15.3 Lemma X.5: *L-10-5-υ-arrow*

15.3.1 Definition and elementary properties

definition *L-10-5-υ-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ a b =
[
($\lambda f \in_\circ$ *Hom* ($\mathfrak{K}(\!|HomCod|\!)$) c ($\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!)$). $\tau(\!|NTMap|\!)(\!|0, b, f|\!)_\bullet$),
Hom ($\mathfrak{K}(\!|HomCod|\!)$) c ($\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!)$),
Hom ($\mathfrak{T}(\!|HomCod|\!)$) a ($\mathfrak{T}(\!|ObjMap|\!)(\!|b|\!)$)
]$_\circ$

Components.

lemma *L-10-5-υ-arrow-components*:
shows *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ a $b(\!|ArrVal|\!)$ =
($\lambda f \in_\circ$ *Hom* ($\mathfrak{K}(\!|HomCod|\!)$) c ($\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!)$). $\tau(\!|NTMap|\!)(\!|0, b, f|\!)_\bullet$)
and *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ a $b(\!|ArrDom|\!)$ = *Hom* ($\mathfrak{K}(\!|HomCod|\!)$) c ($\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!)$)
and *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ a $b(\!|ArrCod|\!)$ = *Hom* ($\mathfrak{T}(\!|HomCod|\!)$) a ($\mathfrak{T}(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

context
fixes α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *L-10-5-υ-arrow-components*′ = *L-10-5-υ-arrow-components*[
where $\mathfrak{T}$=$\mathfrak{T}$ **and** $\mathfrak{K}$=$\mathfrak{K}$, *unfolded cat-cs-simps*
]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-υ-arrow-components*′(*2*,*3*)

end

15.3.2 Arrow value

mk-VLambda *L-10-5-υ-arrow-components*(*1*)
  |*vsv L-10-5-υ-arrow-ArrVal-vsv*[*cat-Kan-cs-intros*]|

context
  fixes α 𝔅 ℭ 𝔄 𝔎 𝔗
  assumes 𝔎: 𝔎 : 𝔅 $\mapsto\mapsto_{C\alpha}$ ℭ
    and 𝔗: 𝔗 : 𝔅 $\mapsto\mapsto_{C\alpha}$ 𝔄
begin

mk-VLambda *L-10-5-υ-arrow-components'*(*1*)[*OF* 𝔎 𝔗]
  |*vdomain L-10-5-υ-arrow-ArrVal-vdomain*[*cat-Kan-cs-simps*]|
  |*app L-10-5-υ-arrow-ArrVal-app*[*unfolded in-Hom-iff*]|

end

lemma *L-10-5-υ-arrow-ArrVal-app'*:
  assumes 𝔎 : 𝔅 $\mapsto\mapsto_{C\alpha}$ ℭ
    and 𝔗 : 𝔅 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and f : c $\mapsto_{ℭ}$ 𝔎⦇*ObjMap*⦈⦇b⦈
  shows *L-10-5-υ-arrow* 𝔗 𝔎 c τ a b⦇*ArrVal*⦈⦇f⦈ = τ⦇*NTMap*⦈⦇*0*, b, f⦈•
⟨*proof*⟩

15.3.3 *L-10-5-υ-arrow* is an arrow

lemma *L-10-5-υ-arrow-ArrVal-is-arr*:
  assumes 𝔎 : 𝔅 $\mapsto\mapsto_{C\alpha}$ ℭ
    and 𝔗 : 𝔅 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and τ' = *ntcf-arrow* τ
    and τ : a $<_{CF.cone}$ 𝔗 $\circ_{CF}$ c $_O\sqcap_{CF}$ 𝔎 : c $\downarrow_{CF}$ 𝔎 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and f : c $\mapsto_{ℭ}$ 𝔎⦇*ObjMap*⦈⦇b⦈
    and b $\in_\circ$ 𝔅⦇*Obj*⦈
  shows *L-10-5-υ-arrow* 𝔗 𝔎 c τ' a b⦇*ArrVal*⦈⦇f⦈ : a $\mapsto_{𝔄}$ 𝔗⦇*ObjMap*⦈⦇b⦈
⟨*proof*⟩

lemma *L-10-5-υ-arrow-ArrVal-is-arr'*[*cat-Kan-cs-intros*]:
  assumes 𝔎 : 𝔅 $\mapsto\mapsto_{C\alpha}$ ℭ
    and 𝔗 : 𝔅 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and τ' = *ntcf-arrow* τ
    and a' = a
    and b' = 𝔗⦇*ObjMap*⦈⦇b⦈
    and 𝔄' = 𝔄
    and τ : a $<_{CF.cone}$ 𝔗 $\circ_{CF}$ c $_O\sqcap_{CF}$ 𝔎 : c $\downarrow_{CF}$ 𝔎 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and f : c $\mapsto_{ℭ}$ 𝔎⦇*ObjMap*⦈⦇b⦈
    and b $\in_\circ$ 𝔅⦇*Obj*⦈
  shows *L-10-5-υ-arrow* 𝔗 𝔎 c τ' a b⦇*ArrVal*⦈⦇f⦈ : a' $\mapsto_{𝔄}$ b'
  ⟨*proof*⟩

15.3.4 Further properties

lemma *L-10-5-υ-arrow-is-arr*:
  assumes 𝔎 : 𝔅 $\mapsto\mapsto_{C\alpha}$ ℭ
    and 𝔗 : 𝔅 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and c $\in_\circ$ ℭ⦇*Obj*⦈
    and τ' = *ntcf-arrow* τ
    and τ : a $<_{CF.cone}$ 𝔗 $\circ_{CF}$ c $_O\sqcap_{CF}$ 𝔎 : c $\downarrow_{CF}$ 𝔎 $\mapsto\mapsto_{C\alpha}$ 𝔄
    and b $\in_\circ$ 𝔅⦇*Obj*⦈
  shows *L-10-5-υ-arrow* 𝔗 𝔎 c τ' a b :

$Hom\ \mathfrak{C}\ c\ (\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!)) \mapsto_{cat\text{-}Set\ \alpha} Hom\ \mathfrak{A}\ a\ (\mathfrak{T}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

lemma *L-10-5-υ-arrow-is-arr'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $A = Hom\ \mathfrak{C}\ c\ (\mathfrak{K}(\!|ObjMap|\!)(\!|b|\!))$
and $B = Hom\ \mathfrak{A}\ a\ (\mathfrak{T}(\!|ObjMap|\!)(\!|b|\!))$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b : A \mapsto_{\mathfrak{C}'} B$
⟨*proof*⟩

lemma *L-10-5-υ-cf-hom*[*cat-Kan-cs-simps*]:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $f : a' \mapsto_{\mathfrak{B}} b'$
shows
$L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b' \circ_{A\,cat\text{-}Set\ \alpha}$
$cf\text{-}hom\ \mathfrak{C}\ [\mathfrak{C}(\!|CId|\!)(\!|c|\!), \mathfrak{K}(\!|ArrMap|\!)(\!|f|\!)]_\circ =$
$cf\text{-}hom\ \mathfrak{A}\ [\mathfrak{A}(\!|CId|\!)(\!|a|\!), \mathfrak{T}(\!|ArrMap|\!)(\!|f|\!)]_\circ \circ_{A\,cat\text{-}Set\ \alpha}$
$L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ a'$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

15.4 Lemma X.5: *L-10-5-τ*

15.4.1 Definition and elementary properties

definition *L-10-5-τ* **where** $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a =$
[
$(\lambda bf \in_\circ c \downarrow_{CF} \mathfrak{K}(\!|Obj|\!).\ \upsilon(\!|NTMap|\!)(\!|bf(\!|1_{\mathbb{N}}|\!)|\!)(\!|ArrVal|\!)(\!|bf(\!|2_{\mathbb{N}}|\!)|\!)),$
$cf\text{-}const\ (c \downarrow_{CF} \mathfrak{K})\ (\mathfrak{T}(\!|HomCod|\!))\ a,$
$\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K},$
$c \downarrow_{CF} \mathfrak{K},$
$(\mathfrak{T}(\!|HomCod|\!))$
$]_\circ$

Components.

lemma *L-10-5-τ-components*:
shows $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a(\!|NTMap|\!) =$
$(\lambda bf \in_\circ c \downarrow_{CF} \mathfrak{K}(\!|Obj|\!).\ \upsilon(\!|NTMap|\!)(\!|bf(\!|1_{\mathbb{N}}|\!)|\!)(\!|ArrVal|\!)(\!|bf(\!|2_{\mathbb{N}}|\!)|\!))$
and $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a(\!|NTDom|\!) = cf\text{-}const\ (c \downarrow_{CF} \mathfrak{K})\ (\mathfrak{T}(\!|HomCod|\!))\ a$
and $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a(\!|NTCod|\!) = \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K}$
and $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a(\!|NTDGDom|\!) = c \downarrow_{CF} \mathfrak{K}$
and $L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a(\!|NTDGCod|\!) = (\mathfrak{T}(\!|HomCod|\!))$
⟨*proof*⟩

context
fixes $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

  and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *L-10-5-*τ*-components*$'$ = *L-10-5-*τ*-components*[
 where $\mathfrak{T}$=$\mathfrak{T}$ **and** $\mathfrak{K}$=$\mathfrak{K}$, *unfolded cat-cs-simps*
 ]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-*τ*-components*$'$(*2–5*)

end

15.4.2 Natural transformation map

mk-VLambda *L-10-5-*τ*-components*(*1*)
 |*vsv L-10-5-*τ*-NTMap-vsv*[*cat-Kan-cs-intros*]|
 |*vdomain L-10-5-*τ*-NTMap-vdomain*[*cat-Kan-cs-simps*]|

lemma *L-10-5-*τ*-NTMap-app*[*cat-Kan-cs-simps*]:
 assumes $bf = [0, b, f]_\circ$ **and** $bf \in_\circ c \downarrow_{CF} \mathfrak{K}(\!|Obj|\!)$
 shows *L-10-5-*τ $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTMap|\!)(\!|bf|\!) = \upsilon(\!|NTMap|\!)(\!|b|\!)(\!|ArrVal|\!)(\!|f|\!)$
 ⟨*proof*⟩

15.4.3 *L-10-5-*τ is a cone

lemma *L-10-5-*τ*-is-cat-cone*[*cat-cs-intros*]:
 assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
  and υ'*-def*: $\upsilon' =$ *ntcf-arrow* υ
  and υ: υ :
   $Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  and a: $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
 shows *L-10-5-*τ $\mathfrak{T}$ $\mathfrak{K}$ c υ' $a : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *L-10-5-*τ*-is-cat-cone*$'$[*cat-Kan-cs-intros*]:
 assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
  and $\upsilon' =$ *ntcf-arrow* υ
  and $\mathfrak{F}' = \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K}$
  and $c\mathfrak{K} = c \downarrow_{CF} \mathfrak{K}$
  and $\mathfrak{A}' = \mathfrak{A}$
  and $\alpha' = \alpha$
  and υ :
   $Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$:
   $\mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
  and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
 shows *L-10-5-*τ $\mathfrak{T}$ $\mathfrak{K}$ c υ' $a : a <_{CF.cone} \mathfrak{F}' : c\mathfrak{K} \mapsto\mapsto_{C_{\alpha'}} \mathfrak{A}'$
 ⟨*proof*⟩

15.5 Lemma X.5: *L-10-5-*υ

15.5.1 Definition and elementary properties

definition *L-10-5-*υ :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a$ =
[
($\lambda b \in_\circ \mathfrak{T}(|HomDom|)(|Obj|)$. $L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a\ b$),
$Hom_{O.C\alpha}\mathfrak{K}(|HomCod|)(c,-) \circ_{CF} \mathfrak{K}$,
$Hom_{O.C\alpha}\mathfrak{T}(|HomCod|)(a,-) \circ_{CF} \mathfrak{T}$,
$\mathfrak{T}(|HomDom|)$,
cat-Set α
]$_\circ$

Components.

lemma *L-10-5-υ-components*:
shows $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(|NTMap|)$ =
($\lambda b \in_\circ \mathfrak{T}(|HomDom|)(|Obj|)$. $L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a\ b$)
and $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(|NTDom|) = Hom_{O.C\alpha}\mathfrak{K}(|HomCod|)(c,-) \circ_{CF} \mathfrak{K}$
and $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(|NTCod|) = Hom_{O.C\alpha}\mathfrak{T}(|HomCod|)(a,-) \circ_{CF} \mathfrak{T}$
and $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(|NTDGDom|) = \mathfrak{T}(|HomDom|)$
and $L\text{-}10\text{-}5\text{-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(|NTDGCod|)$ = *cat-Set* α
⟨*proof*⟩

context
fixes $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{A}\ \mathfrak{T}$ ⟨*proof*⟩

lemmas *L-10-5-υ-components′* = *L-10-5-υ-components*[
where $\mathfrak{T}$=$\mathfrak{T}$ **and** $\mathfrak{K}$=$\mathfrak{K}$, *unfolded cat-cs-simps*
]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-υ-components′*(*2–5*)

end

15.5.2 Natural transformation map

mk-VLambda *L-10-5-υ-components*(*1*)
|*vsv L-10-5-υ-NTMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{K}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{A}\ \mathfrak{T}$ ⟨*proof*⟩

mk-VLambda *L-10-5-υ-components′*(*1*)[*OF* $\mathfrak{K}\ \mathfrak{T}$]
|*vdomain L-10-5-υ-NTMap-vdomain*[*cat-Kan-cs-simps*]|
|*app L-10-5-υ-NTMap-app*[*cat-Kan-cs-simps*]|

end

15.5.3 *L-10-5-υ* is a natural transformation

lemma *L-10-5-υ-is-ntcf*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and $c \in_\circ \mathfrak{C}(|Obj|)$
  and τ'-*def*: $\tau' = \textit{ntcf-arrow}\ \tau$
  and τ: $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and a: $a \in_\circ \mathfrak{A}(|Obj|)$
shows $\textit{L-10-5-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a :$
  $Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
  (**is** ‹$?\textit{L-10-5-}\upsilon : ?H\text{-}\mathfrak{C}\ c \circ_{CF} \mathfrak{K} \mapsto_{CF} ?H\text{-}\mathfrak{A}\ a \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$›)
⟨*proof*⟩

lemma *L-10-5-υ-is-ntcf'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and $c \in_\circ \mathfrak{C}(|Obj|)$
  and $\tau' = \textit{ntcf-arrow}\ \tau$
  and $\mathfrak{F}' = Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K}$
  and $\mathfrak{G}' = Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$
  and $\mathfrak{B}' = \mathfrak{B}$
  and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
  and $\alpha' = \alpha$
  and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
  and $a \in_\circ \mathfrak{A}(|Obj|)$
shows $\textit{L-10-5-}\upsilon\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B}' \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*proof*⟩

15.6 Lemma X.5: *L-10-5-χ-arrow*

15.6.1 Definition and elementary properties

definition *L-10-5-χ-arrow*
where $\textit{L-10-5-}\chi\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a =$
  [
    $(\lambda\upsilon\in_\circ \textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(|ObjMap|)(|a|).\ \textit{ntcf-arrow}\ (\textit{L-10-5-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a)),$
    $\textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(|ObjMap|)(|a|),$
    $\textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K})(|ObjMap|)(|a|)$
  $]_\circ$

Components.

lemma *L-10-5-χ-arrow-components*:
shows $\textit{L-10-5-}\chi\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a(|ArrVal|) =$
    $(\lambda\upsilon\in_\circ \textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(|ObjMap|)(|a|).\ \textit{ntcf-arrow}\ (\textit{L-10-5-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a))$
  and $\textit{L-10-5-}\chi\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a(|ArrDom|) = \textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(|ObjMap|)(|a|)$
  and $\textit{L-10-5-}\chi\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a(|ArrCod|) =$
    $\textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K})(|ObjMap|)(|a|)$
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *L-10-5-χ-arrow-components*(*2*,*3*)

15.6.2 Arrow value

mk-VLambda *L-10-5-χ-arrow-components*(*1*)
  |*vsv L-10-5-χ-arrow-vsv*[*cat-Kan-cs-intros*]|
  |*vdomain L-10-5-χ-arrow-vdomain*|
  |*app L-10-5-χ-arrow-app*|

lemma *L-10-5-χ-arrow-vdomain'*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}\ \beta$
  and $\alpha \in_\circ \beta$

and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(|Obj|)$
and $a \in_\circ \mathfrak{A}(|Obj|)$
shows $\mathcal{D}_\circ$ (*L-10-5-χ-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c $a(|ArrVal|)$) = *Hom*
(*cat-FUNCT* α $\mathfrak{B}$ (*cat-Set* α))
(*cf-map* ($Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K}$))
(*cf-map* ($Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$))
⟨*proof*⟩

lemma *L-10-5-χ-arrow-app'*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(|Obj|)$
and *υ'-def*: υ' = *ntcf-arrow* υ
and *υ*: υ :
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and *a*: $a \in_\circ \mathfrak{A}(|Obj|)$
shows
L-10-5-χ-arrow α β $\mathfrak{T}$ $\mathfrak{K}$ c $a(|ArrVal|)(|\upsilon'|)$ =
ntcf-arrow (*L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ' a)
⟨*proof*⟩

lemma *υτa-def*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(|Obj|)$
and *υτa'-def*: $\upsilon\tau a'$ = *ntcf-arrow* $\upsilon\tau a$
and *υτa*: $\upsilon\tau a$:
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$:
$\mathfrak{B} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and *a*: $a \in_\circ \mathfrak{A}(|Obj|)$
shows $\upsilon\tau a$ = *L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ c (*ntcf-arrow* (*L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c $\upsilon\tau a'$ a)) a
(**is** ‹$\upsilon\tau a$ = *?L-10-5-υ* (*ntcf-arrow ?L-10-5-τ*) *a*›)
⟨*proof*⟩

15.7 Lemma X.5: *L-10-5-χ'-arrow*

15.7.1 Definition and elementary properties

definition *L-10-5-χ'-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *L-10-5-χ'-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a =
[
(
$\lambda\tau\in_\circ$*cf-Cone* α β ($\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K}$)$(|ObjMap|)(|a|)$.
ntcf-arrow (*L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ c τ a)
),
cf-Cone α β ($\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K}$)$(|ObjMap|)(|a|)$,
L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ $c(|ObjMap|)(|a|)$
]$_\circ$

Components.

lemma *L-10-5-χ'-arrow-components*:
shows *L-10-5-χ'-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c $a(|ArrVal|)$ =
(
$\lambda\tau\in_\circ$*cf-Cone* α β ($\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF} \mathfrak{K}$)$(|ObjMap|)(|a|)$.

ntcf-arrow (*L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ c τ a)
)
and [*cat-Kan-cs-simps*]: *L-10-5-χ′-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a⦇*ArrDom*⦈ =
cf-Cone α β ($\mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K}$)⦇*ObjMap*⦈⦇$a$⦈
and [*cat-Kan-cs-simps*]: *L-10-5-χ′-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a⦇*ArrCod*⦈ =
L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ c⦇*ObjMap*⦈⦇a⦈
⟨*proof*⟩

15.7.2 Arrow value

mk-VLambda *L-10-5-χ′-arrow-components(1)*
|*vsv L-10-5-χ′-arrow-ArrVal-vsv*[*cat-Kan-cs-intros*]|
|*vdomain L-10-5-χ′-arrow-ArrVal-vdomain*|
|*app L-10-5-χ′-arrow-ArrVal-app*|

lemma *L-10-5-χ′-arrow-ArrVal-vdomain′*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and τ: $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and a: $a \in_\circ \mathfrak{A}$⦇*Obj*⦈
shows $\mathcal{D}_\circ$ (*L-10-5-χ′-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a⦇*ArrVal*⦈) = *Hom*
(*cat-FUNCT* α ($c \downarrow_{CF} \mathfrak{K}$) $\mathfrak{A}$)
(*cf-map* (*cf-const* ($c \downarrow_{CF} \mathfrak{K}$) $\mathfrak{A}$ a))
(*cf-map* ($\mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K}$))
⟨*proof*⟩

lemma *L-10-5-χ′-arrow-ArrVal-app′*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and τ'-*def*: τ' = *ntcf-arrow* τ
and τ: $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and a: $a \in_\circ \mathfrak{A}$⦇*Obj*⦈
shows *L-10-5-χ′-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a⦇*ArrVal*⦈⦇τ'⦈ =
ntcf-arrow (*L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ c τ' a)
⟨*proof*⟩

15.7.3 *L-10-5-χ′-arrow* is an isomorphism in the category *Set*

lemma *L-10-5-χ′-arrow-is-iso-arr*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}$⦇*Obj*⦈
and $a \in_\circ \mathfrak{A}$⦇*Obj*⦈
shows *L-10-5-χ′-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ c a :
cf-Cone α β ($\mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K}$)⦇*ObjMap*⦈⦇$a$⦈ $\mapsto_{iso}$*cat-Set* β
L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ c⦇*ObjMap*⦈⦇a⦈
(
is
‹
?L-10-5-χ′-arrow :
cf-Cone α β ($\mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K}$)⦇*ObjMap*⦈⦇$a$⦈ $\mapsto_{iso}$*cat-Set* β
?L-10-5-N⦇*ObjMap*⦈⦇a⦈
›
)
⟨*proof*⟩

lemma *L-10-5-*χ'*-arrow-is-iso-arr*$'$[*cat-Kan-cs-intros*]:
  assumes $\mathcal{Z}\ \beta$
    and $\alpha \in_\circ \beta$
    and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
    and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
    and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $A = \textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!)$
    and $B = \textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
    and $\mathfrak{C}' = \textit{cat-Set}\ \beta$
  shows $\textit{L-10-5-}\chi'\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a : A \mapsto_{iso_{\mathfrak{C}'}} B$
  ⟨*proof*⟩

lemma *L-10-5-*χ'*-arrow-is-arr*:
  assumes $\mathcal{Z}\ \beta$
    and $\alpha \in_\circ \beta$
    and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
    and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
    and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
  shows $\textit{L-10-5-}\chi'\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a :$
      $\textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\textit{cat-Set}\ \beta}$
      $\textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
    ⟨*proof*⟩

lemma *L-10-5-*χ'*-arrow-is-arr*$'$[*cat-Kan-cs-intros*]:
  assumes $\mathcal{Z}\ \beta$
    and $\alpha \in_\circ \beta$
    and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
    and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
    and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $A = \textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!)$
    and $B = \textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
    and $\mathfrak{C}' = \textit{cat-Set}\ \beta$
  shows $\textit{L-10-5-}\chi'\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a : A \mapsto_{\mathfrak{C}'} B$
  ⟨*proof*⟩

15.8 Lemma X.5: *L-10-5-*χ

15.8.1 Definition and elementary properties

definition *L-10-5-*χ :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where $\textit{L-10-5-}\chi\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c =$
    [
      $(\lambda a{\in_\circ}\mathfrak{T}(\!|HomCod|\!)(\!|Obj|\!).\ \textit{L-10-5-}\chi'\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a)$,
      $\textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K})$,
      $\textit{L-10-5-N}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c$,
      $\textit{op-cat}\ (\mathfrak{T}(\!|HomCod|\!))$,
      $\textit{cat-Set}\ \beta$
    $]_\circ$

Components.

lemma *L-10-5-*χ*-components*:
  shows $\textit{L-10-5-}\chi\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|NTMap|\!) =$
    $(\lambda a{\in_\circ}\mathfrak{T}(\!|HomCod|\!)(\!|Obj|\!).\ \textit{L-10-5-}\chi'\textit{-arrow}\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a)$
    and [*cat-Kan-cs-simps*]:
      $\textit{L-10-5-}\chi\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|NTDom|\!) = \textit{cf-Cone}\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K})$

and [*cat-Kan-cs-simps*]:
L-10-5-χ α β $\mathfrak{T}$ $\mathfrak{K}$ c(|*NTCod*|) = *L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ c*
and *L-10-5-χ α β $\mathfrak{T}$ $\mathfrak{K}$ c*(|*NTDGDom*|) = *op-cat* ($\mathfrak{T}$(|*HomCod*|))
and [*cat-Kan-cs-simps*]: *L-10-5-χ α β $\mathfrak{T}$ $\mathfrak{K}$ c*(|*NTDGCod*|) = *cat-Set β*
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{T}$
assumes $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *L-10-5-χ-components′* =
L-10-5-χ-components[**where** $\mathfrak{T}$=$\mathfrak{T}$, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-χ-components′*(*4*)

end

15.8.2 Natural transformation map

mk-VLambda *L-10-5-χ-components*(*1*)
|*vsv L-10-5-χ-NTMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{T}$
assumes $\mathfrak{T}$: $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

mk-VLambda *L-10-5-χ-components*(*1*)[**where** $\mathfrak{T}$=$\mathfrak{T}$, *unfolded cat-cs-simps*]
|*vdomain L-10-5-χ-NTMap-vdomain*[*cat-Kan-cs-simps*]|
|*app L-10-5-χ-NTMap-app*[*cat-Kan-cs-simps*]|

end

15.8.3 *L-10-5-χ* is a natural isomorphism

lemma *L-10-5-χ-is-iso-ntcf*:
— See lemma on page 245 in [9].
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}$(|*Obj*|)
shows *L-10-5-χ α β $\mathfrak{T}$ $\mathfrak{K}$ c* :
cf-Cone α β ($\mathfrak{T} \circ_{CF} c\ {}_{O}\sqcap_{CF} \mathfrak{K}$) $\mapsto_{CF.iso}$ *L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ c* :
op-cat $\mathfrak{A} \mapsto\mapsto_{C\beta}$ *cat-Set* β
(**is** ‹*?L-10-5-χ* : *?cf-Cone* $\mapsto_{CF.iso}$ *?L-10-5-N* : *op-cat* $\mathfrak{A} \mapsto\mapsto_{C\beta}$ *cat-Set* β›)
⟨*proof*⟩

15.9 The existence of a canonical limit or a canonical colimit for the pointwise Kan extensions

lemma (**in** *is-cat-pw-rKe*) *cat-pw-rKe-ex-cat-limit*:
— Based on the elements of Chapter X-5 in [9].

assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(|Obj|)$
obtains UA
where $UA : \mathfrak{G}(|ObjMap|)(|c|) <_{CF.lim} \mathfrak{T} \circ_{CF} c \ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (in *is-cat-pw-lKe*) *cat-pw-lKe-ex-cat-colimit*:
— Based on the elements of Chapter X-5 in [9].
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(|Obj|)$
obtains UA
where $UA : \mathfrak{T} \circ_{CF} \mathfrak{K} \ {}_{CF}\sqcap_O \ c >_{CF.colim} \mathfrak{F}(|ObjMap|)(|c|) : \mathfrak{K} \ {}_{CF}\downarrow \ c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

15.10 The limit and the colimit for the pointwise Kan extensions

15.10.1 Definition and elementary properties

See Theorem 3 in Chapter X-5 in [9].

definition *the-pw-cat-rKe-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ c$ =
[
$\mathfrak{G}(|ObjMap|)(|c|)$,
(
SOME UA.
$UA : \mathfrak{G}(|ObjMap|)(|c|) <_{CF.lim} \mathfrak{T} \circ_{CF} c \ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{T}(|HomCod|)$
)
$]_\circ$

definition *the-pw-cat-lKe-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ c$ =
[
$\mathfrak{F}(|ObjMap|)(|c|)$,
op-ntcf
(
the-pw-cat-rKe-limit α (*op-cf* $\mathfrak{K}$) (*op-cf* $\mathfrak{T}$) (*op-cf* $\mathfrak{F}$) $c(|UArr|) \circ_{NTCF-CF}$
op-cf-obj-comma $\mathfrak{K}$ c
)
$]_\circ$

Components.

lemma *the-pw-cat-rKe-limit-components*:
shows *the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ c(|UObj|) = \mathfrak{G}(|ObjMap|)(|c|)$
and *the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ c(|UArr|)$ =
(
SOME UA.
$UA : \mathfrak{G}(|ObjMap|)(|c|) <_{CF.lim} \mathfrak{T} \circ_{CF} c \ {}_O\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{T}(|HomCod|)$
)
⟨*proof*⟩

lemma *the-pw-cat-lKe-colimit-components*:
shows *the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ c(|UObj|) = \mathfrak{F}(|ObjMap|)(|c|)$
and *the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ c(|UArr|)$ = *op-ntcf*
(
the-pw-cat-rKe-limit α (*op-cf* $\mathfrak{K}$) (*op-cf* $\mathfrak{T}$) (*op-cf* $\mathfrak{F}$) $c(|UArr|) \circ_{NTCF-CF}$
op-cf-obj-comma $\mathfrak{K}$ c

)
⟨*proof*⟩

context *is-functor*
begin

lemmas *the-pw-cat-rKe-limit-components′* =
the-pw-cat-rKe-limit-components[**where** $\mathfrak{T}$=$\mathfrak{F}$, *unfolded cat-cs-simps*]

end

15.10.2 The limit for the pointwise right Kan extension is a limit, the colimit for the pointwise left Kan extension is a colimit

lemma (**in** *is-cat-pw-rKe*) *cat-pw-rKe-the-pw-cat-rKe-limit-is-cat-limit*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $c \in_\circ \mathfrak{C}(|Obj|)$
shows *the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ c(|UArr|)$:
the-pw-cat-rKe-limit $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}\ c(|UObj|) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\sqcap_{CF}\ \mathfrak{K}$:
$c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-pw-lKe*) *cat-pw-lKe-the-pw-cat-lKe-colimit-is-cat-colimit*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$ **and** $c \in_\circ \mathfrak{C}(|Obj|)$
shows *the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ c(|UArr|)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\sqcap_O\ c >_{CF.colim}$ *the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}\ c(|UObj|)$:
$\mathfrak{K}\ {}_{CF}\!\downarrow c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-pw-rKe*) *cat-pw-rKe-the-ntcf-rKe-is-cat-rKe*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ (*the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}$) :
the-cf-rKe $\alpha\ \mathfrak{T}\ \mathfrak{K}$ (*the-pw-cat-rKe-limit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{G}$) $\circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T}$:
$\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-cat-pw-lKe*) *cat-pw-lKe-the-ntcf-lKe-is-cat-lKe*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ (*the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}$) :
$\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *the-cf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ (*the-pw-cat-lKe-colimit* $\alpha\ \mathfrak{K}\ \mathfrak{T}\ \mathfrak{F}$) $\circ_{CF} \mathfrak{K}$:
$\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

16 Pointwise Kan extensions: application example

16.1 Background

The application example presented in this section is based on Exercise 6.1.ii in [14]. The primary purpose of this section is the instantiation of the locales associated with the pointwise Kan extensions.

lemma *cat-ordinal-2-is-arrE*:
assumes $f : a \mapsto_{cat\text{-}ordinal\ (2_{\mathbb{N}})} b$
obtains $f = [0,\ 0]_\circ$ **and** $a = 0$ **and** $b = 0$
$|\ f = [0,\ 1_{\mathbb{N}}]_\circ$ **and** $a = 0$ **and** $b = 1_{\mathbb{N}}$
$|\ f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ$ **and** $a = 1_{\mathbb{N}}$ **and** $b = 1_{\mathbb{N}}$
⟨*proof*⟩

lemma *cat-ordinal-3-is-arrE*:
assumes $f : a \mapsto_{cat\text{-}ordinal\ (3_{\mathbb{N}})} b$
obtains $f = [0,\ 0]_\circ$ **and** $a = 0$ **and** $b = 0$
$|\ f = [0,\ 1_{\mathbb{N}}]_\circ$ **and** $a = 0$ **and** $b = 1_{\mathbb{N}}$
$|\ f = [0,\ 2_{\mathbb{N}}]_\circ$ **and** $a = 0$ **and** $b = 2_{\mathbb{N}}$
$|\ f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ$ **and** $a = 1_{\mathbb{N}}$ **and** $b = 1_{\mathbb{N}}$
$|\ f = [1_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$ **and** $a = 1_{\mathbb{N}}$ **and** $b = 2_{\mathbb{N}}$
$|\ f = [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$ **and** $a = 2_{\mathbb{N}}$ **and** $b = 2_{\mathbb{N}}$
⟨*proof*⟩

lemma *0123*: $0 \in_\circ 2_{\mathbb{N}}\ 1_{\mathbb{N}} \in_\circ 2_{\mathbb{N}}\ 0 \in_\circ 3_{\mathbb{N}}\ 1_{\mathbb{N}} \in_\circ 3_{\mathbb{N}}\ 2_{\mathbb{N}} \in_\circ 3_{\mathbb{N}}$ ⟨*proof*⟩

16.2 $\mathfrak{K}23$

16.2.1 Definition and elementary properties

definition $\mathfrak{K}23$ :: V
where $\mathfrak{K}23$ =
[
$(\lambda a \in_\circ cat\text{-}ordinal\ (2_{\mathbb{N}})(|Obj|).\ if\ a = 0\ then\ 0\ else\ 2_{\mathbb{N}})$,
(
$\lambda f \in_\circ cat\text{-}ordinal\ (2_{\mathbb{N}})(|Arr|).$
$if\ f = [0,\ 0]_\circ \Rightarrow [0,\ 0]_\circ$
$|\ f = [0,\ 1_{\mathbb{N}}]_\circ \Rightarrow [0,\ 2_{\mathbb{N}}]_\circ$
$|\ f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ \Rightarrow [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$
$|\ otherwise \Rightarrow 0$
),
$cat\text{-}ordinal\ (2_{\mathbb{N}})$,
$cat\text{-}ordinal\ (3_{\mathbb{N}})$
$]_\circ$

Components.

lemma *$\mathfrak{K}23$-components*:
shows $\mathfrak{K}23(|ObjMap|) = (\lambda a \in_\circ cat\text{-}ordinal\ (2_{\mathbb{N}})(|Obj|).\ if\ a = 0\ then\ 0\ else\ 2_{\mathbb{N}})$
and $\mathfrak{K}23(|ArrMap|)$ =
(
$\lambda f \in_\circ cat\text{-}ordinal\ (2_{\mathbb{N}})(|Arr|).$
$if\ f = [0,\ 0]_\circ \Rightarrow [0,\ 0]_\circ$
$|\ f = [0,\ 1_{\mathbb{N}}]_\circ \Rightarrow [0,\ 2_{\mathbb{N}}]_\circ$
$|\ f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ \Rightarrow [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$
$|\ otherwise \Rightarrow 0$
)

and [*cat-Kan-cs-simps*]: $\mathfrak{K}23(|HomDom|)$ = *cat-ordinal* ($2_{\mathbb{N}}$)
and [*cat-Kan-cs-simps*]: $\mathfrak{K}23(|HomCod|)$ = *cat-ordinal* ($3_{\mathbb{N}}$)
⟨*proof*⟩

16.2.2 Object map

mk-VLambda $\mathfrak{K}$*23-components*(*1*)
|*vsv* $\mathfrak{K}$*23-ObjMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain* $\mathfrak{K}$*23-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app* $\mathfrak{K}$*23-ObjMap-app*|

lemma $\mathfrak{K}$*23-ObjMap-app-0*[*cat-Kan-cs-simps*]:
assumes $x = 0$
shows $\mathfrak{K}23(|ObjMap|)(|x|) = 0$
⟨*proof*⟩

lemma $\mathfrak{K}$*23-ObjMap-app-1*[*cat-Kan-cs-simps*]:
assumes $x = 1_{\mathbb{N}}$
shows $\mathfrak{K}23(|ObjMap|)(|x|) = 2_{\mathbb{N}}$
⟨*proof*⟩

16.2.3 Arrow map

mk-VLambda $\mathfrak{K}$*23-components*(*2*)
|*vsv* $\mathfrak{K}$*23-ArrMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain* $\mathfrak{K}$*23-ArrMap-vdomain*[*cat-Kan-cs-simps*]|
|*app* $\mathfrak{K}$*23-ArrMap-app*|

lemma $\mathfrak{K}$*23-ArrMap-app-00*[*cat-Kan-cs-simps*]:
assumes $f = [0, 0]_\circ$
shows $\mathfrak{K}23(|ArrMap|)(|f|) = [0, 0]_\circ$
⟨*proof*⟩

lemma $\mathfrak{K}$*23-ArrMap-app-01*[*cat-Kan-cs-simps*]:
assumes $f = [0, 1_{\mathbb{N}}]_\circ$
shows $\mathfrak{K}23(|ArrMap|)(|f|) = [0, 2_{\mathbb{N}}]_\circ$
⟨*proof*⟩

lemma $\mathfrak{K}$*23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_\circ$
shows $\mathfrak{K}23(|ArrMap|)(|f|) = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$
⟨*proof*⟩

16.2.4 $\mathfrak{K}23$ is a tiny functor

lemma (**in** $\mathcal{Z}$) $\mathfrak{K}$*23-is-functor*: $\mathfrak{K}23$: *cat-ordinal* ($2_{\mathbb{N}}$) $\mapsto\mapsto_{C\alpha}$ *cat-ordinal* ($3_{\mathbb{N}}$)
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) $\mathfrak{K}$*23-is-functor'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{A}'$ = *cat-ordinal* ($2_{\mathbb{N}}$)
and $\mathfrak{B}'$ = *cat-ordinal* ($3_{\mathbb{N}}$)
shows $\mathfrak{K}23 : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = $\mathcal{Z}$.$\mathfrak{K}$*23-is-functor'*

lemma (**in** $\mathcal{Z}$) $\mathfrak{K}$*23-is-tiny-functor*:
$\mathfrak{K}23$: *cat-ordinal* ($2_{\mathbb{N}}$) $\mapsto\mapsto_{C.tiny\alpha}$ *cat-ordinal* ($3_{\mathbb{N}}$)
⟨*proof*⟩

lemma (**in** *𝒵*) *𝔎23-is-tiny-functor'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{A}'$ = *cat-ordinal* ($2_{\mathbb{N}}$)
and $\mathfrak{B}'$ = *cat-ordinal* ($3_{\mathbb{N}}$)
shows *𝔎23* : $\mathfrak{A}' \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = *𝒵.𝔎23-is-tiny-functor'*

16.3 *LK23*: the functor associated with the left Kan extension along *𝔎23*

16.3.1 Definition and elementary properties

definition *LK23* :: *V* ⇒ *V*
where *LK23* 𝔉 =
[
(
$\lambda a \in_\circ$ *cat-ordinal* ($3_{\mathbb{N}}$)⦇*Obj*⦈.
if a = *0* ⇒ 𝔉⦇*ObjMap*⦈⦇*0*⦈
| *a* = $1_{\mathbb{N}}$ ⇒ 𝔉⦇*ObjMap*⦈⦇*0*⦈
| *a* = $2_{\mathbb{N}}$ ⇒ 𝔉⦇*ObjMap*⦈⦇$1_{\mathbb{N}}$⦈
| *otherwise* ⇒ 𝔉⦇*HomCod*⦈⦇*Obj*⦈
),
(
$\lambda f \in_\circ$ *cat-ordinal* ($3_{\mathbb{N}}$)⦇*Arr*⦈.
if f = $[0, 0]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[0, 1_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[0, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, $1_{\mathbb{N}}$⦈$_\bullet$
| *f* = $[1_{\mathbb{N}}, 1_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[1_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, $1_{\mathbb{N}}$⦈$_\bullet$
| *f* = $[2_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇$1_{\mathbb{N}}$, $1_{\mathbb{N}}$⦈$_\bullet$
| *otherwise* ⇒ 𝔉⦇*HomCod*⦈⦇*Arr*⦈
),
cat-ordinal ($3_{\mathbb{N}}$),
𝔉⦇*HomCod*⦈
]$_\circ$

Components.

lemma *LK23-components*:
shows *LK23* 𝔉⦇*ObjMap*⦈ =
(
$\lambda a \in_\circ$ *cat-ordinal* ($3_{\mathbb{N}}$)⦇*Obj*⦈.
if a = *0* ⇒ 𝔉⦇*ObjMap*⦈⦇*0*⦈
| *a* = $1_{\mathbb{N}}$ ⇒ 𝔉⦇*ObjMap*⦈⦇*0*⦈
| *a* = $2_{\mathbb{N}}$ ⇒ 𝔉⦇*ObjMap*⦈⦇$1_{\mathbb{N}}$⦈
| *otherwise* ⇒ 𝔉⦇*HomCod*⦈⦇*Obj*⦈
)
and *LK23* 𝔉⦇*ArrMap*⦈ =
(
$\lambda f \in_\circ$ *cat-ordinal* ($3_{\mathbb{N}}$)⦇*Arr*⦈.
if f = $[0, 0]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[0, 1_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[0, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, $1_{\mathbb{N}}$⦈$_\bullet$
| *f* = $[1_{\mathbb{N}}, 1_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_\bullet$
| *f* = $[1_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇*0*, $1_{\mathbb{N}}$⦈$_\bullet$
| *f* = $[2_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$ ⇒ 𝔉⦇*ArrMap*⦈⦇$1_{\mathbb{N}}$, $1_{\mathbb{N}}$⦈$_\bullet$
| *otherwise* ⇒ 𝔉⦇*HomCod*⦈⦇*Arr*⦈
)

and *LK23* 𝔉⦇*HomDom*⦈ = *cat-ordinal* (*3*$_{\mathbb{N}}$)
and *LK23* 𝔉⦇*HomCod*⦈ = 𝔉⦇*HomCod*⦈
⟨*proof*⟩

context *is-functor*
begin

lemmas *LK23-components′* = *LK23-components*[**where** 𝔉=𝔉, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *LK23-components′*(*3*,*4*)

end

lemmas [*cat-Kan-cs-simps*] = *is-functor*.*LK23-components′*(*3*,*4*)

16.3.2 Object map

mk-VLambda *LK23-components*(*1*)
|*vsv LK23-ObjMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain LK23-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app LK23-ObjMap-app*|

lemma *LK23-ObjMap-app-0*[*cat-Kan-cs-simps*]:
assumes *a* = *0*
shows *LK23* 𝔉⦇*ObjMap*⦈⦇*a*⦈ = 𝔉⦇*ObjMap*⦈⦇*0*⦈
⟨*proof*⟩

lemma *LK23-ObjMap-app-1*[*cat-Kan-cs-simps*]:
assumes *a* = *1*$_{\mathbb{N}}$
shows *LK23* 𝔉⦇*ObjMap*⦈⦇*a*⦈ = 𝔉⦇*ObjMap*⦈⦇*0*⦈
⟨*proof*⟩

lemma *LK23-ObjMap-app-2*[*cat-Kan-cs-simps*]:
assumes *a* = *2*$_{\mathbb{N}}$
shows *LK23* 𝔉⦇*ObjMap*⦈⦇*a*⦈ = 𝔉⦇*ObjMap*⦈⦇*1*$_{\mathbb{N}}$⦈
⟨*proof*⟩

16.3.3 Arrow map

mk-VLambda *LK23-components*(*2*)
|*vsv LK23-ArrMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain LK23-ArrMap-vdomain*[*cat-Kan-cs-simps*]|
|*app LK23-ArrMap-app*|

lemma *LK23-ArrMap-app-00*[*cat-Kan-cs-simps*]:
assumes *f* = [*0*, *0*]$_{\circ}$
shows *LK23* 𝔉⦇*ArrMap*⦈⦇*f*⦈ = 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-01*[*cat-Kan-cs-simps*]:
assumes *f* = [*0*, *1*$_{\mathbb{N}}$]$_{\circ}$
shows *LK23* 𝔉⦇*ArrMap*⦈⦇*f*⦈ = 𝔉⦇*ArrMap*⦈⦇*0*, *0*⦈$_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-02*[*cat-Kan-cs-simps*]:
assumes *f* = [*0*, *2*$_{\mathbb{N}}$]$_{\circ}$
shows *LK23* 𝔉⦇*ArrMap*⦈⦇*f*⦈ = 𝔉⦇*ArrMap*⦈⦇*0*, *1*$_{\mathbb{N}}$⦈$_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
  assumes $f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ$
  shows $LK23\ \mathfrak{F}(|ArrMap|)(|f|) = \mathfrak{F}(|ArrMap|)(|0,\ 0|)_\bullet$
⟨*proof*⟩

lemma *LK23-ArrMap-app-12*[*cat-Kan-cs-simps*]:
  assumes $f = [1_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$
  shows $LK23\ \mathfrak{F}(|ArrMap|)(|f|) = \mathfrak{F}(|ArrMap|)(|0,\ 1_{\mathbb{N}}|)_\bullet$
⟨*proof*⟩

lemma *LK23-ArrMap-app-22*[*cat-Kan-cs-simps*]:
  assumes $f = [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ$
  shows $LK23\ \mathfrak{F}(|ArrMap|)(|f|) = \mathfrak{F}(|ArrMap|)(|1_{\mathbb{N}},\ 1_{\mathbb{N}}|)_\bullet$
⟨*proof*⟩

16.3.4 *LK23* is a functor

lemma *cat-LK23-is-functor*:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $LK23\ \mathfrak{F} : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-LK23-is-functor'*[*cat-Kan-cs-intros*]:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{A}' = cat\text{-}ordinal\ (3_{\mathbb{N}})$
  shows $LK23\ \mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  ⟨*proof*⟩

16.3.5 The fundamental property of *LK23*

lemma *cf-comp-LK23-𝔎23*[*cat-Kan-cs-simps*]:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $LK23\ \mathfrak{F} \circ_{CF} \mathfrak{K}23 = \mathfrak{F}$
⟨*proof*⟩

16.4 *RK23*: the functor associated with the right Kan extension along 𝔎*23*

16.4.1 Definition and elementary properties

definition $RK23 :: V \Rightarrow V$
  where $RK23\ \mathfrak{F} =$
    [
      (
        $\lambda a \in_\circ cat\text{-}ordinal\ (3_{\mathbb{N}})(|Obj|).$
          $if\ a = 0 \Rightarrow \mathfrak{F}(|ObjMap|)(|0|)$
          $|\ a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(|ObjMap|)(|1_{\mathbb{N}}|)$
          $|\ a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(|ObjMap|)(|1_{\mathbb{N}}|)$
          $|\ otherwise \Rightarrow \mathfrak{F}(|HomCod|)(|Obj|)$
      ),
      (
        $\lambda f \in_\circ cat\text{-}ordinal\ (3_{\mathbb{N}})(|Arr|).$
          $if\ f = [0,\ 0]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|0,\ 0|)_\bullet$
          $|\ f = [0,\ 1_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|0,\ 1_{\mathbb{N}}|)_\bullet$
          $|\ f = [0,\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|0,\ 1_{\mathbb{N}}|)_\bullet$
          $|\ f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|1_{\mathbb{N}},\ 1_{\mathbb{N}}|)_\bullet$
          $|\ f = [1_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|1_{\mathbb{N}},\ 1_{\mathbb{N}}|)_\bullet$
          $|\ f = [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}(|ArrMap|)(|1_{\mathbb{N}},\ 1_{\mathbb{N}}|)_\bullet$
          $|\ otherwise \Rightarrow \mathfrak{F}(|HomCod|)(|Arr|)$

),
cat-ordinal ($3_{\mathbb{N}}$),
$\mathfrak{F}$(|*HomCod*|)
]$_\circ$

Components.

lemma *RK23-components*:
shows *RK23* $\mathfrak{F}$(|*ObjMap*|) =
(
$\lambda a\in_\circ$*cat-ordinal* ($3_{\mathbb{N}}$)(|*Obj*|).
if $a = 0 \Rightarrow \mathfrak{F}$(|*ObjMap*|)(|*0*|)
| $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}$(|*ObjMap*|)(|$1_{\mathbb{N}}$|)
| $a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}$(|*ObjMap*|)(|$1_{\mathbb{N}}$|)
| *otherwise* $\Rightarrow \mathfrak{F}$(|*HomCod*|)(|*Obj*|)
)
and *RK23* $\mathfrak{F}$(|*ArrMap*|) =
(
$\lambda f\in_\circ$*cat-ordinal* ($3_{\mathbb{N}}$)(|*Arr*|).
if $f = [0,\ 0]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|*0*, *0*|)$_\bullet$
| $f = [0,\ 1_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|*0*, $1_{\mathbb{N}}$|)$_\bullet$
| $f = [0,\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|*0*, $1_{\mathbb{N}}$|)$_\bullet$
| $f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|$1_{\mathbb{N}}$, $1_{\mathbb{N}}$|)$_\bullet$
| $f = [1_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|$1_{\mathbb{N}}$, $1_{\mathbb{N}}$|)$_\bullet$
| $f = [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_\circ \Rightarrow \mathfrak{F}$(|*ArrMap*|)(|$1_{\mathbb{N}}$, $1_{\mathbb{N}}$|)$_\bullet$
| *otherwise* $\Rightarrow \mathfrak{F}$(|*HomCod*|)(|*Arr*|)
)
and *RK23* $\mathfrak{F}$(|*HomDom*|) = *cat-ordinal* ($3_{\mathbb{N}}$)
and *RK23* $\mathfrak{F}$(|*HomCod*|) = $\mathfrak{F}$(|*HomCod*|)
⟨*proof*⟩

context *is-functor*
begin

lemmas *RK23-components'* = *RK23-components*[**where** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *RK23-components'*(*3*,*4*)

end

lemmas [*cat-Kan-cs-simps*] = *is-functor*.*RK23-components'*(*3*,*4*)

16.4.2 Object map

mk-VLambda *RK23-components*(*1*)
|*vsv RK23-ObjMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain RK23-ObjMap-vdomain*[*cat-Kan-cs-simps*]|
|*app RK23-ObjMap-app*|

lemma *RK23-ObjMap-app-0*[*cat-Kan-cs-simps*]:
assumes $a = 0$
shows *RK23* $\mathfrak{F}$(|*ObjMap*|)(|*a*|) = $\mathfrak{F}$(|*ObjMap*|)(|*0*|)
⟨*proof*⟩

lemma *RK23-ObjMap-app-1*[*cat-Kan-cs-simps*]:
assumes $a = 1_{\mathbb{N}}$
shows *RK23* $\mathfrak{F}$(|*ObjMap*|)(|*a*|) = $\mathfrak{F}$(|*ObjMap*|)(|$1_{\mathbb{N}}$|)
⟨*proof*⟩

lemma *RK23-ObjMap-app-2*[*cat-Kan-cs-simps*]:
  assumes $a = 2_{\mathbb{N}}$
  shows $RK23\ \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|1_{\mathbb{N}}|\!)$
  ⟨*proof*⟩

16.4.3 Arrow map

mk-VLambda *RK23-components*(*2*)
  |*vsv RK23-ArrMap-vsv*[*cat-Kan-cs-intros*]|
  |*vdomain RK23-ArrMap-vdomain*[*cat-Kan-cs-simps*]|
  |*app RK23-ArrMap-app*|

lemma *RK23-ArrMap-app-00*[*cat-Kan-cs-simps*]:
  assumes $f = [0, 0]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 0|\!)_\bullet$
⟨*proof*⟩

lemma *RK23-ArrMap-app-01*[*cat-Kan-cs-simps*]:
  assumes $f = [0, 1_{\mathbb{N}}]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 1_{\mathbb{N}}|\!)_\bullet$
⟨*proof*⟩

lemma *RK23-ArrMap-app-02*[*cat-Kan-cs-simps*]:
  assumes $f = [0, 2_{\mathbb{N}}]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 1_{\mathbb{N}}|\!)_\bullet$
⟨*proof*⟩

lemma *RK23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
  assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|1_{\mathbb{N}}, 1_{\mathbb{N}}|\!)_\bullet$
⟨*proof*⟩

lemma *RK23-ArrMap-app-12*[*cat-Kan-cs-simps*]:
  assumes $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|1_{\mathbb{N}}, 1_{\mathbb{N}}|\!)_\bullet$
⟨*proof*⟩

lemma *RK23-ArrMap-app-22*[*cat-Kan-cs-simps*]:
  assumes $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_\circ$
  shows $RK23\ \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|1_{\mathbb{N}}, 1_{\mathbb{N}}|\!)_\bullet$
⟨*proof*⟩

16.4.4 *RK23* is a functor

lemma *cat-RK23-is-functor*:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
  shows $RK23\ \mathfrak{F} : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-RK23-is-functor'*[*cat-Kan-cs-intros*]:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{A}' = cat\text{-}ordinal\ (3_{\mathbb{N}})$
  shows $RK23\ \mathfrak{F} : \mathfrak{A}' \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
  ⟨*proof*⟩

16.4.5 The fundamental property of *RK23*

lemma *cf-comp-RK23-*$\mathfrak{K}$*23*[*cat-Kan-cs-simps*]:
  assumes $\mathfrak{F} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$

shows $RK23\ \mathfrak{F} \circ_{CF} \mathfrak{K}23 = \mathfrak{F}$
⟨*proof*⟩

16.5 *RK-σ23*: towards the universal property of the right Kan extension along $\mathfrak{K}23$

16.5.1 Definition and elementary properties

definition *RK-σ23* :: *V* ⇒ *V* ⇒ *V* ⇒ *V*
where $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}' =$
[
(
$\lambda a \in_\circ cat\text{-}ordinal\ (3_{\mathbb{N}})(\!|Obj|\!).$
if $a = 0 \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|0|\!)$
$|\ a = 1_{\mathbb{N}} \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!) \circ_{A\mathfrak{T}(\!|HomCod|\!)} \mathfrak{F}'(\!|ArrMap|\!)(\!|1_{\mathbb{N}}, 2_{\mathbb{N}}|\!)_\bullet$
$|\ a = 2_{\mathbb{N}} \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
$|\ otherwise \Rightarrow \mathfrak{T}(\!|HomCod|\!)(\!|Arr|\!)$
),
$\mathfrak{F}'$,
$RK23\ \mathfrak{T}$,
$cat\text{-}ordinal\ (3_{\mathbb{N}})$,
$\mathfrak{F}'(\!|HomCod|\!)$
$]_\circ$

Components.

lemma *RK-σ23-components*:
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(\!|NTMap|\!) =$
(
$\lambda a \in_\circ cat\text{-}ordinal\ (3_{\mathbb{N}})(\!|Obj|\!).$
if $a = 0 \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|0|\!)$
$|\ a = 1_{\mathbb{N}} \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!) \circ_{A\mathfrak{T}(\!|HomCod|\!)} \mathfrak{F}'(\!|ArrMap|\!)(\!|1_{\mathbb{N}}, 2_{\mathbb{N}}|\!)_\bullet$
$|\ a = 2_{\mathbb{N}} \Rightarrow \varepsilon'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
$|\ otherwise \Rightarrow \mathfrak{T}(\!|HomCod|\!)(\!|Arr|\!)$
)
and $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(\!|NTDom|\!) = \mathfrak{F}'$
and $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(\!|NTCod|\!) = RK23\ \mathfrak{T}$
and $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(\!|NTDGDom|\!) = cat\text{-}ordinal\ (3_{\mathbb{N}})$
and $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(\!|NTDGCod|\!) = \mathfrak{F}'(\!|HomCod|\!)$
⟨*proof*⟩

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{F}'\ \mathfrak{T}$
assumes $\mathfrak{F}'$: $\mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T}$: $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{F}'$: *is-functor* α ‹$cat\text{-}ordinal\ (3_{\mathbb{N}})$› $\mathfrak{A}\ \mathfrak{F}'$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α ‹$cat\text{-}ordinal\ (2_{\mathbb{N}})$› $\mathfrak{A}\ \mathfrak{T}$ ⟨*proof*⟩

lemmas *RK-σ23-components'* =
RK-σ23-components[**where** $\mathfrak{F}'$=$\mathfrak{F}'$ **and** $\mathfrak{T}$=$\mathfrak{T}$, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *RK-σ23-components'*(*2–5*)

end

16.5.2 Natural transformation map

mk-VLambda *RK-σ23-components*(*1*)
|*vsv RK-σ23-NTMap-vsv*[*cat-Kan-cs-intros*]|
|*vdomain RK-σ23-NTMap-vdomain*[*cat-Kan-cs-simps*]|
|*app RK-σ23-NTMap-app*|

lemma *RK-σ23-NTMap-app-0*[*cat-Kan-cs-simps*]:
assumes $a = 0$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(|NTMap|)(|a|) = \varepsilon'(|NTMap|)(|0|)$
⟨*proof*⟩

lemma (**in** *is-functor*) *RK-σ23-NTMap-app-1*[*cat-Kan-cs-simps*]:
assumes $a = 1_{\mathbb{N}}$
shows $RK\text{-}\sigma 23\ \mathfrak{F}\ \varepsilon'\ \mathfrak{F}'(|NTMap|)(|a|) = \varepsilon'(|NTMap|)(|1_{\mathbb{N}}|) \circ_{A\mathfrak{B}} \mathfrak{F}'(|ArrMap|)(|1_{\mathbb{N}}, 2_{\mathbb{N}}|)$•
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.RK-σ23-NTMap-app-1*

lemma *RK-σ23-NTMap-app-2*[*cat-Kan-cs-simps*]:
assumes $a = 2_{\mathbb{N}}$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}'(|NTMap|)(|a|) = \varepsilon'(|NTMap|)(|1_{\mathbb{N}}|)$
⟨*proof*⟩

16.5.3 *RK-σ23* is a natural transformation

lemma *RK-σ23-is-ntcf*:
assumes $\mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\varepsilon' : \mathfrak{F}' \circ_{CF} \mathfrak{K}23 \mapsto_{CF} \mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}' : \mathfrak{F}' \mapsto_{CF} RK23\ \mathfrak{T} : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *RK-σ23-is-ntcf'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\varepsilon' : \mathfrak{F}' \circ_{CF} \mathfrak{K}23 \mapsto_{CF} \mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G}' = \mathfrak{F}'$
and $\mathfrak{H}' = RK23\ \mathfrak{T}$
and $\mathfrak{C}' = cat\text{-}ordinal\ (3_{\mathbb{N}})$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon'\ \mathfrak{F}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

16.6 The right Kan extension along $\mathfrak{K}23$

lemma *ε23-is-cat-rKe*:
assumes $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $ntcf\text{-}id\ \mathfrak{T}$:
$RK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23 \mapsto_{CF.rKe\alpha} \mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto_{C} cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

16.7 *LK-σ23*: towards the universal property of the left Kan extension along $\mathfrak{K}23$

16.7.1 Definition and elementary properties

definition *LK-σ23* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where $LK\text{-}\sigma 23\ \mathfrak{T}\ \eta'\ \mathfrak{F}' =$
[

(
  $\lambda a \in_\circ$ *cat-ordinal* $(3_{\mathbb{N}})(\!|Obj|\!)$.
   if $a = 0 \Rightarrow \eta'(\!|NTMap|\!)(\!|0|\!)$
   $|$ $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}'(\!|ArrMap|\!)(\!|0,\ 1_{\mathbb{N}}|\!)_\bullet \circ_{A\mathfrak{T}(\!|HomCod|\!)} \eta'(\!|NTMap|\!)(\!|0|\!)$
   $|$ $a = 2_{\mathbb{N}} \Rightarrow \eta'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
   $|$ *otherwise* $\Rightarrow \mathfrak{T}(\!|HomCod|\!)(\!|Arr|\!)$
),
LK23 $\mathfrak{T}$,
$\mathfrak{F}'$,
cat-ordinal $(3_{\mathbb{N}})$,
$\mathfrak{F}'(\!|HomCod|\!)$
$]_\circ$

Components.

lemma *LK-σ23-components*:
  shows *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTMap|\!)$ =
    (
      $\lambda a \in_\circ$ *cat-ordinal* $(3_{\mathbb{N}})(\!|Obj|\!)$.
        if $a = 0 \Rightarrow \eta'(\!|NTMap|\!)(\!|0|\!)$
        $|$ $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}'(\!|ArrMap|\!)(\!|0,\ 1_{\mathbb{N}}|\!)_\bullet \circ_{A\mathfrak{T}(\!|HomCod|\!)} \eta'(\!|NTMap|\!)(\!|0|\!)$
        $|$ $a = 2_{\mathbb{N}} \Rightarrow \eta'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
        $|$ *otherwise* $\Rightarrow \mathfrak{T}(\!|HomCod|\!)(\!|Arr|\!)$
)
    and *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTDom|\!)$ = *LK23* $\mathfrak{T}$
    and *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTCod|\!)$ = $\mathfrak{F}'$
    and *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTDGDom|\!)$ = *cat-ordinal* $(3_{\mathbb{N}})$
    and *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTDGCod|\!)$ = $\mathfrak{F}'(\!|HomCod|\!)$
  ⟨*proof*⟩

context
  fixes α $\mathfrak{A}$ $\mathfrak{F}'$ $\mathfrak{T}$
  assumes $\mathfrak{F}'$: $\mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
    and $\mathfrak{T}$: $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{F}'$: *is-functor* α ‹*cat-ordinal* $(3_{\mathbb{N}})$› $\mathfrak{A}$ $\mathfrak{F}'$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α ‹*cat-ordinal* $(2_{\mathbb{N}})$› $\mathfrak{A}$ $\mathfrak{T}$ ⟨*proof*⟩

lemmas *LK-σ23-components'* =
  LK-σ23-components[**where** $\mathfrak{F}'$=$\mathfrak{F}'$ **and** $\mathfrak{T}$=$\mathfrak{T}$, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *LK-σ23-components'*(*2–5*)

end

16.7.2 Natural transformation map

mk-VLambda *LK-σ23-components*(*1*)
  |*vsv LK-σ23-NTMap-vsv*[*cat-Kan-cs-intros*]|
  |*vdomain LK-σ23-NTMap-vdomain*[*cat-Kan-cs-simps*]|
  |*app LK-σ23-NTMap-app*|

lemma *LK-σ23-NTMap-app-0*[*cat-Kan-cs-simps*]:
  assumes $a = 0$
  shows *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'(\!|NTMap|\!)(\!|a|\!) = \eta'(\!|NTMap|\!)(\!|0|\!)$
  ⟨*proof*⟩

lemma (**in** *is-functor*) *LK-σ23-NTMap-app-1*[*cat-Kan-cs-simps*]:

assumes $a = 1_{\mathbb{N}}$
shows $LK\text{-}\sigma 23\ \mathfrak{F}\ \eta'\ \mathfrak{F}'(\!|NTMap|\!)(\!|a|\!) = \mathfrak{F}'(\!|ArrMap|\!)(\!|0,\ 1_{\mathbb{N}}|\!)_{\bullet} \circ_{A\mathfrak{B}} \eta'(\!|NTMap|\!)(\!|0|\!)$
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.LK-σ23-NTMap-app-1*

lemma *LK-σ23-NTMap-app-2*[*cat-Kan-cs-simps*]:
assumes $a = 2_{\mathbb{N}}$
shows $LK\text{-}\sigma 23\ \mathfrak{T}\ \eta'\ \mathfrak{F}'(\!|NTMap|\!)(\!|a|\!) = \eta'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
⟨*proof*⟩

16.7.3 *LK-σ23* is a natural transformation

lemma *LK-σ23-is-ntcf*:
assumes $\mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K}23 : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $LK\text{-}\sigma 23\ \mathfrak{T}\ \eta'\ \mathfrak{F}' : LK23\ \mathfrak{T} \mapsto_{CF} \mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *LK-σ23-is-ntcf'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F}' : cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K}23 : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G}' = LK23\ \mathfrak{T}$
and $\mathfrak{H}' = \mathfrak{F}'$
and $\mathfrak{C}' = cat\text{-}ordinal\ (3_{\mathbb{N}})$
shows $LK\text{-}\sigma 23\ \mathfrak{T}\ \eta'\ \mathfrak{F}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

16.8 The left Kan extension along *𝔎23*

lemma *η23-is-cat-rKe*:
assumes $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* 𝔗 :
$\mathfrak{T} \mapsto_{CF.lKe\alpha} LK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23 : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto_{C} cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

16.9 Pointwise Kan extensions along *𝔎23*

lemma *ε23-is-cat-pw-rKe*:
assumes $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* 𝔗 :
$RK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23 \mapsto_{CF.rKe.pw\alpha} \mathfrak{T}$:
$cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto_{C} cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

lemma *η23-is-cat-pw-lKe*:
assumes $\mathfrak{T} : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* 𝔗 :
$\mathfrak{T} \mapsto_{CF.lKe.pw\alpha} LK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23$:
$cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto_{C} cat\text{-}ordinal\ (3_{\mathbb{N}}) \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

References

[1] nLab. URL https://ncatlab.org/nlab/show/HomePage.

[2] Wikipedia, 2001. URL https://www.wikipedia.org/.

[3] S. Awodey. *Category Theory*. Oxford University Press, Oxford, UK, 2010. ISBN 978-0-19-958736-0.

[4] P. Bodo. *Categories and Functors*. Academic Press, New York, 1970.

[5] F. Haftmann. Sketch-and-Explore, 2021. URL https://isabelle.in.tum.de/library/HOL/HOL-ex/Sketch_and_Explore.html.

[6] T. W. Hungerford. *Algebra*. Springer, New York, 2003. ISBN 978-0-387-90518-1.

[7] D. M. Kan. Adjoint Functors. *Transactions of the American Mathematical Society*, 87(2): 294–329, 1958.

[8] M. C. Lehner. "All Concepts are Kan Extensions": Kan Extensions as the Most Universal of the Universal Constructions. Bachelor of Arts Thesis, Harvard College, Cambridge, MA, 2014.

[9] S. Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer, New York, 2 edition, 2010. ISBN 978-1-4419-3123-8.

[10] M. Milehins. Category Theory for ZFC in HOL II: Elementary Theory of 1-Categories. *Archive of Formal Proofs*, 2021.

[11] S. Obua. Partizan Games in Isabelle/HOLZF. In K. Barkaoui, A. Cavalcanti, and A. Cerone, editors, *ICTAC 2006*, volume 4281, pages 272–286. Springer, Berlin, 2006. ISBN 978-3-540-48815-6.

[12] L. C. Paulson. Natural Deduction as Higher-Order Resolution. *The Journal of Logic Programming*, 3(3):237–258, 1986. doi: 10.1016/0743-1066(86)90015-4.

[13] L. C. Paulson. Zermelo Fraenkel Set Theory in Higher-Order Logic. *Archive of Formal Proofs*, 2019.

[14] E. Riehl. *Category Theory in Context*. Emily Riehl, 2016.