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

Mihails Milehins

December 14, 2021

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* ([12], also see [10]) of the formal proof assistant *Isabelle* [11].

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" [4] 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 [9].

[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 [8].

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

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

Components.

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

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

Universal maps for the opposite functor.

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

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

lemmas [*cat-op-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 $d(\lfloor ArrVal \rfloor)$)
  ⟨*proof*⟩

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

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

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

lemma (**in** *is-functor*) *umap-fo-ArrVal-vdomain*:
assumes $u : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor) \mapsto_{\mathfrak{B}} c$
shows $\mathcal{D}_{\circ}\ (umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d(\lfloor ArrVal \rfloor)) = Hom\ \mathfrak{A}\ d\ r$
⟨*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** $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor)$
shows $umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ d(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f' \rfloor) \circ_{A\mathfrak{B}} u$
⟨*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** $u : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor) \mapsto_{\mathfrak{B}} c$
shows $umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor) = u \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f' \rfloor)$
⟨*proof*⟩

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

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

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

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

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

lemma (**in** *is-functor*) *cf-arr-Set-umap-fo*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and *r*: $r \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
and *d*: $d \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
and *u*: $u : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor) \mapsto_{\mathfrak{B}} c$
shows *arr-Set* $\alpha\ (umap\text{-}fo\ \mathfrak{F}\ c\ r\ u\ d)$
⟨*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 $\mathfrak{C} = cat\text{-}Set\ \alpha$
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 $\mathfrak{C} = cat\text{-}Set\ \alpha$
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 [8].

definition *universal-arrow-of* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *universal-arrow-of* $\mathfrak{F}$ c r u ⟷
(
$r \in_\circ \mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!) \wedge$
$u : c \mapsto_{\mathfrak{F}(\!|HomCod|\!)} \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!) \wedge$
(
$\forall r'\ u'.$

$r' \in_\circ \mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!) \longrightarrow$
$u' : c \mapsto_{\mathfrak{F}(\!|HomCod|\!)} \mathfrak{F}(\!|ObjMap|\!)(\!|r'|\!) \longrightarrow$
$(\exists! f'.\ f' : r \mapsto_{\mathfrak{F}(\!|HomDom|\!)} r' \wedge u' = umap\text{-}of\ \mathfrak{F}\ c\ r\ u\ r'(\!|ArrVal|\!)(\!|f'|\!))$
)
)

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* $(op\text{-}cf\ \mathfrak{F})\ c\ r\ u$

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 $(op\text{-}cf\ \mathfrak{F})\ c\ r\ u \longleftrightarrow$ *universal-arrow-fo* $\mathfrak{F}\ c\ r\ u$
⟨*proof*⟩

lemma (**in** *is-functor*) *op-cf-universal-arrow-fo*[*cat-op-simps*]:
universal-arrow-fo $(op\text{-}cf\ \mathfrak{F})\ c\ r\ u \longleftrightarrow$ *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 [8].

lemma (in *is-functor*) *cf-universal-arrow-of-ex-is-arr-isomorphism*:
— The proof is based on the ideas expressed in the proof of Theorem 5.2 in Chapter Introduction in [5].
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-arr-isomorphism*:
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-arr-isomorphism*:
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-arr-isomorphism*:
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*⟩

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 [8].

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

definition *ntcf-ua-fo* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

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

Components.

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

lemma *ntcf-ua-fo-components*:
shows *ntcf-ua-fo* α $\mathfrak{F}$ c r $u(\!|NTMap|\!)$ = $(\lambda d \in_\circ \mathfrak{F}(\!|HomDom|\!)(\!|Obj|\!).$ *umap-fo* $\mathfrak{F}$ c r u $d)$
and *ntcf-ua-fo* α $\mathfrak{F}$ c r $u(\!|NTDom|\!)$ = $Hom_{O.C\alpha}$*op-cat* $(\mathfrak{F}(\!|HomDom|\!))(r,-)$
and *ntcf-ua-fo* α $\mathfrak{F}$ c r $u(\!|NTCod|\!)$ =
$Hom_{O.C\alpha}$*op-cat* $(\mathfrak{F}(\!|HomCod|\!))(c,-) \circ_{CF}$ *op-cf* $\mathfrak{F}$
and *ntcf-ua-fo* α $\mathfrak{F}$ c r $u(\!|NTDGDom|\!)$ = *op-cat* $(\mathfrak{F}(\!|HomDom|\!))$
and *ntcf-ua-fo* α $\mathfrak{F}$ 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* α $\mathfrak{F}$ c r $u(\!|NTMap|\!)$ = $(\lambda d \in_\circ \mathfrak{A}(\!|Obj|\!).$ *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* $\alpha(\!|Arr|\!)$
⟨*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-of $\mathfrak{F}$ c r u b $\circ_{A\,cat\text{-}Set\ \alpha}$ *cf-hom* $\mathfrak{A}$ $[\mathfrak{A}(\!|CId|\!)(\!|r|\!), f]_{\circ}$ =
cf-hom $\mathfrak{B}$ $[\mathfrak{B}(\!|CId|\!)(\!|c|\!), \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)]_{\circ}$ $\circ_{A\,cat\text{-}Set\ \alpha}$ *umap-of* $\mathfrak{F}$ c r u a
(**is** ‹*?uof-b* $\circ_{A\,cat\text{-}Set\ \alpha}$ *?rf* = *?cf* $\circ_{A\,cat\text{-}Set\ \alpha}$ *?uof-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 η : *cf-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-of $\mathfrak{G}$ c' $(\mathfrak{F}(\!|ObjMap|\!)(\!|c'|\!))$ $(\eta(\!|NTMap|\!)(\!|c'|\!))$ d' $\circ_{A\,cat\text{-}Set\ \alpha}$
cf-hom $\mathfrak{D}$ $[\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!), f]_{\circ}$ =
cf-hom $\mathfrak{C}$ $[g, \mathfrak{G}(\!|ArrMap|\!)(\!|f|\!)]_{\circ}$ $\circ_{A\,cat\text{-}Set\ \alpha}$
umap-of $\mathfrak{G}$ c $(\mathfrak{F}(\!|ObjMap|\!)(\!|c|\!))$ $(\eta(\!|NTMap|\!)(\!|c|\!))$ d

(**is** ‹*?uof-c′d′* ∘$_{A\,cat\text{-}Set\ \alpha}$ *?𝔉gf* = *?g𝔊f* ∘$_{A\,cat\text{-}Set\ \alpha}$ *?uof-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-ua-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-ua-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 [8].

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

3 Limits

3.1 Background

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

3.2 Cone and cocone

In the context of this work, the concept of a cone corresponds to that of a cone to the base of a functor from a vertex, as defined in Chapter III-4 in [8]; the concept of a cocone corresponds to that of a cone from the base of a functor to a vertex, as defined in Chapter III-3 in [8].

locale *is-cat-cone* = *is-ntcf* α $\mathfrak{J}$ $\mathfrak{C}$ ‹*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ c› $\mathfrak{F}$ $\mathfrak{N}$ **for** α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$ +
  assumes *cat-cone-obj*[*cat-lim-cs-intros*]: $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$

syntax *-is-cat-cone* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (‹(- :/ - <$_{CF.cone}$ - :/ - ↦↦$_{C^1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations $\mathfrak{N}$: c <$_{CF.cone}$ $\mathfrak{F}$: $\mathfrak{J}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-cone α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

locale *is-cat-cocone* = *is-ntcf* α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ ‹*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ c› $\mathfrak{N}$ **for** α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$ +
  assumes *cat-cocone-obj*[*cat-lim-cs-intros*]: $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$

syntax *-is-cat-cocone* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (‹(- :/ - >$_{CF.cocone}$ - :/ - ↦↦$_{C^1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations $\mathfrak{N}$: $\mathfrak{F}$ >$_{CF.cocone}$ c : $\mathfrak{J}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-cocone α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

Rules.

lemma (**in** *is-cat-cone*) *is-cat-cone-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $c' = c$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
  shows $\mathfrak{N}$: c' <$_{CF.cone}$ $\mathfrak{F}'$: $\mathfrak{J}'$ ↦↦$_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-cone-def*[*unfolded is-cat-cone-axioms-def*]
  |*intro is-cat-coneI*|
  |*dest is-cat-coneD*[*dest*!]|
  |*elim is-cat-coneE*[*elim*!]|

lemma (**in** *is-cat-cone*) *is-cat-coneD'*[*cat-lim-cs-intros*]:
  assumes $c' = cf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ c$
  shows $\mathfrak{N}$: c' ↦$_{CF}$ $\mathfrak{F}$: $\mathfrak{J}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$
  ⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-cone.is-cat-coneD'*

lemma (**in** *is-cat-cocone*) *is-cat-cocone-axioms'*[*cat-lim-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $c' = c$ **and** $\mathfrak{J}' = \mathfrak{J}$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{F}' = \mathfrak{F}$
  shows $\mathfrak{N}$: $\mathfrak{F}'$ >$_{CF.cocone}$ c' : $\mathfrak{J}'$ ↦↦$_{C\alpha'}$ $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *is-cat-cocone-def*[*unfolded is-cat-cocone-axioms-def*]
  |*intro is-cat-coconeI*|
  |*dest is-cat-coconeD*[*dest*!]|
  |*elim is-cat-coconeE*[*elim*!]|

lemma (**in** *is-cat-cocone*) *is-cat-coconeD'*[*cat-lim-cs-intros*]:

assumes $c' = \textit{cf-const}\ \mathfrak{J}\ \mathfrak{C}\ c$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} c' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-cocone.is-cat-coconeD'*

Duality.

lemma (**in** *is-cat-cone*) *is-cat-cocone-op*:
$\textit{op-ntcf}\ \mathfrak{N} : \textit{op-cf}\ \mathfrak{F} >_{CF.cocone} c : \textit{op-cat}\ \mathfrak{J} \mapsto\mapsto_{C\alpha} \textit{op-cat}\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *is-cat-cocone-op'*[*cat-op-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{J}' = \textit{op-cat}\ \mathfrak{J}$ **and** $\mathfrak{C}' = \textit{op-cat}\ \mathfrak{C}$ **and** $\mathfrak{F}' = \textit{op-cf}\ \mathfrak{F}$
shows $\textit{op-ntcf}\ \mathfrak{N} : \mathfrak{F}' >_{CF.cocone} c : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

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

lemma (**in** *is-cat-cocone*) *is-cat-cone-op*:
$\textit{op-ntcf}\ \mathfrak{N} : c <_{CF.cone} \textit{op-cf}\ \mathfrak{F} : \textit{op-cat}\ \mathfrak{J} \mapsto\mapsto_{C\alpha} \textit{op-cat}\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *is-cat-cone-op'*[*cat-op-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{J}' = \textit{op-cat}\ \mathfrak{J}$ **and** $\mathfrak{C}' = \textit{op-cat}\ \mathfrak{C}$ **and** $\mathfrak{F}' = \textit{op-cf}\ \mathfrak{F}$
shows $\textit{op-ntcf}\ \mathfrak{N} : c <_{CF.cone} \mathfrak{F}' : \mathfrak{J}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

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

Elementary properties.

lemma (**in** *is-cat-cone*) *cat-cone-LArr-app-is-arr*:
assumes $j \in_{\circ} \mathfrak{J}(\!|Obj|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|j|\!) : c \mapsto_{\mathfrak{C}} \mathfrak{F}(\!|ObjMap|\!)(\!|j|\!)$
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *cat-cone-LArr-app-is-arr'*[*cat-lim-cs-intros*]:
assumes $j \in_{\circ} \mathfrak{J}(\!|Obj|\!)$ **and** $\mathfrak{F}j = \mathfrak{F}(\!|ObjMap|\!)(\!|j|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|j|\!) : c \mapsto_{\mathfrak{C}} \mathfrak{F}j$
⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-cone.cat-cone-LArr-app-is-arr'*

lemma (**in** *is-cat-cocone*) *cat-cocone-LArr-app-is-arr*:
assumes $j \in_{\circ} \mathfrak{J}(\!|Obj|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|j|\!) : \mathfrak{F}(\!|ObjMap|\!)(\!|j|\!) \mapsto_{\mathfrak{C}} c$
⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *cat-cocone-LArr-app-is-arr'*[*cat-lim-cs-intros*]:
assumes $j \in_{\circ} \mathfrak{J}(\!|Obj|\!)$ **and** $\mathfrak{F}j = \mathfrak{F}(\!|ObjMap|\!)(\!|j|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|j|\!) : \mathfrak{F}j \mapsto_{\mathfrak{C}} c$
⟨*proof*⟩

lemmas [*cat-lim-cs-intros*] = *is-cat-cocone.cat-cocone-LArr-app-is-arr'*

lemma (**in** *is-cat-cone*) *cat-cone-Comp-commute*[*cat-lim-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{J}} b$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) \circ_{A\mathfrak{C}} \mathfrak{N}(\!|NTMap|\!)(\!|a|\!) = \mathfrak{N}(\!|NTMap|\!)(\!|b|\!)$
⟨*proof*⟩

lemmas [*cat-lim-cs-simps*] = *is-cat-cone.cat-cone-Comp-commute*

lemma (**in** *is-cat-cocone*) *cat-cocone-Comp-commute*[*cat-lim-cs-simps*]:
  assumes $f : a \mapsto_{\mathfrak{J}} b$
  shows $\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{C}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
  ⟨*proof*⟩

lemmas [*cat-lim-cs-simps*] = *is-cat-cocone.cat-cocone-Comp-commute*

Utilities/helper lemmas.

lemma (**in** *is-cat-cone*) *helper-cat-cone-ntcf-vcomp-Comp*:
  assumes $\mathfrak{N}' : c' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $f' : c' \mapsto_{\mathfrak{C}} c$
    and $\mathfrak{N}' = \mathfrak{N} \cdot_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f'$
    and $j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor)$
  shows $\mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f'$
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *helper-cat-cone-Comp-ntcf-vcomp*:
  assumes $\mathfrak{N}' : c' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $f' : c' \mapsto_{\mathfrak{C}} c$
    and $\bigwedge j.\ j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f'$
  shows $\mathfrak{N}' = \mathfrak{N} \cdot_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f'$
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *helper-cat-cone-Comp-ntcf-vcomp-iff*:
  assumes $\mathfrak{N}' : c' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $f' : c' \mapsto_{\mathfrak{C}} c \wedge \mathfrak{N}' = \mathfrak{N} \cdot_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \longleftrightarrow$
    $f' : c' \mapsto_{\mathfrak{C}} c \wedge (\forall j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor).\ \mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f')$
  ⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *helper-cat-cocone-ntcf-vcomp-Comp*:
  assumes $\mathfrak{N}' : \mathfrak{F} >_{CF.cocone} c' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $f' : c \mapsto_{\mathfrak{C}} c'$
    and $\mathfrak{N}' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \cdot_{NTCF} \mathfrak{N}$
    and $j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor)$
  shows $\mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = f' \circ_{A\mathfrak{C}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor)$
⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *helper-cat-cocone-Comp-ntcf-vcomp*:
  assumes $\mathfrak{N}' : \mathfrak{F} >_{CF.cocone} c' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $f' : c \mapsto_{\mathfrak{C}} c'$
    and $\bigwedge j.\ j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = f' \circ_{A\mathfrak{C}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor)$
  shows $\mathfrak{N}' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \cdot_{NTCF} \mathfrak{N}$
⟨*proof*⟩

lemma (**in** *is-cat-cocone*) *helper-cat-cocone-Comp-ntcf-vcomp-iff*:
  assumes $\mathfrak{N}' : \mathfrak{F} >_{CF.cocone} c' : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $f' : c \mapsto_{\mathfrak{C}} c' \wedge \mathfrak{N}' = ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f' \cdot_{NTCF} \mathfrak{N} \longleftrightarrow$
    $f' : c \mapsto_{\mathfrak{C}} c' \wedge (\forall j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor).\ \mathfrak{N}'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = f' \circ_{A\mathfrak{C}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor j \rfloor))$
  ⟨*proof*⟩

3.3 Limit and colimit

3.3.1 Definition and elementary properties

The concept of a limit is introduced in Chapter III-4 in [8]; the concept of a colimit is introduced in Chapter III-3 in [8].

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

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

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

syntax *-is-cat-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  ($\langle$(- :/ - $>_{CF.colim}$ - :/ - $\mapsto\mapsto_{C^1}$ -)$\rangle$ [*51*, *51*, *51*, *51*, *51*] *51*)
translations $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
  CONST is-cat-colimit α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ 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 (Δ_{CF} α $\mathfrak{J}$ $\mathfrak{C}$) (*cf-map* $\mathfrak{F}$) r (*ntcf-arrow* u)
⟨*proof*⟩

lemma (**in** *is-cat-cone*) *cat-cone-is-cat-limit*:
  assumes *universal-arrow-fo* (Δ_{CF} α $\mathfrak{J}$ $\mathfrak{C}$) (*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'*

3.3.2 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
$\exists! f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge (\forall j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor).\ u'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = u(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f')$
⟨*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 $\exists! f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \bullet_{NTCF} ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f'$
⟨*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
$\exists! f'.\ f' : r' \mapsto_{\mathfrak{C}} r \wedge (\forall j \in_{\circ} \mathfrak{J}(\lfloor Obj \rfloor).\ u'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = u(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f')$
⟨*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-arr-isomorphism*:
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-arr-isomorphism′*:
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-arr-isomorphism*:
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-arr-isomorphism′*:
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*⟩

3.3.3 Further properties

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*⟩

3.4 Finite limit and finite colimit

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

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

locale *is-cat-finite-colimit* = *is-cat-colimit* α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u + *finite-category* α $\mathfrak{J}$
for α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ r u

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

3.5 Product and coproduct

3.5.1 Definition and elementary properties

The definition of the product object is a specialization of the definition presented in Chapter III-4 in [8]. In the definition presented below, the discrete category that is used in the definition presented in [8] 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* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - <$_{CF.\prod}$ - :/ - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations $\pi : P <_{CF.\prod} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
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* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - >$_{CF.\coprod}$ - :/ - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations $\pi : A >_{CF.\coprod} U : I \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
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 $\pi : P' <_{CF.\prod} A' : I' \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*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 $\pi : A' >_{CF.\coprod} U' : I' \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*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*:
$\textit{op-ntcf}\ \pi : A >_{CF.\coprod} P : I \mapsto\mapsto_{C\alpha} \textit{op-cat}\ \mathfrak{C}$
⟨*proof*⟩

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

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

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

3.5.2 Universal property

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

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

lemma *cat-obj-prod-ex-is-arr-isomorphism*:
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** $\pi' = \pi \cdot_{NTCF} \textit{ntcf-const}\ (:_C I)\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-obj-prod-ex-is-arr-isomorphism'*:
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 $\bigwedge j.\ j \in_\circ I \Longrightarrow \pi'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = \pi(\lfloor NTMap \rfloor)(\lfloor j \rfloor) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-obj-coprod-ex-is-arr-isomorphism*:
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** $\pi' = ntcf\text{-}const\ (:_C\ I)\ \mathfrak{C}\ f \bullet_{NTCF} \pi$
⟨*proof*⟩

lemma *cat-obj-coprod-ex-is-arr-isomorphism'*:
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 $\bigwedge j.\ j \in_\circ I \Longrightarrow \pi'(\lfloor NTMap \rfloor)(\lfloor j \rfloor) = f \circ_{A\mathfrak{C}} \pi(\lfloor NTMap \rfloor)(\lfloor j \rfloor)$
⟨*proof*⟩

3.6 Finite product and finite coproduct

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

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

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

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

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 $\pi : A >_{CF.\coprod.fin} P : I \mapsto\mapsto_{C\alpha}$ *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* $\pi : A >_{CF.\coprod.fin} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*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 $\pi : U <_{CF.\prod.fin} A : I \mapsto\mapsto_{C\alpha}$ *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* $\pi : U <_{CF.\prod.fin} A : I \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

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

3.7 Product and coproduct of two objects

3.7.1 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*
(‹(- :/ - <$_{CF.\times}$ {-,-} :/ $2_C \mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations $\pi : P <_{CF.\times} \{a,b\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
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* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
($\langle$(- :/ {-,-} $>_{CF.\uplus}$ - :/ $2_C \mapsto\mapsto_{C1}$ -)$\rangle$ [51, 51, 51, 51, 51] 51)
translations $\pi : \{a,b\} >_{CF.\uplus} U : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-obj-coprod-2 α a b $\mathfrak{C}$ U π

abbreviation *proj-fst* **where** *proj-fst* $\pi \equiv$ *vpfst* ($\pi(\!|NTMap|\!)$)
abbreviation *proj-snd* **where** *proj-snd* $\pi \equiv$ *vpsnd* ($\pi(\!|NTMap|\!)$)

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 $\pi : P' <_{CF.\times} \{a',b'\} : 2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*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-fst* π' = *proj-fst* $\pi \circ_{A\mathfrak{C}} f' \wedge$ *proj-snd* π' = *proj-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-fst* π' = $f' \circ_{A\mathfrak{C}}$ *proj-fst* $\pi \wedge$ *proj-snd* π' = $f' \circ_{A\mathfrak{C}}$ *proj-snd* π)
⟨*proof*⟩

3.7.2 Universal property

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

lemma (in *is-cat-obj-coprod-2*) *cat-obj-coprod-2-unique-cocone'*:

assumes $\pi' : :\!\!\rightarrow\!\!: (2_{\mathbb{N}})\ (\mathit{if2}\ a\ b)\ \mathfrak{C} >_{CF.cocone} P' : :_C (2_{\mathbb{N}}) \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
shows
$\exists! f'.\ f' : P \mapsto_{\mathfrak{C}} P' \wedge$
$\mathit{proj\text{-}fst}\ \pi' = f' \circ_{A\mathfrak{C}} \mathit{proj\text{-}fst}\ \pi \wedge$
$\mathit{proj\text{-}snd}\ \pi' = f' \circ_{A\mathfrak{C}} \mathit{proj\text{-}snd}\ \pi$
⟨*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 $\exists! f'.\ f' : P \mapsto_{\mathfrak{C}} P' \wedge \pi' = \mathit{ntcf\text{-}const}\ (:_C (2_{\mathbb{N}}))\ \mathfrak{C}\ f' \cdot_{NTCF} \pi$
⟨*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
$\exists! f'.\ f' : P \mapsto_{\mathfrak{C}} P' \wedge$
$\mathit{proj\text{-}fst}\ \pi' = f' \circ_{A\mathfrak{C}} \mathit{proj\text{-}fst}\ \pi \wedge$
$\mathit{proj\text{-}snd}\ \pi' = f' \circ_{A\mathfrak{C}} \mathit{proj\text{-}snd}\ \pi$
⟨*proof*⟩

lemma *cat-obj-prod-2-ex-is-arr-isomorphism*:
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** $\pi' = \pi \cdot_{NTCF} \mathit{ntcf\text{-}const}\ (:_C (2_{\mathbb{N}}))\ \mathfrak{C}\ f$
⟨*proof*⟩

lemma *cat-obj-coprod-2-ex-is-arr-isomorphism*:
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** $\pi' = \mathit{ntcf\text{-}const}\ (:_C (2_{\mathbb{N}}))\ \mathfrak{C}\ f \cdot_{NTCF} \pi$
⟨*proof*⟩

3.8 Pullbacks and pushouts

3.8.1 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 [8].

locale *is-cat-pullback* =
is-cat-limit α ‹$\rightarrow\!\!\cdot\!\!\leftarrow_{C}$› $\mathfrak{C}$ ‹$\langle \mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b}\rangle_{CF\mathfrak{C}}$› $X\ x$ +
cf-scospan $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}$
for $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}\ X\ x$

syntax *-is-cat-pullback* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - <$_{CF.pb}$ -→-→-←-←- ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations $x : X <_{CF.pb} \mathfrak{a}\rightarrow\mathfrak{g}\rightarrow\mathfrak{o}\leftarrow\mathfrak{f}\leftarrow\mathfrak{b} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-pullback $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}\ X\ x$

locale *is-cat-pushout* =
is-cat-colimit α ‹$\leftarrow\!\!\cdot\!\!\rightarrow_{C}$› $\mathfrak{C}$ ‹$\langle \mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b}\rangle_{CF\mathfrak{C}}$› $X\ x$ +
cf-sspan $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}$
for $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}\ X\ x$

syntax *-is-cat-pushout* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ -←-←-→-→- >$_{CF.po}$ - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations $x : \mathfrak{a}\leftarrow\mathfrak{g}\leftarrow\mathfrak{o}\rightarrow\mathfrak{f}\rightarrow\mathfrak{b} >_{CF.po} X \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-pushout $\alpha\ \mathfrak{a}\ \mathfrak{g}\ \mathfrak{o}\ \mathfrak{f}\ \mathfrak{b}\ \mathfrak{C}\ X\ x$

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 $x : X' <_{CF.pb} \mathfrak{a}' \to \mathfrak{g}' \to \mathfrak{o}' \leftarrow \mathfrak{f}' \leftarrow \mathfrak{b}' \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*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 $x : \mathfrak{a}' \leftarrow \mathfrak{g}' \leftarrow \mathfrak{o}' \to \mathfrak{f}' \to \mathfrak{b}' >_{CF.po} X' \mapsto\mapsto_{C\alpha'} \mathfrak{C}'$
⟨*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 $x : \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha}$ *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* $x : \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} >_{CF.po} X \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

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

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

⟨*proof*⟩

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

Elementary properties.

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

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

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

lemma (**in** *is-cat-pushout*) *cat-po-cocone-span*:
shows $x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{o}_{SS} \rfloor) = x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{a}_{SS} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{g}$
and $x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{o}_{SS} \rfloor) = x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{b}_{SS} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{f}$
and $x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{a}_{SS} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{g} = x(\lfloor NTMap \rfloor)(\lfloor \mathfrak{b}_{SS} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

3.8.2 Universal property

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

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

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

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

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

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

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

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

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

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

lemma *cat-pushout-ex-is-arr-isomorphism*:

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-arr-isomorphism'*:
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 $x'(\lfloor NTMap\rfloor)(\lfloor \mathfrak{a}_{SS}\rfloor) = f \circ_{A\mathfrak{C}} x(\lfloor NTMap\rfloor)(\lfloor \mathfrak{a}_{SS}\rfloor)$
and $x'(\lfloor NTMap\rfloor)(\lfloor \mathfrak{b}_{SS}\rfloor) = f \circ_{A\mathfrak{C}} x(\lfloor NTMap\rfloor)(\lfloor \mathfrak{b}_{SS}\rfloor)$
⟨*proof*⟩

3.9 Equalizers and coequalizers

3.9.1 Definition and elementary properties

See [2][2].

locale *is-cat-equalizer* =
is-cat-limit α ‹$\uparrow\uparrow_C$ $\mathfrak{a}_{PL}$ $\mathfrak{b}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{f}_{PL}$› $\mathfrak{C}$ ‹$\uparrow\uparrow\rightarrow\uparrow\uparrow$ $\mathfrak{C}$ $\mathfrak{a}_{PL}$ $\mathfrak{b}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$› 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* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - <$_{CF.eq}$ '(-,-,-,-') :/ $\uparrow\uparrow^2{}_C \mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations $\varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow^2{}_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-equalizer α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ E ε

locale *is-cat-coequalizer* =
is-cat-colimit α ‹$\uparrow\uparrow_C$ $\mathfrak{b}_{PL}$ $\mathfrak{a}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$› $\mathfrak{C}$ ‹$\uparrow\uparrow\rightarrow\uparrow\uparrow$ $\mathfrak{C}$ $\mathfrak{b}_{PL}$ $\mathfrak{a}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$› 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* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ '(-,-,-,-') >$_{CF.coeq}$ - :/ $\uparrow\uparrow^2{}_C \mapsto\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations $\varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow^2{}_C \mapsto\mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
CONST is-cat-coequalizer α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$ 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 $\mathfrak{g}' = \mathfrak{g}$
and $\mathfrak{f}' = \mathfrak{f}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : E' <_{CF.eq} (\mathfrak{a}',\mathfrak{b}',\mathfrak{g}',\mathfrak{f}') : \uparrow\uparrow^2{}_C \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

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

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

|*elim is-cat-equalizerE*[*elim*]|

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

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 $\mathfrak{g}' = \mathfrak{g}$
    and $\mathfrak{f}' = \mathfrak{f}$
    and $\mathfrak{C}' = \mathfrak{C}$
  shows $\varepsilon : (\mathfrak{a}',\mathfrak{b}',\mathfrak{g}',\mathfrak{f}') >_{CF.coeq} E' : \uparrow\uparrow^2_C \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
  ⟨*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.

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

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

Duality.

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

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

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

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

Elementary properties.

lemma *cf-parallel-if-is-cat-cone*:
  assumes ε :
    $E <_{CF.cone} \uparrow\uparrow\to\uparrow\uparrow \mathfrak{C}\ \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f} : \uparrow\uparrow_C\ \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$

and $\mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$
shows *cf-parallel* α $\mathfrak{a}_{PL}$ $\mathfrak{b}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *cf-parallel-if-is-cat-cocone*:
assumes ε' :
$\uparrow\uparrow\rightarrow\uparrow\uparrow$ $\mathfrak{C}$ $\mathfrak{b}_{PL}$ $\mathfrak{a}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$ $>_{CF.cocone}$ E' : $\uparrow\uparrow_C$ $\mathfrak{b}_{PL}$ $\mathfrak{a}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
and $\mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$
shows *cf-parallel* α $\mathfrak{b}_{PL}$ $\mathfrak{a}_{PL}$ $\mathfrak{f}_{PL}$ $\mathfrak{g}_{PL}$ $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$ $\mathfrak{C}$
⟨*proof*⟩

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

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

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

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

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

lemma (**in** *is-cat-coequalizer*) *cat-coeq-2-eps-NTMap-app*:
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor \mathfrak{b}_{PL} \rfloor) = \varepsilon(\lfloor NTMap \rfloor)(\lfloor \mathfrak{a}_{PL} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{g}$
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor \mathfrak{b}_{PL} \rfloor) = \varepsilon(\lfloor NTMap \rfloor)(\lfloor \mathfrak{a}_{PL} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨*proof*⟩

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

lemma (**in** *is-cat-coequalizer*) *cat-coeq-Comp-eq*:

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

3.9.2 Universal property

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

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

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

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

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

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

lemma (**in** *is-cat-coequalizer*) *cat-coeq-unique*:
assumes $\varepsilon' : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow^2{}_C \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$

shows $\exists! f'.$
$f' : E \mapsto_{\mathfrak{C}} E' \wedge$
$\varepsilon' = \textit{ntcf-const}\ (\uparrow\uparrow_C\ \mathfrak{b}_{PL}\ \mathfrak{a}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{g}_{PL})\ \mathfrak{C}\ f' \cdot_{NTCF} \varepsilon$
⟨*proof*⟩

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

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

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

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

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

3.10 Projection cone

3.10.1 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* $\mathfrak{C}\ I\ F\ P\ f =$
$[(\lambda j \in_{\circ} :_C I(\lfloor Obj \rfloor).\ f\ j),\ \textit{cf-const}\ (:_C I)\ \mathfrak{C}\ P,\ :\!\!\to\!\!:\ I\ F\ \mathfrak{C},\ :_C I,\ \mathfrak{C}]_{\circ}$

Components.

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

3.10.2 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*]|

3.10.3 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-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 $\bigwedge u'\ r'.$
$[\![\ u' : r' <_{CF.cone} :\!\rightarrow\!: I\ F\ \mathfrak{C} : :_C I \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}\ ]\!] \Longrightarrow$
$\exists!f'.$
$f' : r' \mapsto_{\mathfrak{C}} P \wedge$
$u' = ntcf\text{-}obj\text{-}prod\text{-}base\ \mathfrak{C}\ I\ F\ P\ f \cdot_{NTCF} ntcf\text{-}const\ (:_C I)\ \mathfrak{C}\ f'$
shows *ntcf-obj-prod-base* $\mathfrak{C}\ I\ F\ P\ f : P <_{CF.\prod} F : I \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

3.11 Equalizer cone

3.11.1 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* $\mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ E\ e =$
$[$
$(\lambda x \in_{\circ} \uparrow\uparrow_C \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}(\!|Obj|\!).\ e\ x)$,
$cf\text{-}const\ (\uparrow\uparrow_C \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL})\ \mathfrak{C}\ E$,
$\uparrow\uparrow\!\rightarrow\!\uparrow\uparrow\ \mathfrak{C}\ \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$,
$\uparrow\uparrow_C \mathfrak{a}_{PL}\ \mathfrak{b}_{PL}\ \mathfrak{g}_{PL}\ \mathfrak{f}_{PL}$,
$\mathfrak{C}$
$]_{\circ}$

Components.

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

3.11.2 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*]|

3.11.3 Equalizer cone is a cone

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

3.12 Limits by products and equalizers

lemma *cat-limit-of-cat-prod-obj-and-cat-equalizer*:
— See Theorem 1 in Chapter V-2 in [8].
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b};\ \mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow^2{}_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
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 [8].
assumes $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a};\ \mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow^2{}_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
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*⟩

4 Completeness for categories

4.1 Small-complete category

4.1.1 Definition and elementary properties

locale *cat-small-complete* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  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* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  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* α $\mathfrak{C}$ **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* α $\mathfrak{C}$ **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*⟩

4.1.2 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*

4.1.3 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 [8]
  assumes $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b};\ \mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}\ ]\!] \Longrightarrow$
    $\exists E\ \varepsilon.\ \varepsilon : E <_{CF.eq} (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) : \uparrow\uparrow^2_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
   and $\bigwedge A\ I.$ *tm-cf-discrete* α 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 $\bigwedge \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}.\ [\![\ \mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a};\ \mathfrak{g} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}\ ]\!] \Longrightarrow$
$\exists E\ \varepsilon.\ \varepsilon : (\mathfrak{a},\mathfrak{b},\mathfrak{g},\mathfrak{f}) >_{CF.coeq} E : {\uparrow\uparrow^2}_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge A\ I.$ *tm-cf-discrete* $\alpha\ I\ A\ \mathfrak{C} \Longrightarrow \exists P\ \pi.\ \pi : A >_{CF.\amalg} P : I \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cat-small-cocomplete* $\alpha\ \mathfrak{C}$
⟨*proof*⟩

4.2 Finite-complete 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*⟩

4.3 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* $\alpha\ I\ F$ (*cat-Set* α)
⟨*proof*⟩

4.4 Product cone for the category *Set*

4.4.1 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)

Components.

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

4.4.2 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*]|

4.4.3 Product cone for the category *Set* is a universal cone

lemma (**in** $\mathcal{Z}$) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-obj-prod*:
  — See Theorem 5.2 in Chapter Introduction in [5].
  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*⟩

4.5 Equalizer for the category *Set*

4.5.1 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 $i \equiv$
    (
      $i = \mathfrak{a}_{PL}$?
        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

4.5.2 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-*$\mathfrak{a}$[*cat-Set-cs-simps*]:
assumes $x = \mathfrak{a}_{PL}$
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-*$\mathfrak{b}$[*cat-Set-cs-simps*]:
assumes $x = \mathfrak{b}_{PL}$
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*⟩

4.5.3 Equalizer for the category *Set* is an equalizer

lemma (**in** $\mathcal{Z}$) *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 $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$:
vequalizer $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{f}$ $<_{CF.eq}$ ($\mathfrak{a}$,$\mathfrak{b}$,$\mathfrak{g}$,$\mathfrak{f}$) : $\uparrow\uparrow^2_C \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

4.6 The category *Set* is small-complete

lemma (**in** $\mathcal{Z}$) *cat-small-complete-cat-Set*: *cat-small-complete* α (*cat-Set* α)
— This lemma appears as a remark on page 113 in [8].
⟨*proof*⟩

5 Adjoints

5.1 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* = $2_{\mathbb{N}}$

5.2 Definition and elementary properties

See subsection 2.1 in [3] or Chapter IV-1 in [8].

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*
  (‹(- : - ⇌$_{CF}$ - : - ⇌⇌$_{C^1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
translations Φ : $\mathfrak{F}$ ⇌$_{CF}$ $\mathfrak{G}$: $\mathfrak{C}$ ⇌⇌$_{C\alpha}$ $\mathfrak{D}$ ⇌
  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*]:
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(\!|AdjLeft|\!)$ = $\mathfrak{F}$
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(\!|AdjRight|\!)$ = $\mathfrak{G}$
  $[\mathfrak{F}, \mathfrak{G}, \varphi]_\circ(\!|AdjNT|\!)$ = φ
  ⟨*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 : $\mathfrak{F}'$ ⇌$_{CF}$ $\mathfrak{G}'$: $\mathfrak{C}'$ ⇌⇌$_{C\alpha'}$ $\mathfrak{D}'$
  ⟨*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*⟩

5.3 Opposite adjunction

5.3.1 Definition and elementary properties

See [6] for further information.

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

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

lemma *op-cf-adj-components*:
shows *op-cf-adj* $\Phi(\!|AdjLeft|\!) = op\text{-}cf\ (\Phi(\!|AdjRight|\!))$
and *op-cf-adj* $\Phi(\!|AdjRight|\!) = op\text{-}cf\ (\Phi(\!|AdjLeft|\!))$
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* $\Phi(\!|AdjLeft|\!) = op\text{-}cf\ \mathfrak{G}$
and *op-cf-adj* $\Phi(\!|AdjRight|\!) = op\text{-}cf\ \mathfrak{F}$
and *op-cf-adj* $\Phi(\!|AdjNT|\!) = inv\text{-}ntcf\ (bnt\text{-}flip\ (op\text{-}cat\ \mathfrak{C})\ \mathfrak{D}\ (\Phi(\!|AdjNT|\!)))$
⟨*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 [3].
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*

5.3.2 Alternative form of the naturality condition

The lemmas in this subsection are based on the comments on page 81 in [8].

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*⟩

5.4 Unit

5.4.1 Definition and elementary properties

See Chapter IV-1 in [8].

definition *cf-adjunction-unit* :: $V \Rightarrow V$ ($\langle\eta_C\rangle$)
where η_C Φ =

$$
\begin{array}{l}
[\\
\quad (\\
\quad\quad \lambda x \in_\circ \Phi(\!|AdjLeft|\!)(\!|HomDom|\!)(\!|Obj|\!). \\
\quad\quad\quad (\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ \Phi(\!|AdjLeft|\!)(\!|ObjMap|\!)(\!|x|\!)|\!)_\bullet)(\!|ArrVal|\!)(\!| \\
\quad\quad\quad\quad \Phi(\!|AdjLeft|\!)(\!|HomCod|\!)(\!|CId|\!)(\Phi(\!|AdjLeft|\!)(\!|ObjMap|\!)(\!|x|\!)) \\
\quad\quad\quad |\!) \\
\quad), \\
\quad cf\text{-}id\ (\Phi(\!|AdjLeft|\!)(\!|HomDom|\!)), \\
\quad (\Phi(\!|AdjRight|\!))\ \circ_{CF}\ (\Phi(\!|AdjLeft|\!)), \\
\quad \Phi(\!|AdjLeft|\!)(\!|HomDom|\!), \\
\quad \Phi(\!|AdjLeft|\!)(\!|HomDom|\!) \\
]_\circ
\end{array}
$$

Components.

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

context *is-cf-adjunction*
begin

lemma *cf-adjunction-unit-components′*:
shows $\eta_C\ \Phi(\!|NTMap|\!) =$
$$
\begin{array}{l}
\quad (\\
\quad\quad \lambda x \in_\circ \mathfrak{C}(\!|Obj|\!). \\
\quad\quad\quad (\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ \mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)|\!)_\bullet)(\!|ArrVal|\!)(\!|\mathfrak{D}(\!|CId|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!)|\!)|\!) \\
\quad)
\end{array}
$$
and $\eta_C\ \Phi(\!|NTDom|\!) = cf\text{-}id\ \mathfrak{C}$
and $\eta_C\ \Phi(\!|NTCod|\!) = \mathfrak{G}\ \circ_{CF}\ \mathfrak{F}$
and $\eta_C\ \Phi(\!|NTDGDom|\!) = \mathfrak{C}$
and $\eta_C\ \Phi(\!|NTDGCod|\!) = \mathfrak{C}$
⟨*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

5.4.2 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*⟩

5.4.3 Unit is a natural transformation

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

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-is-ntcf′*:
assumes $\mathfrak{S} = \mathit{cf\text{-}id}\ \mathfrak{C}$
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′*

5.4.4 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 [8].

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 $\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ a|\!)_{\bullet} =$
umap-of $\mathfrak{G}\ x\ (\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))\ (\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!))\ a$
(**is** ‹$\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ a|\!)_{\bullet} = ?uof\text{-}a$›)
⟨*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 $\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ a|\!)_{\bullet} = \mathit{umap\text{-}of}\ \mathfrak{G}\ x\ \mathfrak{F}x\ \eta\ a$
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-component-is-ua-of*:
assumes $x \in_{\circ} \mathfrak{C}(\!|Obj|\!)$

shows *universal-arrow-of* $\mathfrak{G}$ x $(\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))$ $(\eta_C\ \Phi(\!|NTMap|\!)(\!|x|\!))$
(**is** ‹*universal-arrow-of* $\mathfrak{G}$ x $(\mathfrak{F}(\!|ObjMap|\!)(\!|x|\!))$ *?ηx*›)
⟨*proof*⟩

5.5 Counit

5.5.1 Definition and elementary properties

definition *cf-adjunction-counit* :: $V \Rightarrow V$ (‹ε_C›)
where $\varepsilon_C\ \Phi$ =
[
(
$\lambda x \in_\circ \Phi(\!|AdjLeft|\!)(\!|HomCod|\!)(\!|Obj|\!).$
$(\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\Phi(\!|AdjRight|\!)(\!|ObjMap|\!)(\!|x|\!),\ x|\!|_\bullet)^{-1}{}_{Set}(\!|ArrVal|\!)(\!|$
$\Phi(\!|AdjLeft|\!)(\!|HomDom|\!)(\!|CId|\!)(\Phi(\!|AdjRight|\!)(\!|ObjMap|\!)(\!|x|\!))$
$|\!|$
),
$(\Phi(\!|AdjLeft|\!))\ \circ_{CF}\ (\Phi(\!|AdjRight|\!))$,
cf-id $(\Phi(\!|AdjLeft|\!)(\!|HomCod|\!))$,
$\Phi(\!|AdjLeft|\!)(\!|HomCod|\!)$,
$\Phi(\!|AdjLeft|\!)(\!|HomCod|\!)$
]$_\circ$

Components.

lemma *cf-adjunction-counit-components*:
shows $\varepsilon_C\ \Phi(\!|NTMap|\!)$ =
(
$\lambda x \in_\circ \Phi(\!|AdjLeft|\!)(\!|HomCod|\!)(\!|Obj|\!).$
$(\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\Phi(\!|AdjRight|\!)(\!|ObjMap|\!)(\!|x|\!),\ x|\!|_\bullet)^{-1}{}_{Set}(\!|ArrVal|\!)(\!|$
$\Phi(\!|AdjLeft|\!)(\!|HomDom|\!)(\!|CId|\!)(\Phi(\!|AdjRight|\!)(\!|ObjMap|\!)(\!|x|\!))$
$|\!|$
)
and $\varepsilon_C\ \Phi(\!|NTDom|\!) = (\Phi(\!|AdjLeft|\!))\ \circ_{CF}\ (\Phi(\!|AdjRight|\!))$
and $\varepsilon_C\ \Phi(\!|NTCod|\!) =$ *cf-id* $(\Phi(\!|AdjLeft|\!)(\!|HomCod|\!))$
and $\varepsilon_C\ \Phi(\!|NTDGDom|\!) = \Phi(\!|AdjLeft|\!)(\!|HomCod|\!)$
and $\varepsilon_C\ \Phi(\!|NTDGCod|\!) = \Phi(\!|AdjLeft|\!)(\!|HomCod|\!)$
⟨*proof*⟩

context *is-cf-adjunction*
begin

lemma *cf-adjunction-counit-components′*:
shows $\varepsilon_C\ \Phi(\!|NTMap|\!)$ =
(
$\lambda x \in_\circ \mathfrak{D}(\!|Obj|\!).$
$(\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\mathfrak{G}(\!|ObjMap|\!)(\!|x|\!),\ x|\!|_\bullet)^{-1}{}_{Set}(\!|ArrVal|\!)(\mathfrak{C}(\!|CId|\!)(\mathfrak{G}(\!|ObjMap|\!)(\!|x|\!))|\!|)$
)
and $\varepsilon_C\ \Phi(\!|NTDom|\!) = \mathfrak{F} \circ_{CF} \mathfrak{G}$
and $\varepsilon_C\ \Phi(\!|NTCod|\!) =$ *cf-id* $\mathfrak{D}$
and $\varepsilon_C\ \Phi(\!|NTDGDom|\!) = \mathfrak{D}$
and $\varepsilon_C\ \Phi(\!|NTDGCod|\!) = \mathfrak{D}$
⟨*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

5.5.2 Duality for the unit and counit

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-NTMap-op*:
  $\eta_C\ (op\text{-}cf\text{-}adj\ \Phi)(\lfloor NTMap \rfloor) = \varepsilon_C\ \Phi(\lfloor NTMap \rfloor)$
⟨*proof*⟩

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

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-NTMap-op*:
  $\varepsilon_C\ (op\text{-}cf\text{-}adj\ \Phi)(\lfloor NTMap \rfloor) = \eta_C\ \Phi(\lfloor NTMap \rfloor)$
  ⟨*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 $(\varepsilon_C\ \Phi) = \eta_C\ (op\text{-}cf\text{-}adj\ \Phi)$
  (**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 $(\eta_C\ \Phi) = \varepsilon_C\ (op\text{-}cf\text{-}adj\ \Phi)$
  (**is** ‹*?η* = *?ε*›)
⟨*proof*⟩

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

5.5.3 Natural transformation map

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

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

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

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

5.5.4 Counit is a natural transformation

lemma (**in** *is-cf-adjunction*) *cf-adjunction-counit-is-ntcf*:
$\varepsilon_C\ \Phi : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \textit{cf-id}\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*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'*

5.5.5 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 [8].

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 *op-cf-adj* $\Phi(\!|AdjNT|\!)(\!|NTMap|\!)(\!|x,\ a|\!)_\bullet =$
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 *universal-arrow-fo* $\mathfrak{F}\ x\ (\mathfrak{G}(\!|ObjMap|\!)(\!|x|\!))\ (\varepsilon_C\ \Phi(\!|NTMap|\!)(\!|x|\!))$
⟨*proof*⟩

5.6 Counit-unit equations

The following equations appear as part of the statement of Theorem 1 in Chapter IV-1 in [8]. 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** ‹$(\mathfrak{G} \circ_{CF-NTCF} ?\varepsilon) \bullet_{NTCF} (?\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$›)
⟨*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** ‹$(?\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \bullet_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} ?\eta) = \textit{ntcf-id}\ \mathfrak{F}$›)
⟨*proof*⟩

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

5.7 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 [8].

5.7.1 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* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta$ =
[
$(\lambda cd\in_\circ(op\text{-}cat\ (\mathfrak{F}(\lfloor HomDom\rfloor)) \times_C \mathfrak{F}(\lfloor HomCod\rfloor))(\lfloor Obj\rfloor).$
$umap\text{-}of\ \mathfrak{G}\ (cd(\lfloor 0\rfloor))\ (\mathfrak{F}(\lfloor ObjMap\rfloor)(\lfloor cd(\lfloor 0\rfloor)\rfloor))\ (\eta(\lfloor NTMap\rfloor)(\lfloor cd(\lfloor 0\rfloor)\rfloor))\ (cd(\lfloor 1_{\mathbb{N}}\rfloor))),$
$Hom_{O.C\alpha}\mathfrak{F}(\lfloor HomCod\rfloor)(\mathfrak{F}-,-),$
$Hom_{O.C\alpha}\mathfrak{F}(\lfloor HomDom\rfloor)(-,\mathfrak{G}-),$
$op\text{-}cat\ (\mathfrak{F}(\lfloor HomDom\rfloor)) \times_C (\mathfrak{F}(\lfloor HomCod\rfloor)),$
$cat\text{-}Set\ \alpha$
]$_\circ$

Components.

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

5.7.2 Natural transformation map

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vsv*[*adj-cs-intros*]:
$vsv\ (cf\text{-}adjunction\text{-}AdjNT\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\lfloor NTMap\rfloor))$
⟨*proof*⟩

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

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

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

5.7.3 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 $\mathfrak{F}$ $\mathfrak{G}$ $\eta : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

5.7.4 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* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
and η_C (*cf-adjunction-of-unit* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta$) = η
⟨*proof*⟩

5.8 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 [8].

5.8.1 Left adjoint

definition *cf-la-of-ra* :: $(V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-la-of-ra* $F\ \mathfrak{G}\ \eta$ =

```
[
  (λx∈∘𝔊(|HomCod|)(|Obj|). F x),
  (
    λh∈∘𝔊(|HomCod|)(|Arr|). THE f'.
      f' : F (𝔊(|HomCod|)(|Dom|)(|h|)) ↦_𝔊(|HomDom|) F (𝔊(|HomCod|)(|Cod|)(|h|)) ∧
        η(|NTMap|)(|𝔊(|HomCod|)(|Cod|)(|h|)|) ∘_A𝔊(|HomCod|) h =
          (
            umap-of
              𝔊
              (𝔊(|HomCod|)(|Dom|)(|h|))
              (F (𝔊(|HomCod|)(|Dom|)(|h|)))
              (η(|NTMap|)(|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* $F\ \mathfrak{G}\ \eta(\!|ObjMap|\!)$ = $(\lambda x \in_\circ \mathfrak{G}(\!|HomCod|\!)(\!|Obj|\!).\ F\ x)$
and *cf-la-of-ra* $F\ \mathfrak{G}\ \eta(\!|ArrMap|\!)$ =

```
  (
    λh∈∘𝔊(|HomCod|)(|Arr|). THE f'.
      f' : F (𝔊(|HomCod|)(|Dom|)(|h|)) ↦_𝔊(|HomDom|) F (𝔊(|HomCod|)(|Cod|)(|h|)) ∧
      η(|NTMap|)(|𝔊(|HomCod|)(|Cod|)(|h|)|) ∘_A𝔊(|HomCod|) h =
        (
          umap-of
            𝔊
            (𝔊(|HomCod|)(|Dom|)(|h|))
            (F (𝔊(|HomCod|)(|Dom|)(|h|)))
            (η(|NTMap|)(|𝔊(|HomCod|)(|Dom|)(|h|)|))
            (F (𝔊(|HomCod|)(|Cod|)(|h|)))
        )(|ArrVal|)(|f'|)
  )
```

and *cf-la-of-ra* $F\ \mathfrak{G}\ \eta(\!|HomDom|\!)$ = $\mathfrak{G}(\!|HomCod|\!)$
and *cf-la-of-ra* $F\ \mathfrak{G}\ \eta(\!|HomCod|\!)$ = $\mathfrak{G}(\!|HomDom|\!)$
⟨*proof*⟩

5.8.2 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** *?*$\mathfrak{G}$=$\mathfrak{F}$, *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

5.8.3 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** *?*$\mathfrak{G}$=$\mathfrak{F}$, *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 $h : a \mapsto_{\mathfrak{B}} b$
shows
$cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{F}\ \eta(\lfloor ArrMap \rfloor)(\lfloor h \rfloor) =$
(
THE f'.
$f' : F\ a \mapsto_{\mathfrak{A}} F\ b \wedge$
$\eta(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} h = umap\text{-}of\ \mathfrak{F}\ a\ (F\ a)\ (\eta(\lfloor NTMap \rfloor)(\lfloor a \rfloor))\ (F\ b)(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor)$
)
⟨*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 $universal\text{-}arrow\text{-}of\ \mathfrak{G}\ a\ (cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ObjMap \rfloor)(\lfloor a \rfloor))\ (\eta(\lfloor NTMap \rfloor)(\lfloor a \rfloor))$
and $universal\text{-}arrow\text{-}of\ \mathfrak{G}\ b\ (cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ObjMap \rfloor)(\lfloor b \rfloor))\ (\eta(\lfloor NTMap \rfloor)(\lfloor b \rfloor))$
shows $cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : F\ a \mapsto_{\mathfrak{D}} F\ b$
and $\eta(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{C}} f = umap\text{-}of$
$\mathfrak{G}\ a\ (F\ a)\ (\eta(\lfloor NTMap \rfloor)(\lfloor a \rfloor))\ (F\ b)(\lfloor ArrVal \rfloor)(\lfloor cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \rfloor)$
and $\bigwedge f'$.
⟦
$f' : F\ a \mapsto_{\mathfrak{D}} F\ b$;
$\eta(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{C}} f = umap\text{-}of\ \mathfrak{G}\ a\ (F\ a)\ (\eta(\lfloor NTMap \rfloor)(\lfloor a \rfloor))\ (F\ b)(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor)$
⟧ ⟹ $cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = f'$
⟨*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 $universal\text{-}arrow\text{-}of\ \mathfrak{G}\ a\ (cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ObjMap \rfloor)(\lfloor a \rfloor))\ (\eta(\lfloor NTMap \rfloor)(\lfloor a \rfloor))$
and $universal\text{-}arrow\text{-}of\ \mathfrak{G}\ b\ (cf\text{-}la\text{-}of\text{-}ra\ F\ \mathfrak{G}\ \eta(\lfloor ObjMap \rfloor)(\lfloor b \rfloor))\ (\eta(\lfloor NTMap \rfloor)(\lfloor b \rfloor))$
and $Fa = F\ a$
and $Fb = F\ b$

shows *cf-la-of-ra* F $\mathfrak{G}$ $\eta(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : Fa \mapsto_{\mathfrak{D}} Fb$
⟨*proof*⟩

5.8.4 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 $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow F\ c \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor)$
and $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-of $\mathfrak{G}$ c (*cf-la-of-ra* F $\mathfrak{G}$ $\eta(\lfloor ObjMap \rfloor)(\lfloor c \rfloor))$ $(\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor))$
and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
$\mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor$*cf-la-of-ra* F $\mathfrak{G}$ $\eta(\lfloor ArrMap \rfloor)(\lfloor h \rfloor)\rfloor) \circ_{A\mathfrak{C}} (\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor)) =$
$(\eta(\lfloor NTMap \rfloor)(\lfloor c' \rfloor)) \circ_{A\mathfrak{C}} h$
shows *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 $\mathfrak{G}$ η
defines $\mathfrak{F} \equiv$ *cf-la-of-ra* F $\mathfrak{G}$ η
assumes $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow F\ c \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor)$
and $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-of $\mathfrak{G}$ c $(\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor))$ $(\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor))$
and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
$\mathfrak{G}(\lfloor ArrMap \rfloor)(\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor)) \circ_{A\mathfrak{C}} (\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor)) = (\eta(\lfloor NTMap \rfloor)(\lfloor c' \rfloor)) \circ_{A\mathfrak{C}} h$
and *vfsequence* η
and *vcard* $\eta = 5_{\mathbb{N}}$
and $\eta(\lfloor NTDom \rfloor) =$ *cf-id* $\mathfrak{C}$
and $\eta(\lfloor NTCod \rfloor) = \mathfrak{G} \circ_{CF} \mathfrak{F}$
and $\eta(\lfloor NTDGDom \rfloor) = \mathfrak{C}$
and $\eta(\lfloor NTDGCod \rfloor) = \mathfrak{C}$
and *vsv* $(\eta(\lfloor NTMap \rfloor))$
and $\mathcal{D}_{\circ}(\eta(\lfloor NTMap \rfloor)) = \mathfrak{C}(\lfloor Obj \rfloor)$
shows η : *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 $\mathfrak{G}$ η
defines $\mathfrak{F} \equiv$ *cf-la-of-ra* F $\mathfrak{G}$ η
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow F\ c \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor)$
and $\bigwedge c.\ c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-of $\mathfrak{G}$ c $(\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor))$ $(\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor))$
and $\bigwedge c\ c'\ h.\ h : c \mapsto_{\mathfrak{C}} c' \Longrightarrow$
$\mathfrak{G}(\lfloor ArrMap \rfloor)(\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor)) \circ_{A\mathfrak{C}} (\eta(\lfloor NTMap \rfloor)(\lfloor c \rfloor)) = (\eta(\lfloor NTMap \rfloor)(\lfloor c' \rfloor)) \circ_{A\mathfrak{C}} h$
and *vfsequence* η
and *vcard* $\eta = 5_{\mathbb{N}}$
and $\eta(\lfloor NTDom \rfloor) =$ *cf-id* $\mathfrak{C}$
and $\eta(\lfloor NTCod \rfloor) = \mathfrak{G} \circ_{CF} \mathfrak{F}$
and $\eta(\lfloor NTDGDom \rfloor) = \mathfrak{C}$
and $\eta(\lfloor NTDGCod \rfloor) = \mathfrak{C}$
and *vsv* $(\eta(\lfloor NTMap \rfloor))$
and $\mathcal{D}_{\circ}(\eta(\lfloor NTMap \rfloor)) = \mathfrak{C}(\lfloor Obj \rfloor)$
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) $= \eta$

⟨*proof*⟩

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

5.9.1 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 [8].

definition *cf-adjunction-of-counit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ ε =
    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* α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon(\!|AdjLeft|\!)$ = *op-cf* (*op-cf* $\mathfrak{F}$)
    and *cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon(\!|AdjRight|\!)$ = *op-cf* (*op-cf* $\mathfrak{G}$)
    and *cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon(\!|AdjNT|\!)$ = *op-cf-adj-nt*
      (*op-cf* $\mathfrak{G}(\!|HomDom|\!)$)
      (*op-cf* $\mathfrak{G}(\!|HomCod|\!)$)
      (*cf-adjunction-AdjNT-of-unit* α (*op-cf* $\mathfrak{G}$) (*op-cf* $\mathfrak{F}$) (*op-ntcf* ε))
  ⟨*proof*⟩

5.9.2 Natural transformation map

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

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

lemma *cf-adjunction-of-counit-is-cf-adjunction*:
  assumes *category* 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 $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF}$ *cf-id* $\mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $\bigwedge x.\ x \in_\circ \mathfrak{D}(\!|Obj|\!) \Longrightarrow$ *universal-arrow-fo* $\mathfrak{F}$ x ($\mathfrak{G}(\!|ObjMap|\!)(\!|x|\!)$) ($\varepsilon(\!|NTMap|\!)(\!|x|\!)$)
  shows *cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
    and ε_C (*cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ ε) = ε
    and $\mathcal{D}_\circ$ (*cf-adjunction-of-counit* α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon(\!|AdjNT|\!)(\!|NTMap|\!)$) =
      (*op-cat* $\mathfrak{C} \times_C \mathfrak{D}$)$(\!|Obj|\!)$
    and $\bigwedge c\ d.$ $[\![\ c \in_\circ \mathfrak{C}(\!|Obj|\!);\ d \in_\circ \mathfrak{D}(\!|Obj|\!)\ ]\!] \Longrightarrow$
      cf-adjunction-of-counit α $\mathfrak{F}$ $\mathfrak{G}$ $\varepsilon(\!|AdjNT|\!)(\!|NTMap|\!)(\!|c,\ d|\!)_\bullet$ =
        (*umap-fo* $\mathfrak{F}$ d ($\mathfrak{G}(\!|ObjMap|\!)(\!|d|\!)$) ($\varepsilon(\!|NTMap|\!)(\!|d|\!)$) c)$^{-1}{}_{Set}$
⟨*proof*⟩

5.10 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 [8].

5.10.1 Definition and elementary properties

definition *cf-ra-of-la* :: $(V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-ra-of-la F* $\mathfrak{F}$ ε = *op-cf* (*cf-la-of-ra F* (*op-cf* $\mathfrak{F}$) (*op-ntcf* ε))

5.10.2 Object map

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

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

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

5.10.3 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 *universal-arrow-fo* $\mathfrak{F}$ *a* (*cf-ra-of-la F* $\mathfrak{F}$ ε(|*ObjMap*|)(|*a*|)) (ε(|*NTMap*|)(|*a*|))
    and *universal-arrow-fo* $\mathfrak{F}$ *b* (*cf-ra-of-la F* $\mathfrak{F}$ ε(|*ObjMap*|)(|*b*|)) (ε(|*NTMap*|)(|*b*|))
  shows *cf-ra-of-la F* $\mathfrak{F}$ ε(|*ArrMap*|)(|*f*|) : *F a* $\mapsto_{\mathfrak{C}}$ *F b*
    and $f \circ_{A\mathfrak{D}} \varepsilon$(|*NTMap*|)(|*a*|) =
      umap-fo $\mathfrak{F}$ *b* (*F b*) (ε(|*NTMap*|)(|*b*|)) (*F a*)(|*ArrVal*|)(|*cf-ra-of-la F* $\mathfrak{F}$ ε(|*ArrMap*|)(|*f*|)|)
    and $\bigwedge f'$.
      ⟦
        $f' : F\ a \mapsto_{\mathfrak{C}} F\ b$;
        $f \circ_{A\mathfrak{D}} \varepsilon$(|*NTMap*|)(|*a*|) = *umap-fo* $\mathfrak{F}$ *b* (*F b*) (ε(|*NTMap*|)(|*b*|)) (*F a*)(|*ArrVal*|)(|*f'*|)
      ⟧ ⟹ *cf-ra-of-la F* $\mathfrak{F}$ ε(|*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* $\mathfrak{F}$ ε(|*ObjMap*|)(|*a*|)) (ε(|*NTMap*|)(|*a*|))
    and *universal-arrow-fo* $\mathfrak{F}$ *b* (*cf-ra-of-la F* $\mathfrak{F}$ ε(|*ObjMap*|)(|*b*|)) (ε(|*NTMap*|)(|*b*|))
    and *Fa* = *F a*
    and *Fb* = *F b*
  shows *cf-ra-of-la F* $\mathfrak{F}$ ε(|*ArrMap*|)(|*f*|) : *Fa* $\mapsto_{\mathfrak{C}}$ *Fb*
  ⟨*proof*⟩

5.10.4 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* $\mathfrak{F}$) (*op-ntcf* 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 $\bigwedge d.\ d \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}\ d\ (cf\text{-}ra\text{-}of\text{-}la\ F\ \mathfrak{F}\ \varepsilon(\lfloor ObjMap \rfloor)(\lfloor d \rfloor))\ (\varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor))$
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor d' \rfloor) \circ_{A\mathfrak{D}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor cf\text{-}ra\text{-}of\text{-}la\ F\ \mathfrak{F}\ \varepsilon(\lfloor ArrMap \rfloor)(\lfloor h \rfloor)) =$
$h \circ_{A\mathfrak{D}} \varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor)$
and $h : c' \mapsto_{\mathfrak{D}} c$
shows $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor cf\text{-}ra\text{-}of\text{-}la\ F\ \mathfrak{F}\ \varepsilon(\lfloor ArrMap \rfloor)(\lfloor h \rfloor)) \circ_{A\ op\text{-}cat\ \mathfrak{D}} \varepsilon(\lfloor NTMap \rfloor)(\lfloor c \rfloor) =$
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor c' \rfloor) \circ_{A\ op\text{-}cat\ \mathfrak{D}} h$
⟨*proof*⟩

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

lemma *cf-ra-of-la-is-ntcf*:
fixes $F\ \mathfrak{F}\ \varepsilon$
defines $\mathfrak{G} \equiv cf\text{-}ra\text{-}of\text{-}la\ F\ \mathfrak{F}\ \varepsilon$
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor) \Longrightarrow F\ d \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$
and $\bigwedge d.\ d \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}\ d\ (\mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor d \rfloor))\ (\varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor))$
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor d' \rfloor) \circ_{A\mathfrak{D}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor) \rfloor) = h \circ_{A\mathfrak{D}} \varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor)$
and *vfsequence* ε
and $vcard\ \varepsilon = 5_{\mathbb{N}}$
and $\varepsilon(\lfloor NTDom \rfloor) = \mathfrak{F} \circ_{CF} \mathfrak{G}$
and $\varepsilon(\lfloor NTCod \rfloor) = cf\text{-}id\ \mathfrak{D}$
and $\varepsilon(\lfloor NTDGDom \rfloor) = \mathfrak{D}$
and $\varepsilon(\lfloor NTDGCod \rfloor) = \mathfrak{D}$
and $vsv\ (\varepsilon(\lfloor NTMap \rfloor))$
and $\mathcal{D}_{\circ}\ (\varepsilon(\lfloor NTMap \rfloor)) = \mathfrak{D}(\lfloor Obj \rfloor)$
shows $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} cf\text{-}id\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-ra-of-la-is-counit*:
fixes $F\ \mathfrak{F}\ \varepsilon$
defines $\mathfrak{G} \equiv cf\text{-}ra\text{-}of\text{-}la\ F\ \mathfrak{F}\ \varepsilon$
assumes $category\ \alpha\ \mathfrak{C}$
and $category\ \alpha\ \mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge d.\ d \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor) \Longrightarrow F\ d \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$
and $\bigwedge d.\ d \in_{\circ} \mathfrak{D}(\lfloor Obj \rfloor) \Longrightarrow$
universal-arrow-fo $\mathfrak{F}\ d\ (\mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor d \rfloor))\ (\varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor))$
and $\bigwedge d\ d'\ h.\ h : d \mapsto_{\mathfrak{D}} d' \Longrightarrow$
$\varepsilon(\lfloor NTMap \rfloor)(\lfloor d' \rfloor) \circ_{A\mathfrak{D}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor) \rfloor) = h \circ_{A\mathfrak{D}} \varepsilon(\lfloor NTMap \rfloor)(\lfloor d \rfloor)$
and *vfsequence* ε
and $vcard\ \varepsilon = 5_{\mathbb{N}}$
and $\varepsilon(\lfloor NTDom \rfloor) = \mathfrak{F} \circ_{CF} \mathfrak{G}$
and $\varepsilon(\lfloor NTCod \rfloor) = cf\text{-}id\ \mathfrak{D}$

and $\varepsilon(\!|NTDGDom|\!) = \mathfrak{D}$
and $\varepsilon(\!|NTDGCod|\!) = \mathfrak{D}$
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 $\varepsilon_C\ (\textit{cf-adjunction-of-counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \varepsilon$
⟨*proof*⟩

5.11 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 [8].

lemma *counit-unit-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 $\eta : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \textit{cf-id}\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \cdot_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = \textit{ntcf-id}\ \mathfrak{F}$
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 $\eta_C\ (\textit{cf-adjunction-of-unit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \eta$
and $\varepsilon_C\ (\textit{cf-adjunction-of-unit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \varepsilon$
⟨*proof*⟩

lemma *counit-unit-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 $\eta : \textit{cf-id}\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \textit{cf-id}\ \mathfrak{D} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \cdot_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = \textit{ntcf-id}\ \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = \textit{ntcf-id}\ \mathfrak{F}$
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 $\eta_C\ (\textit{cf-adjunction-of-counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \eta$
and $\varepsilon_C\ (\textit{cf-adjunction-of-counit}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon) = \varepsilon$
⟨*proof*⟩

5.12 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 [8]. However, similar results can also be found in section 4 in [13] and in subsection 2.1 in [3].

5.12.1 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* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi =$
[
(
$\lambda x \in_\circ \mathfrak{C}(\!|Obj|\!).\ \textit{THE}\ f'.$
let
$\eta = \eta_C\ \Phi$;

```
      η′ = η_C Ψ;
      𝔉x = 𝔉(⦇ObjMap⦈)(⦇x⦈);
      𝔉′x = 𝔉′(⦇ObjMap⦈)(⦇x⦈)
    in
      f′ : 𝔉x ↦_𝔇 𝔉′x ∧
      η′(⦇NTMap⦈)(⦇x⦈) = umap-of 𝔊 x (𝔉x) (η(⦇NTMap⦈)(⦇x⦈)) (𝔉′x)(⦇ArrVal⦈)(⦇f′⦈)
  ),
  𝔉,
  𝔉′,
  ℭ,
  𝔇
]∘

definition cf-adj-RL-iso :: V ⇒ V ⇒ V ⇒ V ⇒ V ⇒ V ⇒ V ⇒ V
  where cf-adj-RL-iso ℭ 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ =
  [
    (
      λx∈∘𝔇(⦇Obj⦈). THE f′.
      let
        ε = ε_C Φ;
        ε′ = ε_C Ψ;
        𝔊x = 𝔊(⦇ObjMap⦈)(⦇x⦈);
        𝔊′x = 𝔊′(⦇ObjMap⦈)(⦇x⦈)
      in
        f′ : 𝔊′x ↦_ℭ 𝔊x ∧
        ε′(⦇NTMap⦈)(⦇x⦈) = umap-fo 𝔉 x 𝔊x (ε(⦇NTMap⦈)(⦇x⦈)) 𝔊′x(⦇ArrVal⦈)(⦇f′⦈)
    ),
    𝔊′,
    𝔊,
    𝔇,
    ℭ
  ]∘
```

Components.

```
lemma cf-adj-LR-iso-components:
  shows cf-adj-LR-iso ℭ 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ(⦇NTMap⦈) =
    (
      λx∈∘ℭ(⦇Obj⦈). THE f′.
      let
        η = η_C Φ;
        η′ = η_C Ψ;
        𝔉x = 𝔉(⦇ObjMap⦈)(⦇x⦈);
        𝔉′x = 𝔉′(⦇ObjMap⦈)(⦇x⦈)
      in
        f′ : 𝔉x ↦_𝔇 𝔉′x ∧
        η′(⦇NTMap⦈)(⦇x⦈) = umap-of 𝔊 x 𝔉x (η(⦇NTMap⦈)(⦇x⦈)) 𝔉′x(⦇ArrVal⦈)(⦇f′⦈)
    )
    and [adj-cs-simps]: cf-adj-LR-iso ℭ 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ(⦇NTDom⦈) = 𝔉
    and [adj-cs-simps]: cf-adj-LR-iso ℭ 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ(⦇NTCod⦈) = 𝔉′
    and [adj-cs-simps]: cf-adj-LR-iso ℭ 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ(⦇NTDGDom⦈) = ℭ
    and [adj-cs-simps]: cf-adj-LR-iso ℭ 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ(⦇NTDGCod⦈) = 𝔇
  ⟨proof⟩

lemma cf-adj-RL-iso-components:
  shows cf-adj-RL-iso ℭ 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ(⦇NTMap⦈) =
    (
      λx∈∘𝔇(⦇Obj⦈). THE f′.
      let
```

$\varepsilon = \varepsilon_C\ \Phi$;
$\varepsilon' = \varepsilon_C\ \Psi$;
$\mathfrak{G}x = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$;
$\mathfrak{G}'x = \mathfrak{G}'(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$
in
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x \wedge$
$\varepsilon'(\lfloor NTMap \rfloor)(\lfloor x \rfloor) = umap\text{-}fo\ \mathfrak{F}\ x\ \mathfrak{G}x\ (\varepsilon(\lfloor NTMap \rfloor)(\lfloor x \rfloor))\ \mathfrak{G}'x(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor)$
)
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTDom \rfloor) = \mathfrak{G}'$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTCod \rfloor) = \mathfrak{G}$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTDGDom \rfloor) = \mathfrak{D}$
and [*adj-cs-simps*]: *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTDGCod \rfloor) = \mathfrak{C}$
⟨*proof*⟩

5.12.2 Natural transformation map

lemma *cf-adj-LR-iso-vsv*[*adj-cs-intros*]:
vsv (*cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi(\lfloor NTMap \rfloor)$)
⟨*proof*⟩

lemma *cf-adj-RL-iso-vsv*[*adj-cs-intros*]:
vsv (*cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTMap \rfloor)$)
⟨*proof*⟩

lemma *cf-adj-LR-iso-vdomain*[*adj-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi(\lfloor NTMap \rfloor)$) = $\mathfrak{C}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *cf-adj-RL-iso-vdomain*[*adj-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTMap \rfloor)$) = $\mathfrak{D}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *cf-adj-LR-iso-app*:
fixes $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi$
assumes $x \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
defines $\mathfrak{F}x \equiv \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$
and $\mathfrak{F}'x \equiv \mathfrak{F}'(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$
and $\eta \equiv \eta_C\ \Phi$
and $\eta' \equiv \eta_C\ \Psi$
shows *cf-adj-LR-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi(\lfloor NTMap \rfloor)(\lfloor x \rfloor) =$
(
THE f'.
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
$\eta'(\lfloor NTMap \rfloor)(\lfloor x \rfloor) = umap\text{-}of\ \mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(\lfloor NTMap \rfloor)(\lfloor x \rfloor))\ \mathfrak{F}'x(\lfloor ArrVal \rfloor)(\lfloor f' \rfloor)$
)
⟨*proof*⟩

lemma *cf-adj-RL-iso-app*:
fixes $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi$
assumes $x \in_\circ \mathfrak{D}(\lfloor Obj \rfloor)$
defines $\mathfrak{G}x \equiv \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$
and $\mathfrak{G}'x \equiv \mathfrak{G}'(\lfloor ObjMap \rfloor)(\lfloor x \rfloor)$
and $\varepsilon \equiv \varepsilon_C\ \Phi$
and $\varepsilon' \equiv \varepsilon_C\ \Psi$
shows *cf-adj-RL-iso* $\mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi(\lfloor NTMap \rfloor)(\lfloor x \rfloor) =$
(
THE f'.
$f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x \wedge$

$\varepsilon'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}fo\ \mathfrak{F}\ x\ \mathfrak{G}x\ (\varepsilon(\!|NTMap|\!)(\!|x|\!))\ \mathfrak{G}'x(\!|ArrVal|\!)(\!|f'|\!)$
)
⟨*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 $f \equiv cf\text{-}adj\text{-}LR\text{-}iso\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi(\!|NTMap|\!)(\!|x|\!)$
shows
$\exists!f'.$
$f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
$\eta'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}of\ \mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(\!|NTMap|\!)(\!|x|\!))\ \mathfrak{F}'x(\!|ArrVal|\!)(\!|f'|\!)$
$f : \mathfrak{F}x \mapsto_{iso\mathfrak{D}} \mathfrak{F}'x$
$\eta'(\!|NTMap|\!)(\!|x|\!) = umap\text{-}of\ \mathfrak{G}\ x\ \mathfrak{F}x\ (\eta(\!|NTMap|\!)(\!|x|\!))\ \mathfrak{F}'x(\!|ArrVal|\!)(\!|f|\!)$
⟨*proof*⟩

5.12.3 Main results

lemma *cf-adj-LR-iso-is-iso-functor*:
— See Corollary 1 in Chapter IV-1 in [8].
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 $\exists!\vartheta.$
$\vartheta : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D} \wedge$
$\eta_C\ \Psi = (\mathfrak{G} \circ_{CF\text{-}NTCF} \vartheta) \cdot_{NTCF} \eta_C\ \Phi$
and $cf\text{-}adj\text{-}LR\text{-}iso\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\eta_C\ \Psi =$
$(\mathfrak{G} \circ_{CF\text{-}NTCF} cf\text{-}adj\text{-}LR\text{-}iso\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{G}\ \mathfrak{F}\ \Phi\ \mathfrak{F}'\ \Psi) \cdot_{NTCF} \eta_C\ \Phi$
⟨*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 $op\text{-}\mathfrak{D} \equiv op\text{-}cat\ \mathfrak{D}$
and $op\text{-}\mathfrak{C} \equiv op\text{-}cat\ \mathfrak{C}$
and $op\text{-}\mathfrak{F} \equiv op\text{-}cf\ \mathfrak{F}$
and $op\text{-}\mathfrak{G} \equiv op\text{-}cf\ \mathfrak{G}$
and $op\text{-}\Phi \equiv op\text{-}cf\text{-}adj\ \Phi$
and $op\text{-}\mathfrak{G}' \equiv op\text{-}cf\ \mathfrak{G}'$
and $op\text{-}\Psi \equiv op\text{-}cf\text{-}adj\ \Psi$
shows
$op\text{-}ntcf\ (cf\text{-}adj\text{-}RL\text{-}iso\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}\ \Phi\ \mathfrak{G}'\ \Psi) =$
$cf\text{-}adj\text{-}LR\text{-}iso\ op\text{-}\mathfrak{D}\ op\text{-}\mathfrak{C}\ op\text{-}\mathfrak{F}\ op\text{-}\mathfrak{G}\ op\text{-}\Phi\ op\text{-}\mathfrak{G}'\ op\text{-}\Psi$
⟨*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 $op\text{-}\mathfrak{D} \equiv op\text{-}cat\ \mathfrak{D}$
and $op\text{-}\mathfrak{C} \equiv op\text{-}cat\ \mathfrak{C}$
and $op\text{-}\mathfrak{F} \equiv op\text{-}cf\ \mathfrak{F}$
and $op\text{-}\mathfrak{G} \equiv op\text{-}cf\ \mathfrak{G}$
and $op\text{-}\Phi \equiv op\text{-}cf\text{-}adj\ \Phi$
and $op\text{-}\mathfrak{F}' \equiv op\text{-}cf\ \mathfrak{F}'$

and *op-Ψ* ≡ *op-cf-adj* Ψ
shows
op-ntcf (*cf-adj-LR-iso* 𝔈 𝔇 𝔊 𝔉 Φ 𝔉′ Ψ) =
cf-adj-RL-iso op-𝔇 op-𝔈 op-𝔊 op-𝔉 op-Φ op-𝔉′ op-Ψ
⟨*proof*⟩

lemma *cf-adj-RL-iso-app-unique*:
fixes 𝔈 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ
assumes Φ : 𝔉 ⇌$_{CF}$ 𝔊 : 𝔈 ⇌⇌$_{C\alpha}$ 𝔇
and Ψ : 𝔉 ⇌$_{CF}$ 𝔊′ : 𝔈 ⇌⇌$_{C\alpha}$ 𝔇
and *x* ∈$_{\circ}$ 𝔇⦇*Obj*⦈
defines 𝔊*x* ≡ 𝔊⦇*ObjMap*⦈⦇*x*⦈
and 𝔊′*x* ≡ 𝔊′⦇*ObjMap*⦈⦇*x*⦈
and *ε* ≡ *ε*$_C$ Φ
and *ε*′ ≡ *ε*$_C$ Ψ
and *f* ≡ *cf-adj-RL-iso* 𝔈 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ⦇*NTMap*⦈⦇*x*⦈
shows
∃!*f*′.
f′ : 𝔊′*x* ↦$_{\mathfrak{E}}$ 𝔊*x* ∧
ε′⦇*NTMap*⦈⦇*x*⦈ = *umap-fo* 𝔉 *x* 𝔊*x* (*ε*⦇*NTMap*⦈⦇*x*⦈) 𝔊′*x*⦇*ArrVal*⦈⦇*f*′⦈
f : 𝔊′*x* ↦$_{iso\mathfrak{E}}$ 𝔊*x*
ε′⦇*NTMap*⦈⦇*x*⦈ = *umap-fo* 𝔉 *x* 𝔊*x* (*ε*⦇*NTMap*⦈⦇*x*⦈) 𝔊′*x*⦇*ArrVal*⦈⦇*f*⦈
⟨*proof*⟩

lemma *cf-adj-RL-iso-is-iso-functor*:
— See Corollary 1 in Chapter IV-1 in [8].
assumes Φ : 𝔉 ⇌$_{CF}$ 𝔊 : 𝔈 ⇌⇌$_{C\alpha}$ 𝔇 **and** Ψ : 𝔉 ⇌$_{CF}$ 𝔊′ : 𝔈 ⇌⇌$_{C\alpha}$ 𝔇
shows ∃!*ϑ*.
ϑ : 𝔊′ ↦$_{CF}$ 𝔊 : 𝔇 ↦↦$_{C\alpha}$ 𝔈 ∧
ε$_C$ Ψ = *ε*$_C$ Φ ⋆$_{NTCF}$ (𝔉 ∘$_{CF-NTCF}$ *ϑ*)
and *cf-adj-RL-iso* 𝔈 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ : 𝔊′ ↦$_{CF.iso}$ 𝔊 : 𝔇 ↦↦$_{C\alpha}$ 𝔈
and *ε*$_C$ Ψ =
ε$_C$ Φ ⋆$_{NTCF}$ (𝔉 ∘$_{CF-NTCF}$ *cf-adj-RL-iso* 𝔈 𝔇 𝔉 𝔊 Φ 𝔊′ Ψ)
⟨*proof*⟩

5.13 Further properties of the adjoint functors

lemma (**in** *is-cf-adjunction*) *cf-adj-exp-cf-cat*:
— See Proposition 4.4.6 in [13].
assumes 𝒵 *β* **and** *α* ∈$_{\circ}$ *β* **and** *category α* 𝔍
shows
cf-adjunction-of-unit
β
(*exp-cf-cat α* 𝔉 𝔍)
(*exp-cf-cat α* 𝔊 𝔍)
(*exp-ntcf-cat α* (*η*$_C$ Φ) 𝔍) :
exp-cf-cat α 𝔉 𝔍 ⇌$_{CF}$ *exp-cf-cat α* 𝔊 𝔍 :
cat-FUNCT α 𝔍 𝔈 ⇌⇌$_{C\beta}$ *cat-FUNCT α* 𝔍 𝔇
⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adj-exp-cf-cat-exp-cf-cat*:
— See Proposition 4.4.6 in [13].
assumes 𝒵 *β* **and** *α* ∈$_{\circ}$ *β* **and** *category α* 𝔄
shows
cf-adjunction-of-unit
β
(*exp-cat-cf α* 𝔄 𝔊)
(*exp-cat-cf α* 𝔄 𝔉)

$(exp\text{-}cat\text{-}ntcf\ \alpha\ \mathfrak{A}\ (\eta_C\ \Phi)) :$
$exp\text{-}cat\text{-}cf\ \alpha\ \mathfrak{A}\ \mathfrak{G} \rightleftharpoons_{C\,F}\ exp\text{-}cat\text{-}cf\ \alpha\ \mathfrak{A}\ \mathfrak{F} :$
$cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ \mathfrak{A} \rightleftharpoons\rightleftharpoons_{C\,\beta}\ cat\text{-}FUNCT\ \alpha\ \mathfrak{D}\ \mathfrak{A}$
⟨*proof*⟩

6 Simple Kan extensions

6.1 Background

named-theorems *ua-field-simps*

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

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

6.2 Kan extension

6.2.1 Definition and elementary properties

See Chapter X-3 in [8].

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* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *bool*
  (‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe}$ 1 - :/ - $\mapsto_{C}$ - $\mapsto_{C}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations ε : $\mathfrak{G}$ $\circ_{CF}$ $\mathfrak{K}$ $\mapsto_{CF.rKe\alpha}$ $\mathfrak{T}$: $\mathfrak{B}$ $\mapsto_{C}$ $\mathfrak{C}$ $\mapsto_{C}$ $\mathfrak{A}$ ⇌
  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* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *bool*
  (‹(- :/ - $\mapsto_{CF.lKe}$ 1 - $\circ_{CF}$ - :/ - $\mapsto_{C}$ - $\mapsto_{C}$ -)› [*51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*)
translations η : $\mathfrak{T}$ $\mapsto_{CF.lKe\alpha}$ $\mathfrak{F}$ $\circ_{CF}$ $\mathfrak{K}$: $\mathfrak{B}$ $\mapsto_{C}$ $\mathfrak{C}$ $\mapsto_{C}$ $\mathfrak{A}$ ⇌
  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 α' = α
    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 $\mathfrak{K}' = \mathit{op\text{-}cf}\ \mathfrak{K}$
and $\mathfrak{B}' = \mathit{op\text{-}cat}\ \mathfrak{B}$
and $\mathfrak{A}' = \mathit{op\text{-}cat}\ \mathfrak{A}$
and $\mathfrak{C}' = \mathit{op\text{-}cat}\ \mathfrak{C}$
shows $\mathit{op\text{-}ntcf}\ \eta : \mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_{C} \mathfrak{C}' \mapsto_{C} \mathfrak{A}'$
⟨*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 $\mathit{exp\text{-}cat\text{-}cf}\ \alpha\ \mathfrak{A}\ \mathfrak{K} : \mathit{cat\text{-}FUNCT}\ \alpha\ \mathfrak{C}\ \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} \mathit{cat\text{-}FUNCT}\ \alpha\ \mathfrak{B}\ \mathfrak{A}$
⟨*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 $\mathit{exp\text{-}cat\text{-}cf}\ \alpha\ \mathfrak{A}\ \mathfrak{K} : \mathit{cat\text{-}FUNCT}\ \alpha\ \mathfrak{C}\ \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} \mathit{cat\text{-}FUNCT}\ \alpha\ \mathfrak{B}\ \mathfrak{A}$
⟨*proof*⟩

6.2.2 Universal property

See Chapter X-3 in [8] and [2][3].

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 $\bigwedge \mathfrak{G}'\ \varepsilon'.$
$[\![\ \mathfrak{G}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A};\ \varepsilon' : \mathfrak{G}' \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ ]\!] \Longrightarrow$
$\exists!\sigma.\ \sigma : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \cdot_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$
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 $\bigwedge \mathfrak{F}'\ \eta'.$
$[\![\ \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A};\ \eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ ]\!] \Longrightarrow$
$\exists!\sigma.\ \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \cdot_{NTCF} \eta$
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 $\exists!\sigma.\ \sigma : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \cdot_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$
⟨*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 $\exists!\sigma.\ \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \cdot_{NTCF} \eta$
⟨*proof*⟩

6.2.3 Further properties

lemma (in *is-cat-rKe*) *cat-rKe-ntcf-ua-fo-is-iso-ntcf-if-ge-Limit*:

[3] https://en.wikipedia.org/wiki/Kan_extension

assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows
ntcf-ua-fo β (*exp-cat-cf* $\alpha\ \mathfrak{A}\ \mathfrak{K}$) (*cf-map* $\mathfrak{T}$) (*cf-map* $\mathfrak{G}$) (*ntcf-arrow* ε) :
$Hom_{O.C\beta}$*cat-FUNCT* $\alpha\ \mathfrak{C}\ \mathfrak{A}(-,$*cf-map* $\mathfrak{G}) \mapsto_{CF.iso}$
$Hom_{O.C\beta}$*cat-FUNCT* $\alpha\ \mathfrak{B}\ \mathfrak{A}(-,$*cf-map* $\mathfrak{T}) \circ_{CF}$ *op-cf* (*exp-cat-cf* $\alpha\ \mathfrak{A}\ \mathfrak{K}$) :
op-cat (*cat-FUNCT* $\alpha\ \mathfrak{C}\ \mathfrak{A}) \mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

lemma (in *is-cat-lKe*) *cat-lKe-ntcf-ua-fo-is-iso-ntcf-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
defines $\mathfrak{AK} \equiv$ *exp-cat-cf* α (*op-cat* $\mathfrak{A}$) (*op-cf* $\mathfrak{K}$)
and $\mathfrak{AC} \equiv$ *cat-FUNCT* α (*op-cat* $\mathfrak{C}$) (*op-cat* $\mathfrak{A}$)
and $\mathfrak{AB} \equiv$ *cat-FUNCT* α (*op-cat* $\mathfrak{B}$) (*op-cat* $\mathfrak{A}$)
shows
ntcf-ua-fo $\beta\ \mathfrak{AK}$ (*cf-map* $\mathfrak{T}$) (*cf-map* $\mathfrak{F}$) (*ntcf-arrow* (*op-ntcf* η)) :
$Hom_{O.C\beta}\mathfrak{AC}(-,$*cf-map* $\mathfrak{F}) \mapsto_{CF.iso} Hom_{O.C\beta}\mathfrak{AB}(-,$*cf-map* $\mathfrak{T}) \circ_{CF}$ *op-cf* $\mathfrak{AK}$:
op-cat $\mathfrak{AC} \mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

6.3 Opposite universal arrow for Kan extensions

6.3.1 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* :: $(V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
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*⟩

6.3.2 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 $u : \mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\bigsqcap_O\ c >_{CF.colim} r : \mathfrak{K}\ {}_{CF}{\downarrow}\ c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *op-ntcf* $u \circ_{NTCF-CF}$ *inv-cf* (*op-cf-obj-comma* $\mathfrak{K}\ c$) :
$r <_{CF.lim}$ *op-cf* $\mathfrak{T} \circ_{CF} c\ {}_O\bigsqcap_{CF}$ (*op-cf* $\mathfrak{K}$) : $c \downarrow_{CF}$ (*op-cf* $\mathfrak{K}$) $\mapsto\mapsto_{C\alpha}$ *op-cat* $\mathfrak{A}$
⟨*proof*⟩

context
fixes *lim-Obj* :: $V \Rightarrow V$ **and** $c :: V$
begin

lemmas *op-ua-UArr-is-cat-limit'* = *op-ua-UArr-is-cat-limit*
[
unfolded op-ua-components(*2*)[*symmetric*],

where $u{=}\langle \mathit{lim\text{-}Obj}\ c(\!|\mathit{UArr}|\!)\rangle$ **and** $r{=}\langle \mathit{lim\text{-}Obj}\ c(\!|\mathit{UObj}|\!)\rangle$ **and** $c{=}c$,
folded op-ua-components(*2*)[**where** *lim-Obj*=*lim-Obj* **and** $c{=}c$]
]

end

6.4 The Kan extension

The following subsection is based on the statement and proof of Theorem 1 in Chapter X-3 in [8].

6.4.1 Definition and elementary properties

definition *the-cf-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* =
[
$(\lambda c{\in_\circ}\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Obj}|\!).\ \mathit{lim\text{-}Obj}\ c(\!|\mathit{UObj}|\!))$,
(
$\lambda g{\in_\circ}\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Arr}|\!).\ \mathit{THE}\ f.$
f :
lim-Obj $(\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Dom}|\!)(\!|g|\!))(\!|\mathit{UObj}|\!) \mapsto_{\mathfrak{T}(\!|\mathit{HomCod}|\!)}$
lim-Obj $(\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Cod}|\!)(\!|g|\!))(\!|\mathit{UObj}|\!)\ \wedge$
lim-Obj $(\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Dom}|\!)(\!|g|\!))(\!|\mathit{UArr}|\!) \circ_{NTCF-CF} g\ {}_A{\downarrow_{CF}}\ \mathfrak{K} =$
lim-Obj $(\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Cod}|\!)(\!|g|\!))(\!|\mathit{UArr}|\!) \cdot_{NTCF}$
ntcf-const $((\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Cod}|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{K})\ (\mathfrak{T}(\!|\mathit{HomCod}|\!))\ f$
),
$\mathfrak{K}(\!|\mathit{HomCod}|\!)$,
$\mathfrak{T}(\!|\mathit{HomCod}|\!)$
]$_\circ$

definition *the-ntcf-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* =
[
(
$\lambda c{\in_\circ}\mathfrak{T}(\!|\mathit{HomDom}|\!)(\!|\mathit{Obj}|\!).$
lim-Obj $(\mathfrak{K}(\!|\mathit{ObjMap}|\!)(\!|c|\!))(\!|\mathit{UArr}|\!)(\!|\mathit{NTMap}|\!)(\!|0,\ c,\ \mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{CId}|\!)(\mathfrak{K}(\!|\mathit{ObjMap}|\!)(\!|c|\!))|\!)_\bullet$
),
the-cf-rKe $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* $\circ_{CF}\ \mathfrak{K}$,
$\mathfrak{T}$,
$\mathfrak{T}(\!|\mathit{HomDom}|\!)$,
$\mathfrak{T}(\!|\mathit{HomCod}|\!)$
]$_\circ$

definition *the-cf-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *the-cf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* =
op-cf (*the-cf-rKe* α (*op-cf* $\mathfrak{T}$) (*op-cf* $\mathfrak{K}$) (*op-ua lim-Obj* $\mathfrak{K}$))

definition *the-ntcf-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *the-ntcf-lKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* =
op-ntcf (*the-ntcf-rKe* α (*op-cf* $\mathfrak{T}$) (*op-cf* $\mathfrak{K}$) (*op-ua lim-Obj* $\mathfrak{K}$))

Components.

lemma *the-cf-rKe-components*:
shows *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|\mathit{ObjMap}|\!)$ =
$(\lambda c{\in_\circ}\mathfrak{K}(\!|\mathit{HomCod}|\!)(\!|\mathit{Obj}|\!).\ \mathit{lim\text{-}Obj}\ c(\!|\mathit{UObj}|\!))$
and *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|\mathit{ArrMap}|\!)$ =
(

$\lambda g \in_\circ \mathfrak{K}(\!|HomCod|\!)(\!|Arr|\!).$ *THE* $f.$
$f :$
lim-Obj $(\mathfrak{K}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|UObj|\!) \mapsto_{\mathfrak{T}(\!|HomCod|\!)}$
lim-Obj $(\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))(\!|UObj|\!) \wedge$
lim-Obj $(\mathfrak{K}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|UArr|\!) \circ_{NTCF-CF} g_{A\downarrow_{CF}} \mathfrak{K} =$
lim-Obj $(\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))(\!|UArr|\!) \cdot_{NTCF}$
ntcf-const $((\mathfrak{K}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{K})\ (\mathfrak{T}(\!|HomCod|\!))\ f$
)
and *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|HomDom|\!) = \mathfrak{K}(\!|HomCod|\!)$
and *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|HomCod|\!) = \mathfrak{T}(\!|HomCod|\!)$
⟨*proof*⟩

lemma *the-ntcf-rKe-components*:
shows *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|NTMap|\!) =$
(
$\lambda c \in_\circ \mathfrak{T}(\!|HomDom|\!)(\!|Obj|\!).$
lim-Obj $(\mathfrak{K}(\!|ObjMap|\!)(\!|c|\!))(\!|UArr|\!)(\!|NTMap|\!)(\!|0,\ c,\ \mathfrak{K}(\!|HomCod|\!)(\!|CId|\!)(\mathfrak{K}(\!|ObjMap|\!)(\!|c|\!))|\!)_\bullet$
)
and *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|NTDom|\!) =$ *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj* $\circ_{CF}\ \mathfrak{K}$
and *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|NTCod|\!) = \mathfrak{T}$
and *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|NTDGDom|\!) = \mathfrak{T}(\!|HomDom|\!)$
and *the-ntcf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|NTDGCod|\!) = \mathfrak{T}(\!|HomCod|\!)$
⟨*proof*⟩

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}\ \mathfrak{T}$
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* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{A}\ \mathfrak{T}$ ⟨*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

6.4.2 Functor: object map

mk-VLambda *the-cf-rKe-components*(*1*)
|*vsv the-cf-rKe-ObjMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{K}\ \mathfrak{T}$
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 $\bigwedge c.\ c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow lim\text{-}Obj\ c(\lfloor UObj \rfloor) \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $\mathcal{R}_\circ$ (*the-cf-rKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj*($\lfloor ObjMap \rfloor$)) $\subseteq_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

end

6.4.3 Functor: arrow map

mk-VLambda *the-cf-rKe-components*(*2*)
|*vsv the-cf-rKe-ArrMap-vsv*[*cat-Kan-cs-intros*]|

context
fixes α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$
assumes $\mathfrak{K}$: $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
begin

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

lemma g': $g \in_\circ \mathfrak{C}(\lfloor Arr \rfloor)$ ⟨*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 $u : r <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_O\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $u' : r' <_{CF.lim} \mathfrak{T} \circ_{CF} c'\ {}_O\bigsqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\exists!f$.

$f : r \mapsto_{\mathfrak{A}} r' \wedge$
$u \circ_{NTCF-CF} g_{A\downarrow_{CF}} \mathfrak{K} = u' \cdot_{NTCF} \textit{ntcf-const}\ (c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ f$
⟨*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 *lim-Obj* $c(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ c(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and *lim-Obj* $c'(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ c'(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c'\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|\mathit{ArrMap}|\!)(\!|g|\!)$:
$\textit{lim-Obj}\ c(\!|\mathit{UObj}|\!) \mapsto_{\mathfrak{A}} \textit{lim-Obj}\ c'(\!|\mathit{UObj}|\!)$
and
$\textit{lim-Obj}\ c(\!|\mathit{UArr}|\!) \circ_{NTCF-CF} g_{A\downarrow_{CF}} \mathfrak{K} =$
$\textit{lim-Obj}\ c'(\!|\mathit{UArr}|\!) \cdot_{NTCF}$
$\textit{ntcf-const}\ (c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ (\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(\!|\mathit{ArrMap}|\!)(\!|g|\!))$
and
$[\![$
$f : \textit{lim-Obj}\ c(\!|\mathit{UObj}|\!) \mapsto_{\mathfrak{A}} \textit{lim-Obj}\ c'(\!|\mathit{UObj}|\!);$
$\textit{lim-Obj}\ c(\!|\mathit{UArr}|\!) \circ_{NTCF-CF} g_{A\downarrow_{CF}} \mathfrak{K} =$
$\textit{lim-Obj}\ c'(\!|\mathit{UArr}|\!) \cdot_{NTCF} \textit{ntcf-const}\ (c' \downarrow_{CF} \mathfrak{K})\ \mathfrak{A}\ f$
$]\!] \Longrightarrow f = \textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(\!|\mathit{ArrMap}|\!)(\!|g|\!)$
⟨*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 *lim-Obj* $c(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ c(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and *lim-Obj* $c'(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ c'(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c'\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a = \textit{lim-Obj}\ c(\!|\mathit{UObj}|\!)$
and $b = \textit{lim-Obj}\ c'(\!|\mathit{UObj}|\!)$
shows *the-cf-rKe* $\alpha\ \mathfrak{T}\ \mathfrak{K}$ *lim-Obj*$(\!|\mathit{ArrMap}|\!)(\!|g|\!) : a \mapsto_{\mathfrak{A}} b$
⟨*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 *lim-Obj* $a(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ a(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} a\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : a \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and *lim-Obj* $b(\!|\mathit{UArr}|\!)$:
$\textit{lim-Obj}\ b(\!|\mathit{UObj}|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} b\ {}_{O}\!\sqcap_{CF} \mathfrak{K} : b \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $f : a \mapsto_{\mathfrak{C}} b$
and $[a', b', f']_{\circ} \in_{\circ} b \downarrow_{CF} \mathfrak{K}(\!|\mathit{Obj}|\!)$
shows
$\textit{lim-Obj}\ a(\!|\mathit{UArr}|\!)(\!|\mathit{NTMap}|\!)(\!|a', b', f' \circ_{A\mathfrak{C}} f|\!)_{\bullet} =$
$\textit{lim-Obj}\ b(\!|\mathit{UArr}|\!)(\!|\mathit{NTMap}|\!)(\!|a', b', f'|\!)_{\bullet} \circ_{A\mathfrak{A}}$
$\textit{the-cf-rKe}\ \alpha\ \mathfrak{T}\ \mathfrak{K}\ \textit{lim-Obj}(\!|\mathit{ArrMap}|\!)(\!|f|\!)$
⟨*proof*⟩

6.4.4 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

6.4.5 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 $\bigwedge c.\ c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow$ *lim-Obj* $c(\lfloor UArr \rfloor)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\bigsqcap_O c >_{CF.colim}$ *lim-Obj* $c(\lfloor UObj \rfloor) : \mathfrak{K}\ {}_{CF}\!\downarrow c \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* :
$\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *the-cf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ *lim-Obj* $\circ_{CF}$ $\mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

6.5 Preservation of Kan extensions

The following definitions are similar to the definitions that can be found in [13] or [7].

locale *is-cat-rKe-preserves* =
is-cat-rKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ ε + *is-functor* α $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{H}$
for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $\mathfrak{H}$ ε +
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* ::
$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(
‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe1}$ - :/ - $\mapsto_C$ - $\mapsto_C$ - : - $\mapsto\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_C \mathfrak{D}) \rightleftharpoons$
CONST is-cat-rKe-preserves α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $\mathfrak{H}$ ε

locale *is-cat-lKe-preserves* =
is-cat-lKe α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ η + *is-functor* α $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{H}$
for α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $\mathfrak{H}$ η +
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* ::
$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(
‹(- :/ - $\mapsto_{CF.lKe1}$ - $\circ_{CF}$ - :/ - $\mapsto_C$ - $\mapsto_C$ - : - $\mapsto\mapsto_C$ -)›
[*51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*, *51*] *51*
)
translations $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_C \mathfrak{D}) \rightleftharpoons$
CONST is-cat-lKe-preserves α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{D}$ $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $\mathfrak{H}$ η

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* $\mathfrak{C} \mapsto_{C} (\textit{op-cf}\ \mathfrak{H} : \textit{op-cat}\ \mathfrak{A} \mapsto\mapsto_{C} \textit{op-cat}\ \mathfrak{D})$
⟨*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* $\mathfrak{C} \mapsto_{C} (\textit{op-cf}\ \mathfrak{H} : \textit{op-cat}\ \mathfrak{A} \mapsto\mapsto_{C} \textit{op-cat}\ \mathfrak{D})$
⟨*proof*⟩

lemma (**in** *is-cat-lKe-preserves*) *is-cat-rKe-preserves-op'*[*cat-op-intros*]:

assumes $\mathfrak{T}' = op\text{-}cf\ \mathfrak{T}$
and $\mathfrak{F}' = op\text{-}cf\ \mathfrak{F}$
and $\mathfrak{K}' = op\text{-}cf\ \mathfrak{K}$
and $\mathfrak{H}' = op\text{-}cf\ \mathfrak{H}$
and $\mathfrak{B}' = op\text{-}cat\ \mathfrak{B}$
and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{A}$
and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
and $\mathfrak{D}' = op\text{-}cat\ \mathfrak{D}$
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*⟩

6.6 All concepts are Kan extensions

Background information for this subsection is provided in Chapter X-7 in [8] and section 6.5 in [13]. 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).

6.6.1 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 [8] or Proposition 6.5.1 in [13].
assumes $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
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 $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
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 [8] or Proposition 6.5.1 in [13].
assumes $\varepsilon : \mathfrak{G}(\!|ObjMap|\!)(|\mathfrak{a}|) <_{CF.lim} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}1\ \mathfrak{a}\ \mathfrak{f}$
and $\mathfrak{G} : cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
⟨*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 $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}1\ \mathfrak{a}\ \mathfrak{f}$
and $\mathfrak{F} : cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C cat\text{-}1\ \mathfrak{a}\ \mathfrak{f} \mapsto_C \mathfrak{A}$
⟨*proof*⟩

6.6.2 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 [8] or Proposition 6.5.2 in [13]. The proof follows (approximately) the proof in [13].
shows $\varepsilon_C\ \Phi : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF.rKe\alpha} cf\text{-}id\ \mathfrak{D} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

⟨*proof*⟩

lemma (**in** *is-cf-adjunction*) *cf-adjunction-unit-is-lKe*:
shows $\eta_C\ \Phi : cf\text{-}id\ \mathfrak{C} \mapsto_{CF.lKe\alpha} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_C \mathfrak{D} \mapsto_C \mathfrak{C}$
⟨*proof*⟩

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 [8] or Proposition 6.5.2 in [13].
assumes $\eta : cf\text{-}id\ \mathfrak{D} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{G} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_C \mathfrak{C})$
shows $cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{G}\ \mathfrak{F}\ \eta : \mathfrak{G} \rightleftharpoons_{CF} \mathfrak{F} : \mathfrak{D} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-adjunction-if-rKe-preserves*:
assumes $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF.rKe\alpha} cf\text{-}id\ \mathfrak{D} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{G} : \mathfrak{D} \mapsto\mapsto_C \mathfrak{C})$
shows $cf\text{-}adjunction\text{-}of\text{-}counit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons\rightleftharpoons_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

7 Pointwise Kan extensions

7.1 Pointwise Kan extensions

The following subsection is based on elements of the content of section 6.3 in [13] and Chapter X-5 in [8].

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 :
      $\mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T}$:
      $\mathfrak{B} \mapsto_{C} \mathfrak{C} \mapsto_{C} (Hom_{O.C\alpha}\mathfrak{A}(a,-) : \mathfrak{A} \mapsto\mapsto_{C} cat\text{-}Set\ \alpha)$

syntax *-is-cat-pw-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (
    ‹(- :/ - $\circ_{CF}$ - $\mapsto_{CF.rKe.pw1}$ - :/ - $\mapsto_{C}$ - $\mapsto_{C}$ -)›
    [51, 51, 51, 51, 51, 51, 51] 51
)
translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe.pw\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_{C} \mathfrak{C} \mapsto_{C} \mathfrak{A} \rightleftharpoons$
  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*: $a \in_{\circ} op\text{-}cat\ \mathfrak{A}(\!|Obj|\!) \Longrightarrow$
    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* $\mathfrak{C} \mapsto_{C} (Hom_{O.C\alpha}\mathfrak{A}(-,a) : op\text{-}cat\ \mathfrak{A} \mapsto\mapsto_{C} cat\text{-}Set\ \alpha)$

syntax *-is-cat-pw-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (
    ‹(- :/ - $\mapsto_{CF.lKe.pw1}$ - $\circ_{CF}$ - :/ - $\mapsto_{C}$ - $\mapsto_{C}$ -)›
    [51, 51, 51, 51, 51, 51, 51] 51
)
translations $\eta : \mathfrak{T} \mapsto_{CF.lKe.pw\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_{C} \mathfrak{C} \mapsto_{C} \mathfrak{A} \rightleftharpoons$
  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 $\mathfrak{E}' = cat\text{-}Set\ \alpha$
  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 $a \in_{\circ} op\text{-}cat\ \mathfrak{A}(\!|Obj|\!)$
    and $\mathfrak{F}' = op\text{-}cf\ \mathfrak{F}$
    and $\mathfrak{K}' = op\text{-}cf\ \mathfrak{K}$
    and $\mathfrak{T}' = op\text{-}cf\ \mathfrak{T}$
    and $\mathfrak{B}' = op\text{-}cat\ \mathfrak{B}$
    and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
    and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{A}$
    and $\mathfrak{H}' = Hom_{O.C\alpha}\mathfrak{A}(-,a)$
    and $\mathfrak{E}' = cat\text{-}Set\ \alpha$
  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′*

7.2 Cone functor

7.2.1 Definition and elementary properties

definition *cf-Cone* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-Cone* α β $\mathfrak{F}$ =
$Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}$(|*HomDom*|)) ($\mathfrak{F}$(|*HomCod*|))(−,*cf-map* $\mathfrak{F}$) $\circ_{CF}$
op-cf (Δ_{CF} α ($\mathfrak{F}$(|*HomDom*|)) ($\mathfrak{F}$(|*HomCod*|)))

An alternative form of the definition.

context *is-functor*
begin

lemma *cf-Cone-def′*:
cf-Cone α β $\mathfrak{F}$ = $Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(−,*cf-map* $\mathfrak{F}$) $\circ_{CF}$ *op-cf* (Δ_{CF} α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

end

7.2.2 Object map

lemma (**in** *is-functor*) *cf-Cone-ObjMap-vsv*[*cat-Kan-cs-intros*]:
assumes $\mathcal{Z}$ β **and** α $\in_\circ$ β
shows *vsv* (*cf-Cone* α β $\mathfrak{F}$(|*ObjMap*|))
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = *is-functor.cf-Cone-ObjMap-vsv*

lemma (**in** *is-functor*) *cf-Cone-ObjMap-vdomain*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β **and** α $\in_\circ$ β **and** b $\in_\circ$ $\mathfrak{B}$(|*Obj*|)
shows $\mathcal{D}_\circ$ (*cf-Cone* α β $\mathfrak{F}$(|*ObjMap*|)) = $\mathfrak{B}$(|*Obj*|)
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.cf-Cone-ObjMap-vdomain*

lemma (**in** *is-functor*) *cf-Cone-ObjMap-app*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β **and** α $\in_\circ$ β **and** b $\in_\circ$ $\mathfrak{B}$(|*Obj*|)
shows *cf-Cone* α β $\mathfrak{F}$(|*ObjMap*|)(|b|) =

Hom (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* (*cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b)) (*cf-map* $\mathfrak{F}$)
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.cf-Cone-ObjMap-app*

7.2.3 Arrow map

lemma (**in** *is-functor*) *cf-Cone-ArrMap-vsv*[*cat-Kan-cs-intros*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *vsv* (*cf-Cone* α β $\mathfrak{F}$(|*ArrMap*|))
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = *is-functor.cf-Cone-ArrMap-vsv*

lemma (**in** *is-functor*) *cf-Cone-ArrMap-vdomain*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$ **and** $b \in_\circ \mathfrak{B}$(|*Obj*|)
shows $\mathcal{D}_\circ$ (*cf-Cone* α β $\mathfrak{F}$(|*ArrMap*|)) = $\mathfrak{B}$(|*Arr*|)
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.cf-Cone-ArrMap-vdomain*

lemma (**in** *is-functor*) *cf-Cone-ArrMap-app*[*cat-Kan-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $f : a \mapsto_{\mathfrak{B}} b$
shows *cf-Cone* α β $\mathfrak{F}$(|*ArrMap*|)(|f|) = *cf-hom*
(*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$)
[*ntcf-arrow* (*ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ f), *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*CId*|)(|*cf-map* $\mathfrak{F}$|)]$_\circ$
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *is-functor.cf-Cone-ArrMap-app*

7.2.4 The cone functor is a functor

lemma (**in** *is-functor*) *tm-cf-cf-Cone-is-functor-if-ge-Limit*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *cf-Cone* α β $\mathfrak{F}$: *op-cat* $\mathfrak{B}$ ↦↦$_{C\beta}$ *cat-Set* β
⟨*proof*⟩

7.3 Lemma X.5: *L-10-5-N*

This subsection and several further subsections (7.3-7.9) 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 [8].

definition *L-10-5-N* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
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*|)($a$,−) ∘$_{CF}$ $\mathfrak{T}$), 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*|)($f$,−) ∘$_{NTCF-CF}$ $\mathfrak{T}$), $\mathfrak{K}$(|*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

7.3.1 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

7.3.2 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

7.3.3 *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*⟩

7.4 Lemma X.5: *L-10-5-υ-arrow*

7.4.1 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

7.4.2 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 $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{A}$ $\mathfrak{K}$ $\mathfrak{T}$
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

mk-VLambda *L-10-5-υ-arrow-components′*(*1*)[*OF* $\mathfrak{K}$ $\mathfrak{T}$]
|*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 $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $f : c \mapsto_{\mathfrak{C}} \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
shows *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ a $b(\lfloor ArrVal \rfloor)(\lfloor f \rfloor) = \tau(\lfloor NTMap \rfloor)(\lfloor 0,\ b,\ f \rfloor)_{\bullet}$
⟨*proof*⟩

7.4.3 *L-10-5-υ-arrow* is an arrow

lemma *L-10-5-υ-arrow-ArrVal-is-arr*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_{O}\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $f : c \mapsto_{\mathfrak{C}} \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $b \in_{\circ} \mathfrak{B}(\lfloor Obj \rfloor)$
shows *L-10-5-υ-arrow* $\mathfrak{T}$ $\mathfrak{K}$ c τ' a $b(\lfloor ArrVal \rfloor)(\lfloor f \rfloor) : a \mapsto_{\mathfrak{A}} \mathfrak{T}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
⟨*proof*⟩

lemma *L-10-5-υ-arrow-ArrVal-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 $\tau' = ntcf\text{-}arrow\ \tau$
and $a' = a$
and $b' = \mathfrak{T}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $\mathfrak{A}' = \mathfrak{A}$

and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
and $f : c \mapsto_{\mathfrak{C}} \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows *L-10-5-υ-arrow* $\mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b(\lfloor ArrVal \rfloor)(\lfloor f \rfloor) : a' \mapsto_{\mathfrak{A}} b'$
⟨*proof*⟩

7.4.4 Further elementary properties

lemma *L-10-5-υ-arrow-is-arr*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows *L-10-5-υ-arrow* $\mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b$:
$Hom\ \mathfrak{C}\ c\ (\mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)) \mapsto_{cat\text{-}Set\ \alpha} Hom\ \mathfrak{A}\ a\ (\mathfrak{T}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor))$
⟨*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 $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
and $A = Hom\ \mathfrak{C}\ c\ (\mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor))$
and $B = Hom\ \mathfrak{A}\ a\ (\mathfrak{T}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor))$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows *L-10-5-υ-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 $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and $\tau' = ntcf\text{-}arrow\ \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
and $f : a' \mapsto_{\mathfrak{B}} b'$
shows
L-10-5-υ-arrow $\mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b' \circ_{A\, cat\text{-}Set\ \alpha}$
cf-hom $\mathfrak{C}\ [\mathfrak{C}(\lfloor CId \rfloor)(\lfloor c \rfloor), \mathfrak{K}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)]_\circ =$
cf-hom $\mathfrak{A}\ [\mathfrak{A}(\lfloor CId \rfloor)(\lfloor a \rfloor), \mathfrak{T}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)]_\circ \circ_{A\, cat\text{-}Set\ \alpha}$
L-10-5-υ-arrow $\mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ a'$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

7.5 Lemma X.5: *L-10-5-τ*

7.5.1 Definition and elementary properties

definition *L-10-5-τ* **where** *L-10-5-τ* $\mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a =$
[
$(\lambda bf \in_\circ c \downarrow_{CF} \mathfrak{K}(\lfloor Obj \rfloor).\ \upsilon(\lfloor NTMap \rfloor)(\lfloor bf(\lfloor 1_{\mathbb{N}} \rfloor) \rfloor)(\lfloor ArrVal \rfloor)(\lfloor bf(\lfloor 2_{\mathbb{N}} \rfloor) \rfloor))$,
cf-const $(c \downarrow_{CF} \mathfrak{K})\ (\mathfrak{T}(\lfloor HomCod \rfloor))\ a$,
$\mathfrak{T} \circ_{CF} c \ {}_O\!\bigsqcap_{CF} \mathfrak{K}$,
$c \downarrow_{CF} \mathfrak{K}$,
$(\mathfrak{T}(\lfloor HomCod \rfloor))$

]$_\circ$

Components.

lemma *L-10-5-τ-components*:
shows *L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTMap|\!) =$
$(\lambda bf \in_\circ c \downarrow_{CF} \mathfrak{K}(\!|Obj|\!).\ \upsilon(\!|NTMap|\!)(\!|bf(\!|1_{\mathbb{N}}|\!)|\!)(\!|ArrVal|\!)(\!|bf(\!|2_{\mathbb{N}}|\!)|\!))$
and *L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTDom|\!) =$ *cf-const* $(c \downarrow_{CF} \mathfrak{K})$ $(\mathfrak{T}(\!|HomCod|\!))$ a
and *L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTCod|\!) = \mathfrak{T} \circ_{CF} c\ {}_O\!\sqcap_{CF} \mathfrak{K}$
and *L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTDGDom|\!) = c \downarrow_{CF} \mathfrak{K}$
and *L-10-5-τ* $\mathfrak{T}$ $\mathfrak{K}$ c υ $a(\!|NTDGCod|\!) = (\mathfrak{T}(\!|HomCod|\!))$
⟨*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-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

7.5.2 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*⟩

7.5.3 *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*: υ' = *ntcf-arrow* υ
and υ: υ :
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
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 υ' = *ntcf-arrow* υ

and $\mathfrak{F}' = \mathfrak{T} \circ_{CF} c\ {}_{O}\!\bigsqcap_{CF} \mathfrak{K}$
and $c\mathfrak{K} = c \downarrow_{CF} \mathfrak{K}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\alpha' = \alpha$
and υ :
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$:
$\mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *L-10-5-τ* $\mathfrak{T}\ \mathfrak{K}\ c\ \upsilon'\ a : a <_{CF.cone} \mathfrak{F}' : c\mathfrak{K} \mapsto\mapsto_{C_{\alpha'}} \mathfrak{A}'$
⟨*proof*⟩

7.6 Lemma X.5: *L-10-5-υ*

7.6.1 Definition and elementary properties

definition *L-10-5-υ* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a$ =
[
$(\lambda b \in_\circ \mathfrak{T}(\!|HomDom|\!)(\!|Obj|\!).\ L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a\ b)$,
$Hom_{O.C\alpha}\mathfrak{K}(\!|HomCod|\!)(c,-) \circ_{CF} \mathfrak{K}$,
$Hom_{O.C\alpha}\mathfrak{T}(\!|HomCod|\!)(a,-) \circ_{CF} \mathfrak{T}$,
$\mathfrak{T}(\!|HomDom|\!)$,
$cat\text{-}Set\ \alpha$
]$_\circ$

Components.

lemma *L-10-5-υ-components*:
shows *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(\!|NTMap|\!)$ =
$(\lambda b \in_\circ \mathfrak{T}(\!|HomDom|\!)(\!|Obj|\!).\ L\text{-}10\text{-}5\text{-}\upsilon\text{-}arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a\ b)$
and *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(\!|NTDom|\!) = Hom_{O.C\alpha}\mathfrak{K}(\!|HomCod|\!)(c,-) \circ_{CF} \mathfrak{K}$
and *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(\!|NTCod|\!) = Hom_{O.C\alpha}\mathfrak{T}(\!|HomCod|\!)(a,-) \circ_{CF} \mathfrak{T}$
and *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(\!|NTDGDom|\!) = \mathfrak{T}(\!|HomDom|\!)$
and *L-10-5-υ* $\alpha\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a(\!|NTDGCod|\!) = cat\text{-}Set\ \alpha$
⟨*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

7.6.2 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* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{K}$ ⟨*proof*⟩
interpretation $\mathfrak{T}$: *is-functor* α $\mathfrak{B}$ $\mathfrak{A}$ $\mathfrak{T}$ ⟨*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

7.6.3 *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 $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and τ'-*def*: $\tau' = ntcf\text{-}arrow\ \tau$
and τ: $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and *a*: $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows *L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ *c* τ' *a* :
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
(**is** ‹*?L-10-5-υ* : *?H-*$\mathfrak{C}$ *c* $\circ_{CF}$ $\mathfrak{K} \mapsto_{CF}$ *?H-*$\mathfrak{A}$ *a* $\circ_{CF}$ $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$›)
⟨*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 $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and $\tau' = ntcf\text{-}arrow\ \tau$
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 $\mathfrak{C}' = cat\text{-}Set\ \alpha$
and $\alpha' = \alpha$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c\ {}_O\!\bigsqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows *L-10-5-υ* α $\mathfrak{T}$ $\mathfrak{K}$ *c* τ' *a* : $\mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B}' \mapsto\mapsto_{C_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

7.7 Lemma X.5: *L-10-5-χ-arrow*

7.7.1 Definition and elementary properties

definition *L-10-5-χ-arrow*
where *L-10-5-χ-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ *c a* =
[
$(\lambda\upsilon\in_\circ L\text{-}10\text{-}5\text{-}N\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\lfloor ObjMap \rfloor)(\lfloor a \rfloor).\ ntcf\text{-}arrow\ (L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a))$,
L-10-5-N α β $\mathfrak{T}$ $\mathfrak{K}$ $c(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$,
cf-Cone α β $(\mathfrak{T} \circ_{CF} c\ {}_O\!\bigsqcap_{CF} \mathfrak{K})(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
]$_\circ$

Components.

lemma *L-10-5-χ-arrow-components*:
shows *L-10-5-χ-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ *c* $a(\lfloor ArrVal \rfloor)$ =
$(\lambda\upsilon\in_\circ L\text{-}10\text{-}5\text{-}N\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\lfloor ObjMap \rfloor)(\lfloor a \rfloor).\ ntcf\text{-}arrow\ (L\text{-}10\text{-}5\text{-}\tau\ \mathfrak{T}\ \mathfrak{K}\ c\ \upsilon\ a))$
and *L-10-5-χ-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ *c* $a(\lfloor ArrDom \rfloor)$ = *L-10-5-N* α β $\mathfrak{T}$ $\mathfrak{K}$ $c(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and *L-10-5-χ-arrow* α β $\mathfrak{T}$ $\mathfrak{K}$ *c* $a(\lfloor ArrCod \rfloor)$ =

cf-Cone α β $(\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF} \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemmas [*cat-Kan-cs-simps*] = *L-10-5-χ-arrow-components*(*2*,*3*)

7.7.2 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 $\mathcal{Z}$ 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 $\mathcal{D}_{\circ}$ (*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*: $\upsilon' = ntcf\text{-}arrow\ \upsilon$
and *υ*: υ :
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
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*: $\upsilon\tau a' = ntcf\text{-}arrow\ \upsilon\tau a$
and *υτa*: $\upsilon\tau a$:
$Hom_{O.C\alpha}\mathfrak{C}(c,-) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{A}(a,-) \circ_{CF} \mathfrak{T}$:
$\mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
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*⟩

7.8 Lemma X.5: *L-10-5-χ′-arrow*

7.8.1 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 $_O\bigsqcap_{CF}$ $\mathfrak{K}$)(|*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 $_O\bigsqcap_{CF}$ $\mathfrak{K}$)(|*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 $_O\bigsqcap_{CF}$ $\mathfrak{K}$)(|*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 $_O\bigsqcap_{CF}$ $\mathfrak{K}$)(|*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*⟩

7.8.2 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 α $\in_\circ$ β
and τ: τ : a $<_{CF.cone}$ $\mathfrak{T}$ $\circ_{CF}$ c $_O\bigsqcap_{CF}$ $\mathfrak{K}$: 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 $_O\bigsqcap_{CF}$ $\mathfrak{K}$))
⟨*proof*⟩

lemma *L-10-5-χ′-arrow-ArrVal-app′*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β
and α $\in_\circ$ β
and τ'-*def*: τ' = *ntcf-arrow* τ
and τ: τ : a $<_{CF.cone}$ $\mathfrak{T}$ $\circ_{CF}$ c $_O\bigsqcap_{CF}$ $\mathfrak{K}$: 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*⟩

7.8.3 *L-10-5-χ′-arrow* is an isomorphism in the category *Set*

lemma *L-10-5-χ′-arrow-is-arr-isomorphism*:
assumes $\mathcal{Z}$ β
and α $\in_\circ$ β
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* $\alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a$:
cf-Cone $\alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!) \mapsto_{iso} cat\text{-}Set\ \beta$
L-10-5-N $\alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
(
is
‹
?L-10-5-χ′-arrow :
cf-Cone $\alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!) \mapsto_{iso} cat\text{-}Set\ \beta$
?L-10-5-N$(\!|ObjMap|\!)(\!|a|\!)$
›
)
⟨*proof*⟩

lemma *L-10-5-χ′-arrow-is-arr-isomorphism′*[*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 $A = cf\text{-}Cone\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!)$
and $B = L\text{-}10\text{-}5\text{-}N\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
and $\mathfrak{C}' = cat\text{-}Set\ \beta$
shows *L-10-5-χ′-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 *L-10-5-χ′-arrow* $\alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a$:
cf-Cone $\alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!) \mapsto_{cat\text{-}Set\ \beta}$
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 $A = cf\text{-}Cone\ \alpha\ \beta\ (\mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF}\ \mathfrak{K})(\!|ObjMap|\!)(\!|a|\!)$
and $B = L\text{-}10\text{-}5\text{-}N\ \alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c(\!|ObjMap|\!)(\!|a|\!)$
and $\mathfrak{C}' = cat\text{-}Set\ \beta$
shows *L-10-5-χ′-arrow* $\alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c\ a : A \mapsto_{\mathfrak{C}'} B$
⟨*proof*⟩

7.9 Lemma X.5: *L-10-5-χ*

7.9.1 Definition and elementary properties

definition *L-10-5-χ* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *L-10-5-χ* $\alpha\ \beta\ \mathfrak{T}\ \mathfrak{K}\ c =$

[
(λa∈ₒ𝔗(|HomCod|)(|Obj|). L-10-5-χ′-arrow α β 𝔗 ℜ c a),
cf-Cone α β (𝔗 ∘_CF c _O∏_CF ℜ),
L-10-5-N α β 𝔗 ℜ c,
op-cat (𝔗(|HomCod|)),
cat-Set β
]ₒ

Components.

lemma *L-10-5-χ-components*:
shows *L-10-5-χ* α β 𝔗 ℜ c(|*NTMap*|) =
(λa∈ₒ𝔗(|*HomCod*|)(|*Obj*|). *L-10-5-χ′-arrow* α β 𝔗 ℜ c a)
and [*cat-Kan-cs-simps*]:
L-10-5-χ α β 𝔗 ℜ c(|*NTDom*|) = *cf-Cone* α β (𝔗 ∘_CF c _O∏_CF ℜ)
and [*cat-Kan-cs-simps*]:
L-10-5-χ α β 𝔗 ℜ c(|*NTCod*|) = *L-10-5-N* α β 𝔗 ℜ c
and *L-10-5-χ* α β 𝔗 ℜ c(|*NTDGDom*|) = *op-cat* (𝔗(|*HomCod*|))
and [*cat-Kan-cs-simps*]: *L-10-5-χ* α β 𝔗 ℜ c(|*NTDGCod*|) = *cat-Set* β
⟨*proof*⟩

context
fixes α 𝔄 𝔅 𝔗
assumes 𝔗: 𝔗 : 𝔅 ↦↦_Cα 𝔄
begin

interpretation *is-functor* α 𝔅 𝔄 𝔗 ⟨*proof*⟩

lemmas *L-10-5-χ-components′* =
L-10-5-χ-components[**where** 𝔗=𝔗, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *L-10-5-χ-components′*(*4*)

end

7.9.2 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 𝔄 𝔅 𝔗
assumes 𝔗: 𝔗 : 𝔅 ↦↦_Cα 𝔄
begin

interpretation *is-functor* α 𝔅 𝔄 𝔗 ⟨*proof*⟩

mk-VLambda *L-10-5-χ-components*(*1*)[**where** 𝔗=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

7.9.3 *L-10-5-χ* is a natural isomorphism

lemma *L-10-5-χ-is-iso-ntcf*:
— See lemma on page 245 in [8].
assumes 𝒵 β
and α ∈ₒ β

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*⟩

7.10 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 [8].
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 [8].
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*⟩

7.11 The limit and the colimit for the pointwise Kan extensions

7.11.1 Definition and elementary properties

See Theorem 3 in Chapter X-5 in [8].

definition *the-pw-cat-rKe-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *the-pw-cat-rKe-limit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ 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* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ 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* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $c(\!|UObj|\!) = \mathfrak{G}(\!|ObjMap|\!)(\!|c|\!)$
and *the-pw-cat-rKe-limit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $c(\!|UArr|\!) =$
(
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* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $c(\!|UObj|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|c|\!)$
and *the-pw-cat-lKe-colimit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $c(\!|UArr|\!) =$ *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

7.11.2 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* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $c(\!|UArr|\!)$:
the-pw-cat-rKe-limit α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$ $c(\!|UObj|\!) <_{CF.lim} \mathfrak{T} \circ_{CF} c\ {}_{O}\!\sqcap_{CF} \mathfrak{K}$:
$c \downarrow_{CF} \mathfrak{K} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
⟨*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* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $c(\!|UArr|\!)$:
$\mathfrak{T} \circ_{CF} \mathfrak{K}\ {}_{CF}\!\sqcap_{O} c >_{CF.colim}$ *the-pw-cat-lKe-colimit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$ $c(\!|UObj|\!)$:
$\mathfrak{K}\ {}_{CF}\!\downarrow c \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
⟨*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* α $\mathfrak{T}$ $\mathfrak{K}$ (*the-pw-cat-rKe-limit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$) :
the-cf-rKe α $\mathfrak{T}$ $\mathfrak{K}$ (*the-pw-cat-rKe-limit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{G}$) $\circ_{CF}$ $\mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T}$:
$\mathfrak{B} \mapsto_{C} \mathfrak{C} \mapsto_{C} \mathfrak{A}$
⟨*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* α $\mathfrak{T}$ $\mathfrak{K}$ (*the-pw-cat-lKe-colimit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$) :
$\mathfrak{T} \mapsto_{CF.lKe\alpha}$ *the-cf-lKe* α $\mathfrak{T}$ $\mathfrak{K}$ (*the-pw-cat-lKe-colimit* α $\mathfrak{K}$ $\mathfrak{T}$ $\mathfrak{F}$) $\circ_{CF}$ $\mathfrak{K}$:
$\mathfrak{B} \mapsto_{C} \mathfrak{C} \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

8 Pointwise Kan extensions: application example

8.1 Background

The application example presented in this section is based on Exercise 6.1.ii in [13]. 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 $f : a \mapsto_{\textit{cat-ordinal}\ (2_{\mathbb{N}})}\ b$
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 $f : a \mapsto_{\textit{cat-ordinal}\ (3_{\mathbb{N}})}\ b$
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*⟩

8.2 $\mathfrak{K}23$

8.2.1 Definition and elementary properties

definition $\mathfrak{K}23 :: V$
where $\mathfrak{K}23 =$
[
($\lambda a \in_{\circ} \textit{cat-ordinal}\ (2_{\mathbb{N}})(\!|\textit{Obj}|\!).\ \textit{if}\ a = 0\ \textit{then}\ 0\ \textit{else}\ 2_{\mathbb{N}}$),
(
$\lambda f \in_{\circ} \textit{cat-ordinal}\ (2_{\mathbb{N}})(\!|\textit{Arr}|\!).$
$\textit{if}\ f = [0,\ 0]_{\circ} \Rightarrow [0,\ 0]_{\circ}$
$| f = [0,\ 1_{\mathbb{N}}]_{\circ} \Rightarrow [0,\ 2_{\mathbb{N}}]_{\circ}$
$| f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_{\circ} \Rightarrow [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_{\circ}$
$| \textit{otherwise} \Rightarrow 0$
),
cat-ordinal $(2_{\mathbb{N}})$,
cat-ordinal $(3_{\mathbb{N}})$
]$_{\circ}$

Components.

lemma $\mathfrak{K}23$-*components*:
shows $\mathfrak{K}23(\!|\textit{ObjMap}|\!) = (\lambda a \in_{\circ} \textit{cat-ordinal}\ (2_{\mathbb{N}})(\!|\textit{Obj}|\!).\ \textit{if}\ a = 0\ \textit{then}\ 0\ \textit{else}\ 2_{\mathbb{N}})$
and $\mathfrak{K}23(\!|\textit{ArrMap}|\!) =$
(
$\lambda f \in_{\circ} \textit{cat-ordinal}\ (2_{\mathbb{N}})(\!|\textit{Arr}|\!).$
$\textit{if}\ f = [0,\ 0]_{\circ} \Rightarrow [0,\ 0]_{\circ}$
$| f = [0,\ 1_{\mathbb{N}}]_{\circ} \Rightarrow [0,\ 2_{\mathbb{N}}]_{\circ}$
$| f = [1_{\mathbb{N}},\ 1_{\mathbb{N}}]_{\circ} \Rightarrow [2_{\mathbb{N}},\ 2_{\mathbb{N}}]_{\circ}$
$| \textit{otherwise} \Rightarrow 0$
)

and [*cat-Kan-cs-simps*]: $\mathfrak{K}23(\lfloor HomDom \rfloor) = cat\text{-}ordinal\ (2_{\mathbb{N}})$
and [*cat-Kan-cs-simps*]: $\mathfrak{K}23(\lfloor HomCod \rfloor) = cat\text{-}ordinal\ (3_{\mathbb{N}})$
⟨*proof*⟩

8.2.2 Object map

mk-VLambda $\mathfrak{K}23\text{-}components(1)$
|*vsv* $\mathfrak{K}23\text{-}ObjMap\text{-}vsv$[*cat-Kan-cs-intros*]|
|*vdomain* $\mathfrak{K}23\text{-}ObjMap\text{-}vdomain$[*cat-Kan-cs-simps*]|
|*app* $\mathfrak{K}23\text{-}ObjMap\text{-}app$|

lemma $\mathfrak{K}23\text{-}ObjMap\text{-}app\text{-}0$[*cat-Kan-cs-simps*]:
assumes $x = 0$
shows $\mathfrak{K}23(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = 0$
⟨*proof*⟩

lemma $\mathfrak{K}23\text{-}ObjMap\text{-}app\text{-}1$[*cat-Kan-cs-simps*]:
assumes $x = 1_{\mathbb{N}}$
shows $\mathfrak{K}23(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = 2_{\mathbb{N}}$
⟨*proof*⟩

8.2.3 Arrow map

mk-VLambda $\mathfrak{K}23\text{-}components(2)$
|*vsv* $\mathfrak{K}23\text{-}ArrMap\text{-}vsv$[*cat-Kan-cs-intros*]|
|*vdomain* $\mathfrak{K}23\text{-}ArrMap\text{-}vdomain$[*cat-Kan-cs-simps*]|
|*app* $\mathfrak{K}23\text{-}ArrMap\text{-}app$|

lemma $\mathfrak{K}23\text{-}ArrMap\text{-}app\text{-}00$[*cat-Kan-cs-simps*]:
assumes $f = [0, 0]_{\circ}$
shows $\mathfrak{K}23(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = [0, 0]_{\circ}$
⟨*proof*⟩

lemma $\mathfrak{K}23\text{-}ArrMap\text{-}app\text{-}01$[*cat-Kan-cs-simps*]:
assumes $f = [0, 1_{\mathbb{N}}]_{\circ}$
shows $\mathfrak{K}23(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = [0, 2_{\mathbb{N}}]_{\circ}$
⟨*proof*⟩

lemma $\mathfrak{K}23\text{-}ArrMap\text{-}app\text{-}11$[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ}$
shows $\mathfrak{K}23(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
⟨*proof*⟩

8.2.4 $\mathfrak{K}23$ is a tiny functor

lemma (in $\mathcal{Z}$**)** $\mathfrak{K}23\text{-}is\text{-}functor$: $\mathfrak{K}23 : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} cat\text{-}ordinal\ (3_{\mathbb{N}})$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $\mathfrak{K}23\text{-}is\text{-}functor'$[*cat-Kan-cs-intros*]:
assumes $\mathfrak{A}' = cat\text{-}ordinal\ (2_{\mathbb{N}})$
and $\mathfrak{B}' = cat\text{-}ordinal\ (3_{\mathbb{N}})$
shows $\mathfrak{K}23 : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = $\mathcal{Z}.\mathfrak{K}23\text{-}is\text{-}functor'$

lemma (in $\mathcal{Z}$**)** $\mathfrak{K}23\text{-}is\text{-}tiny\text{-}functor$:
$\mathfrak{K}23 : cat\text{-}ordinal\ (2_{\mathbb{N}}) \mapsto\mapsto_{C.tiny\alpha} cat\text{-}ordinal\ (3_{\mathbb{N}})$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ℜ23-is-tiny-functor'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{A}'$ = *cat-ordinal* ($2_{\mathbb{N}}$)
and $\mathfrak{B}'$ = *cat-ordinal* ($3_{\mathbb{N}}$)
shows *ℜ23* : $\mathfrak{A}' \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-Kan-cs-intros*] = $\mathcal{Z}$.*ℜ23-is-tiny-functor'*

8.3 *LK23*: the functor associated with the left Kan extension along *ℜ23*

8.3.1 Definition and elementary properties

definition *LK23* :: $V \Rightarrow V$
where *LK23* $\mathfrak{F}$ =
[
(
$\lambda a \in_{\circ}$ *cat-ordinal* ($3_{\mathbb{N}}$)(⌊*Obj*⌋).
if $a = 0 \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor)$
| *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Obj \rfloor)$
),
(
$\lambda f \in_{\circ}$ *cat-ordinal* ($3_{\mathbb{N}}$)(⌊*Arr*⌋).
if $f = [0, 0]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [0, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [0, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Arr \rfloor)$
),
cat-ordinal ($3_{\mathbb{N}}$),
$\mathfrak{F}(\lfloor HomCod \rfloor)$
]$_{\circ}$

Components.

lemma *LK23-components*:
shows *LK23* $\mathfrak{F}(\lfloor ObjMap \rfloor)$ =
(
$\lambda a \in_{\circ}$ *cat-ordinal* ($3_{\mathbb{N}}$)(⌊*Obj*⌋).
if $a = 0 \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor)$
| *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Obj \rfloor)$
)
and *LK23* $\mathfrak{F}(\lfloor ArrMap \rfloor)$ =
(
$\lambda f \in_{\circ}$ *cat-ordinal* ($3_{\mathbb{N}}$)(⌊*Arr*⌋).
if $f = [0, 0]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [0, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [0, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
| $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
| *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Arr \rfloor)$
)

and *LK23* $\mathfrak{F}(\!|HomDom|\!)$ = *cat-ordinal* $(3_{\mathbb{N}})$
and *LK23* $\mathfrak{F}(\!|HomCod|\!) = \mathfrak{F}(\!|HomCod|\!)$
⟨*proof*⟩

context *is-functor*
begin

lemmas *LK23-components′* = *LK23-components*[**where** $\mathfrak{F}$=$\mathfrak{F}$, *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*)

8.3.2 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* $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|0|\!)$
⟨*proof*⟩

lemma *LK23-ObjMap-app-1*[*cat-Kan-cs-simps*]:
assumes *a* = $1_{\mathbb{N}}$
shows *LK23* $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|0|\!)$
⟨*proof*⟩

lemma *LK23-ObjMap-app-2*[*cat-Kan-cs-simps*]:
assumes *a* = $2_{\mathbb{N}}$
shows *LK23* $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|1_{\mathbb{N}}|\!)$
⟨*proof*⟩

8.3.3 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* $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 0|\!)_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-01*[*cat-Kan-cs-simps*]:
assumes *f* = $[0, 1_{\mathbb{N}}]_{\circ}$
shows *LK23* $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 0|\!)_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-02*[*cat-Kan-cs-simps*]:
assumes *f* = $[0, 2_{\mathbb{N}}]_{\circ}$
shows *LK23* $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|0, 1_{\mathbb{N}}|\!)_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ}$
shows *LK23* $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-12*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows *LK23* $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
⟨*proof*⟩

lemma *LK23-ArrMap-app-22*[*cat-Kan-cs-simps*]:
assumes $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows *LK23* $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
⟨*proof*⟩

8.3.4 *LK23* is a functor

lemma *cat-LK23-is-functor*:
assumes $\mathfrak{F} : \mathit{cat\text{-}ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *LK23* $\mathfrak{F} : \mathit{cat\text{-}ordinal}\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-LK23-is-functor'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F} : \mathit{cat\text{-}ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = \mathit{cat\text{-}ordinal}\ (3_{\mathbb{N}})$
shows *LK23* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.3.5 The fundamental property of *LK23*

lemma *cf-comp-LK23-ℜ23*[*cat-Kan-cs-simps*]:
assumes $\mathfrak{F} : \mathit{cat\text{-}ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *LK23* $\mathfrak{F} \circ_{CF} \mathfrak{K}23 = \mathfrak{F}$
⟨*proof*⟩

8.4 *RK23*: the functor associated with the right Kan extension along $\mathfrak{K}23$

8.4.1 Definition and elementary properties

definition *RK23* :: $V \Rightarrow V$
where *RK23* $\mathfrak{F}$ =
[
(
$\lambda a \in_{\circ} \mathit{cat\text{-}ordinal}\ (3_{\mathbb{N}})(\lfloor Obj \rfloor).$
if $a = 0 \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 0 \rfloor)$
$\mid a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor)$
$\mid a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor)$
$\mid$ *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Obj \rfloor)$
),
(
$\lambda f \in_{\circ} \mathit{cat\text{-}ordinal}\ (3_{\mathbb{N}})(\lfloor Arr \rfloor).$
if $f = [0, 0]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 0 \rfloor)_{\bullet}$
$\mid f = [0, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
$\mid f = [0, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 0, 1_{\mathbb{N}} \rfloor)_{\bullet}$
$\mid f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
$\mid f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
$\mid f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}}, 1_{\mathbb{N}} \rfloor)_{\bullet}$
$\mid$ *otherwise* $\Rightarrow \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor Arr \rfloor)$

```
    ),
    cat-ordinal (3_ℕ),
    𝔉(|HomCod|)
  ]∘
```

Components.

```
lemma RK23-components:
  shows RK23 𝔉(|ObjMap|) =
    (
      λa∈∘cat-ordinal (3_ℕ)(|Obj|).
        if a = 0 ⇒ 𝔉(|ObjMap|)(|0|)
         | a = 1_ℕ ⇒ 𝔉(|ObjMap|)(|1_ℕ|)
         | a = 2_ℕ ⇒ 𝔉(|ObjMap|)(|1_ℕ|)
         | otherwise ⇒ 𝔉(|HomCod|)(|Obj|)
    )
    and RK23 𝔉(|ArrMap|) =
      (
        λf∈∘cat-ordinal (3_ℕ)(|Arr|).
         if f = [0, 0]∘ ⇒ 𝔉(|ArrMap|)(|0, 0|)•
          | f = [0, 1_ℕ]∘ ⇒ 𝔉(|ArrMap|)(|0, 1_ℕ|)•
          | f = [0, 2_ℕ]∘ ⇒ 𝔉(|ArrMap|)(|0, 1_ℕ|)•
          | f = [1_ℕ, 1_ℕ]∘ ⇒ 𝔉(|ArrMap|)(|1_ℕ, 1_ℕ|)•
          | f = [1_ℕ, 2_ℕ]∘ ⇒ 𝔉(|ArrMap|)(|1_ℕ, 1_ℕ|)•
          | f = [2_ℕ, 2_ℕ]∘ ⇒ 𝔉(|ArrMap|)(|1_ℕ, 1_ℕ|)•
          | otherwise ⇒ 𝔉(|HomCod|)(|Arr|)
      )
    and RK23 𝔉(|HomDom|) = cat-ordinal (3_ℕ)
    and RK23 𝔉(|HomCod|) = 𝔉(|HomCod|)
  ⟨proof⟩

context is-functor
begin

lemmas RK23-components′ = RK23-components[where 𝔉=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)
```

8.4.2 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 𝔉(|ObjMap|)(|a|) = 𝔉(|ObjMap|)(|0|)
  ⟨proof⟩

lemma RK23-ObjMap-app-1[cat-Kan-cs-simps]:
  assumes a = 1_ℕ
  shows RK23 𝔉(|ObjMap|)(|a|) = 𝔉(|ObjMap|)(|1_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*⟩

8.4.3 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*⟩

8.4.4 *RK23* is a functor

lemma *cat-RK23-is-functor*:
assumes $\mathfrak{F} : \textit{cat-ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *RK23* $\mathfrak{F} : \textit{cat-ordinal}\ (3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cat-RK23-is-functor'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F} : \textit{cat-ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = \textit{cat-ordinal}\ (3_{\mathbb{N}})$
shows *RK23* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.4.5 The fundamental property of *RK23*

lemma *cf-comp-RK23-ℜ23*[*cat-Kan-cs-simps*]:
assumes $\mathfrak{F} : \textit{cat-ordinal}\ (2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows *RK23* $\mathfrak{F} \circ_{CF} \mathfrak{K}23 = \mathfrak{F}$
⟨*proof*⟩

8.5 *RK-σ23*: towards the universal property of the right Kan extension along $\mathfrak{K}23$

8.5.1 Definition and elementary properties

definition *RK-σ23* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'$ =
[
(
$\lambda a \in_\circ$ *cat-ordinal* $(3_\mathbb{N})(\lfloor Obj \rfloor)$.
if $a = 0 \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 1_\mathbb{N} \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_\mathbb{N} \rfloor) \circ_{A\mathfrak{T}(\lfloor HomCod \rfloor)} \mathfrak{F}'(\lfloor ArrMap \rfloor)(\lfloor 1_\mathbb{N}, 2_\mathbb{N} \rfloor)_\bullet$
| $a = 2_\mathbb{N} \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_\mathbb{N} \rfloor)$
| *otherwise* $\Rightarrow \mathfrak{T}(\lfloor HomCod \rfloor)(\lfloor Arr \rfloor)$
),
$\mathfrak{F}'$,
RK23 $\mathfrak{T}$,
cat-ordinal $(3_\mathbb{N})$,
$\mathfrak{F}'(\lfloor HomCod \rfloor)$
]$_\circ$

Components.

lemma *RK-σ23-components*:
shows *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTMap \rfloor)$ =
(
$\lambda a \in_\circ$ *cat-ordinal* $(3_\mathbb{N})(\lfloor Obj \rfloor)$.
if $a = 0 \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 0 \rfloor)$
| $a = 1_\mathbb{N} \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_\mathbb{N} \rfloor) \circ_{A\mathfrak{T}(\lfloor HomCod \rfloor)} \mathfrak{F}'(\lfloor ArrMap \rfloor)(\lfloor 1_\mathbb{N}, 2_\mathbb{N} \rfloor)_\bullet$
| $a = 2_\mathbb{N} \Rightarrow \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_\mathbb{N} \rfloor)$
| *otherwise* $\Rightarrow \mathfrak{T}(\lfloor HomCod \rfloor)(\lfloor Arr \rfloor)$
)
and *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTDom \rfloor) = \mathfrak{F}'$
and *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTCod \rfloor) =$ *RK23* $\mathfrak{T}$
and *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTDGDom \rfloor) =$ *cat-ordinal* $(3_\mathbb{N})$
and *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTDGCod \rfloor) = \mathfrak{F}'(\lfloor HomCod \rfloor)$
⟨*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 *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

8.5.2 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-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTMap \rfloor)(\lfloor a \rfloor) = \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 0 \rfloor)$
⟨*proof*⟩

lemma (**in** *is-functor*) *RK-σ23-NTMap-app-1*[*cat-Kan-cs-simps*]:
assumes $a = 1_{\mathbb{N}}$
shows *RK-σ23* $\mathfrak{F}$ ε' $\mathfrak{F}'(\lfloor NTMap \rfloor)(\lfloor a \rfloor) = \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}'(\lfloor ArrMap \rfloor)(\lfloor 1_{\mathbb{N}},\ 2_{\mathbb{N}} \rfloor)_{\bullet}$
⟨*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-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'(\lfloor NTMap \rfloor)(\lfloor a \rfloor) = \varepsilon'(\lfloor NTMap \rfloor)(\lfloor 1_{\mathbb{N}} \rfloor)$
⟨*proof*⟩

8.5.3 *RK-σ23* is a natural transformation

lemma *RK-σ23-is-ntcf*:
assumes $\mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and ε' : $\mathfrak{F}' \circ_{CF}$ *ℜ23* $\mapsto_{CF} \mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'$: $\mathfrak{F}' \mapsto_{CF}$ *RK23* $\mathfrak{T}$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *RK-σ23-is-ntcf'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and ε' : $\mathfrak{F}' \circ_{CF}$ *ℜ23* $\mapsto_{CF} \mathfrak{T}$: *cat-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-ordinal* $(3_{\mathbb{N}})$
shows *RK-σ23* $\mathfrak{T}$ ε' $\mathfrak{F}'$: $\mathfrak{G}' \mapsto_{CF} \mathfrak{H}'$: $\mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

8.6 The right Kan extension along *ℜ23*

lemma *ε23-is-cat-rKe*:
assumes $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* $\mathfrak{T}$:
RK23 $\mathfrak{T} \circ_{CF}$ *ℜ23* $\mapsto_{CF.rKe\alpha} \mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto_{C}$ *cat-ordinal* $(3_{\mathbb{N}}) \mapsto_{C} \mathfrak{A}$
⟨*proof*⟩

8.7 *LK-σ23*: towards the universal property of the left Kan extension along *ℜ23*

8.7.1 Definition and elementary properties

definition *LK-σ23* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *LK-σ23* $\mathfrak{T}$ η' $\mathfrak{F}'$ =
[

```
    (
      λa∈₀ cat-ordinal (3_ℕ)(|Obj|).
       if a = 0 ⇒ η′(|NTMap|)(|0|)
        | a = 1_ℕ ⇒ 𝔉′(|ArrMap|)(|0, 1_ℕ|)• ∘_{A𝔗(|HomCod|)} η′(|NTMap|)(|0|)
        | a = 2_ℕ ⇒ η′(|NTMap|)(|1_ℕ|)
        | otherwise ⇒ 𝔗(|HomCod|)(|Arr|)
    ),
    LK23 𝔗,
    𝔉′,
    cat-ordinal (3_ℕ),
    𝔉′(|HomCod|)
  ]₀
```

Components.

```
lemma LK-σ23-components:
  shows LK-σ23 𝔗 η′ 𝔉′(|NTMap|) =
    (
      λa∈₀ cat-ordinal (3_ℕ)(|Obj|).
        if a = 0 ⇒ η′(|NTMap|)(|0|)
         | a = 1_ℕ ⇒ 𝔉′(|ArrMap|)(|0, 1_ℕ|)• ∘_{A𝔗(|HomCod|)} η′(|NTMap|)(|0|)
         | a = 2_ℕ ⇒ η′(|NTMap|)(|1_ℕ|)
         | otherwise ⇒ 𝔗(|HomCod|)(|Arr|)
    )
    and LK-σ23 𝔗 η′ 𝔉′(|NTDom|) = LK23 𝔗
    and LK-σ23 𝔗 η′ 𝔉′(|NTCod|) = 𝔉′
    and LK-σ23 𝔗 η′ 𝔉′(|NTDGDom|) = cat-ordinal (3_ℕ)
    and LK-σ23 𝔗 η′ 𝔉′(|NTDGCod|) = 𝔉′(|HomCod|)
  ⟨proof⟩

context
  fixes α 𝔄 𝔉′ 𝔗
  assumes 𝔉′: 𝔉′ : cat-ordinal (3_ℕ) ↦↦_{Cα} 𝔄
    and 𝔗: 𝔗 : cat-ordinal (2_ℕ) ↦↦_{Cα} 𝔄
begin

interpretation 𝔉′: is-functor α ‹cat-ordinal (3_ℕ)› 𝔄 𝔉′ ⟨proof⟩
interpretation 𝔗: is-functor α ‹cat-ordinal (2_ℕ)› 𝔄 𝔗 ⟨proof⟩

lemmas LK-σ23-components′ =
  LK-σ23-components[where 𝔉′=𝔉′ and 𝔗=𝔗, unfolded cat-cs-simps]

lemmas [cat-Kan-cs-simps] = LK-σ23-components′(2–5)

end
```

8.7.2 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 𝔗 η′ 𝔉′(|NTMap|)(|a|) = η′(|NTMap|)(|0|)
  ⟨proof⟩

lemma (in is-functor) LK-σ23-NTMap-app-1[cat-Kan-cs-simps]:
```

assumes $a = 1_{\mathbb{N}}$
shows *LK-σ23* $\mathfrak{F}\ \eta'\ \mathfrak{F}'(\!|NTMap|\!)(\!|a|\!) = \mathfrak{F}'(\!|ArrMap|\!)(\!|0,\ 1_{\mathbb{N}}|\!)_{\bullet} \circ_{A\mathfrak{B}} \eta'(\!|NTMap|\!)(\!|0|\!)$
⟨*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-σ23* $\mathfrak{T}\ \eta'\ \mathfrak{F}'(\!|NTMap|\!)(\!|a|\!) = \eta'(\!|NTMap|\!)(\!|1_{\mathbb{N}}|\!)$
⟨*proof*⟩

8.7.3 *LK-σ23* is a natural transformation

lemma *LK-σ23-is-ntcf*:
assumes $\mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K}23$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *LK-σ23* $\mathfrak{T}\ \eta'\ \mathfrak{F}' : LK23\ \mathfrak{T} \mapsto_{CF} \mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *LK-σ23-is-ntcf'*[*cat-Kan-cs-intros*]:
assumes $\mathfrak{F}'$: *cat-ordinal* $(3_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K}23$: *cat-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-ordinal* $(3_{\mathbb{N}})$
shows *LK-σ23* $\mathfrak{T}\ \eta'\ \mathfrak{F}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

8.8 The left Kan extension along *ℜ23*

lemma *η23-is-cat-rKe*:
assumes $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* $\mathfrak{T}$:
$\mathfrak{T} \mapsto_{CF.lKe\alpha} LK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto_C$ *cat-ordinal* $(3_{\mathbb{N}}) \mapsto_C \mathfrak{A}$
⟨*proof*⟩

8.9 Pointwise Kan extensions along *ℜ23*

lemma *ε23-is-cat-pw-rKe*:
assumes $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* $\mathfrak{T}$:
$RK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23 \mapsto_{CF.rKe.pw\alpha} \mathfrak{T}$:
cat-ordinal $(2_{\mathbb{N}}) \mapsto_C$ *cat-ordinal* $(3_{\mathbb{N}}) \mapsto_C \mathfrak{A}$
⟨*proof*⟩

lemma *η23-is-cat-pw-lKe*:
assumes $\mathfrak{T}$: *cat-ordinal* $(2_{\mathbb{N}}) \mapsto\mapsto_{C\alpha} \mathfrak{A}$
shows *ntcf-id* $\mathfrak{T}$:
$\mathfrak{T} \mapsto_{CF.lKe.pw\alpha} LK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23$:
cat-ordinal $(2_{\mathbb{N}}) \mapsto_C$ *cat-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] P. Bodo. *Categories and Functors*. Academic Press, New York, 1970.

[4] F. Haftmann. Sketch-and-Explore, 2021. URL https://isabelle.in.tum.de/library/HOL/HOL-ex/Sketch_and_Explore.html.

[5] T. W. Hungerford. *Algebra*. Springer, New York, 2003. ISBN 978-0-387-90518-1.

[6] D. M. Kan. Adjoint Functors. *Transactions of the American Mathematical Society*, 87(2): 294–329, 1958.

[7] 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.

[8] 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.

[9] M. Milehins. Category Theory for ZFC in HOL II: Elementary Theory of 1-Categories. *Archive of Formal Proofs*, 2021.

[10] 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.

[11] L. C. Paulson. Natural Deduction as Higher-Order Resolution. *The Journal of Logic Programming*, 3(3):237–258, 1986.

[12] L. C. Paulson. Zermelo Fraenkel Set Theory in Higher-Order Logic. *Archive of Formal Proofs*, 2019.

[13] E. Riehl. *Category Theory in Context*. Emily Riehl, 2016.