

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

Mihails Milehins

February 6, 2026

Abstract

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

Acknowledgements

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

Contents

1	Introduction	7
2	Universal arrow	8
2.1	Background	8
2.2	Universal map	8
2.3	Universal arrow: definition and elementary properties	11
2.4	Uniqueness	12
2.5	Universal natural transformation	13
3	Limits and colimits	17
3.1	Background	17
3.2	Limit and colimit	17
3.3	Small limit and small colimit	20
3.4	Finite limit and finite colimit	22
3.5	Creation of limits	24
3.6	Preservation of limits and colimits	25
3.7	Continuous and cocontinuous functor	26
3.8	Tiny-continuous and tiny-cocontinuous functor	27
4	Initial and terminal objects as limits and colimits	28
4.1	Initial and terminal objects as limits/colimits of an empty diagram	28
4.2	Initial cone and terminal cocone	29
4.3	Initial and terminal objects as limits/colimits of the identity functor	31
5	Products and coproducts as limits and colimits	34
5.1	Product and coproduct	34
5.2	Small product and small coproduct	36
5.3	Finite product and finite coproduct	38
5.4	Product and coproduct of two objects	39
5.5	Projection cone	42
6	Pullbacks and pushouts as limits and colimits	44
6.1	Pullback and pushout	44
7	Equalizers and coequalizers as limits and colimits	48
7.1	Equalizer and coequalizer	48
7.2	Equalizer and coequalizer for two arrows	53
7.3	Equalizer cone	58
8	Pointed arrows and natural transformations	59
8.1	Pointed arrow	59
8.2	Pointed natural transformation	60
8.3	Inverse pointed natural transformation	61
9	Representable and corepresentable functors	64
9.1	Representable and corepresentable functors	64
9.2	Limits and colimits as universal cones	65

10 Completeness and cocompleteness	66
10.1 Limits by products and equalizers	66
10.2 Small-complete and small-cocomplete category	66
10.3 Finite-complete and finite-cocomplete category	68
11 Comma categories and universal constructions	69
11.1 Relationship between the universal arrows, initial objects and terminal objects . .	69
11.2 A projection for a comma category constructed from a functor and an object creates small limits	69
12 Category <i>Set</i> and universal constructions	70
12.1 Discrete functor with tiny maps to the category <i>Set</i>	70
12.2 Product cone and coproduct cocone for the category <i>Set</i>	70
12.3 Equalizer for the category <i>Set</i>	71
12.4 The category <i>Set</i> is small-complete	72
13 Adjoints	73
13.1 Background	73
13.2 Definition and elementary properties	73
13.3 Opposite adjunction	74
13.4 Unit	75
13.5 Counit	78
13.6 Counit-unit equations	81
13.7 Construction of an adjunction from universal morphisms from objects to functors	81
13.8 Construction of an adjunction from a functor and universal morphisms from objects to functors	83
13.9 Construction of an adjunction from universal morphisms from functors to objects	86
13.10 Construction of an adjunction from a functor and universal morphisms from functors to objects	87
13.11 Construction of an adjunction from the counit-unit equations	89
13.12 Adjoints are unique up to isomorphism	89
13.13 Further properties of the adjoint functors	93
13.14 Adjoints on limits	94
14 Simple Kan extensions	95
14.1 Background	95
14.2 Kan extension	95
14.3 Opposite universal arrow for Kan extensions	98
14.4 The Kan extension	99
14.5 Preservation of Kan extensions	104
14.6 All concepts are Kan extensions	106
15 Pointwise Kan extensions	108
15.1 Pointwise Kan extensions	108
15.2 Lemma X.5: <i>L-10-5-N</i>	110
15.3 Lemma X.5: <i>L-10-5-v-arrow</i>	112
15.4 Lemma X.5: <i>L-10-5-τ</i>	114
15.5 Lemma X.5: <i>L-10-5-v</i>	115
15.6 Lemma X.5: <i>L-10-5-χ-arrow</i>	117
15.7 Lemma X.5: <i>L-10-5-χ'-arrow</i>	118
15.8 Lemma X.5: <i>L-10-5-χ</i>	120

15.9	The existence of a canonical limit or a canonical colimit for the pointwise Kan extensions	121
15.10	The limit and the colimit for the pointwise Kan extensions	122
16	Pointwise Kan extensions: application example	124
16.1	Background	124
16.2	$\mathfrak{K}23$	124
16.3	$LK23$: the functor associated with the left Kan extension along $\mathfrak{K}23$	126
16.4	$RK23$: the functor associated with the right Kan extension along $\mathfrak{K}23$	128
16.5	$RK\text{-}\sigma 23$: towards the universal property of the right Kan extension along $\mathfrak{K}23$. .	131
16.6	The right Kan extension along $\mathfrak{K}23$	132
16.7	$LK\text{-}\sigma 23$: towards the universal property of the left Kan extension along $\mathfrak{K}23$. . .	132
16.8	The left Kan extension along $\mathfrak{K}23$	134
16.9	Pointwise Kan extensions along $\mathfrak{K}23$	134
References		135

1 Introduction

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

¹<https://ncatlab.org/nlab/show/universal+construction>

2 Universal arrow

2.1 Background

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

named-theorems *ua-field-simps*

definition $UObj :: V$ **where** [*ua-field-simps*]: $UObj = 0$

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

lemma [*cat-cs-simps*]:

shows $UObj\text{-simp}$: $[a, b]_{\circ}(\downarrow UObj) = a$

and $UArr\text{-simp}$: $[a, b]_{\circ}(\downarrow UArr) = b$

<proof>

2.2 Universal map

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

2.2.1 Definition and elementary properties

definition $umap\text{-of} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $umap\text{-of} \mathfrak{F} c r u d =$

[
 $(\lambda f' \in_{\circ} Hom (\mathfrak{F}(\downarrow HomDom))) r d. \mathfrak{F}(\downarrow ArrMap)(\downarrow f') \circ_{A\mathfrak{F}(\downarrow HomCod)} u$),
 $Hom (\mathfrak{F}(\downarrow HomDom)) r d$,
 $Hom (\mathfrak{F}(\downarrow HomCod)) c (\mathfrak{F}(\downarrow ObjMap)(\downarrow d))$
]_o

definition $umap\text{-fo} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

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

Components.

lemma (**in** *is-functor*) $umap\text{-of-components}$:

assumes $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\downarrow ObjMap)(\downarrow r)$

shows $umap\text{-of} \mathfrak{F} c r u d(\downarrow ArrVal) = (\lambda f' \in_{\circ} Hom \mathfrak{A} r d. \mathfrak{F}(\downarrow ArrMap)(\downarrow f') \circ_{A\mathfrak{B}} u)$

and $umap\text{-of} \mathfrak{F} c r u d(\downarrow ArrDom) = Hom \mathfrak{A} r d$

and $umap\text{-of} \mathfrak{F} c r u d(\downarrow ArrCod) = Hom \mathfrak{B} c (\mathfrak{F}(\downarrow ObjMap)(\downarrow d))$

<proof>

lemma (**in** *is-functor*) $umap\text{-fo-components}$:

assumes $u : \mathfrak{F}(\downarrow ObjMap)(\downarrow r) \mapsto_{\mathfrak{B}} c$

shows $umap\text{-fo} \mathfrak{F} c r u d(\downarrow ArrVal) = (\lambda f' \in_{\circ} Hom \mathfrak{A} d r. u \circ_{A\mathfrak{B}} \mathfrak{F}(\downarrow ArrMap)(\downarrow f'))$

and $umap\text{-fo} \mathfrak{F} c r u d(\downarrow ArrDom) = Hom \mathfrak{A} d r$

and $umap\text{-fo} \mathfrak{F} c r u d(\downarrow ArrCod) = Hom \mathfrak{B} (\mathfrak{F}(\downarrow ObjMap)(\downarrow d)) c$

<proof>

Universal maps for the opposite functor.

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

<proof>

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

<proof>

lemmas [*cat-op-simps*] =

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

2.2.2 Arrow value

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

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

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

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

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

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

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

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

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

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

<proof>

lemma (in *is-functor*) *cf-arr-Set-umap-fo*:

assumes *category* α \mathfrak{A}
and *category* α \mathfrak{B}
and $r : r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $d : d \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : u : \mathfrak{F}(\text{ObjMap})(|r|) \mapsto_{\mathfrak{B}} c$
shows *arr-Set* α (*umap-fo* $\mathfrak{F} c r u d$)

<proof>

lemma (in *is-functor*) *cf-umap-of-is-arr*:

assumes *category* α \mathfrak{A}
and *category* α \mathfrak{B}
and $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $d \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(|r|)$
shows *umap-of* $\mathfrak{F} c r u d : \text{Hom } \mathfrak{A} r d \mapsto_{\text{cat-Set } \alpha} \text{Hom } \mathfrak{B} c (\mathfrak{F}(\text{ObjMap})(|d|))$

<proof>

lemma (in *is-functor*) *cf-umap-of-is-arr'*:

assumes *category* α \mathfrak{A}
and *category* α \mathfrak{B}
and $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $d \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(|r|)$
and $A = \text{Hom } \mathfrak{A} r d$
and $B = \text{Hom } \mathfrak{B} c (\mathfrak{F}(\text{ObjMap})(|d|))$
and $\mathfrak{C} = \text{cat-Set } \alpha$
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 $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $d \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : \mathfrak{F}(\text{ObjMap})(|r|) \mapsto_{\mathfrak{B}} c$
shows *umap-fo* $\mathfrak{F} c r u d : \text{Hom } \mathfrak{A} d r \mapsto_{\text{cat-Set } \alpha} \text{Hom } \mathfrak{B} (\mathfrak{F}(\text{ObjMap})(|d|)) c$

<proof>

lemma (in *is-functor*) *cf-umap-fo-is-arr'*:

assumes *category* α \mathfrak{A}
and *category* α \mathfrak{B}
and $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $d \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : \mathfrak{F}(\text{ObjMap})(|r|) \mapsto_{\mathfrak{B}} c$
and $A = \text{Hom } \mathfrak{A} d r$
and $B = \text{Hom } \mathfrak{B} (\mathfrak{F}(\text{ObjMap})(|d|)) c$
and $\mathfrak{C} = \text{cat-Set } \alpha$
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 [9].

definition *universal-arrow-of* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *universal-arrow-of* $\mathfrak{F} \ c \ r \ u \longleftrightarrow$

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

definition *universal-arrow-fo* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u \equiv \text{universal-arrow-of } (\text{op-cf } \mathfrak{F}) \ c \ r \ u$

Rules.

mk-ide (in *is-functor*) **rf**

universal-arrow-of-def [**where** $\mathfrak{F} = \mathfrak{F}$, *unfolded cf-HomDom cf-HomCod*]
intro universal-arrow-ofI
dest universal-arrow-ofD [*dest*]
elim universal-arrow-ofE [*elim*]

lemma (in *is-functor*) *universal-arrow-foI*:

assumes $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : \mathfrak{F}(\text{ObjMap})(r) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r' \ u'. [\ [r' \in_{\circ} \mathfrak{A}(\text{Obj}); u' : \mathfrak{F}(\text{ObjMap})(r') \mapsto_{\mathfrak{B}} c]] \implies$
 $\exists ! f'. f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal})(f')$
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 $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : \mathfrak{F}(\text{ObjMap})(r) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r' \ u'. [\ [r' \in_{\circ} \mathfrak{A}(\text{Obj}); u' : \mathfrak{F}(\text{ObjMap})(r') \mapsto_{\mathfrak{B}} c]] \implies$
 $\exists ! f'. f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal})(f')$
<proof>

lemma (in *is-functor*) *universal-arrow-foE* [*elim*]:

assumes *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$
obtains $r \in_{\circ} \mathfrak{A}(\text{Obj})$
and $u : \mathfrak{F}(\text{ObjMap})(r) \mapsto_{\mathfrak{B}} c$
and $\bigwedge r' \ u'. [\ [r' \in_{\circ} \mathfrak{A}(\text{Obj}); u' : \mathfrak{F}(\text{ObjMap})(r') \mapsto_{\mathfrak{B}} c]] \implies$
 $\exists ! f'. f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal})(f')$
<proof>

Elementary properties.

lemma (in *is-functor*) *op-cf-universal-arrow-of* [*cat-op-simps*]:

universal-arrow-of $(\text{op-cf } \mathfrak{F}) \ c \ r \ u \longleftrightarrow \text{universal-arrow-fo } \mathfrak{F} \ c \ r \ u$
<proof>

lemma (in *is-functor*) *op-cf-universal-arrow-fo* [*cat-op-simps*]:

universal-arrow-fo $(\text{op-cf } \mathfrak{F}) \ c \ r \ u \longleftrightarrow \text{universal-arrow-of } \mathfrak{F} \ c \ r \ u$

<proof>

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

2.4 Uniqueness

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

lemma (in *is-functor*) *cf-universal-arrow-of-ex-is-iso-arr*:

— The proof is based on the ideas expressed in the proof of Theorem 5.2 in Chapter Introduction in [6].

assumes *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$ **and** *universal-arrow-of* $\mathfrak{F} \ c \ r' \ u'$
obtains f **where** $f : r \mapsto_{\text{iso}\mathfrak{A}} r'$ **and** $u' = \text{umap-of } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f)$
<proof>

lemma (in *is-functor*) *cf-universal-arrow-fo-ex-is-iso-arr*:

assumes *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$
and *universal-arrow-fo* $\mathfrak{F} \ c \ r' \ u'$
obtains f **where** $f : r' \mapsto_{\text{iso}\mathfrak{A}} r$ **and** $u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f)$
<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 $\exists ! f'. f' : r \mapsto_{\mathfrak{A}} r' \wedge u' = \text{umap-of } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f')$
<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 $\exists ! f'. f' : r' \mapsto_{\mathfrak{A}} r \wedge u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f')$
<proof>

lemma (in *is-functor*) *cf-universal-arrow-of-is-iso-arr*:

assumes *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$
and *universal-arrow-of* $\mathfrak{F} \ c \ r' \ u'$
and $f : r \mapsto_{\mathfrak{A}} r'$
and $u' = \text{umap-of } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f)$
shows $f : r \mapsto_{\text{iso}\mathfrak{A}} r'$
<proof>

lemma (in *is-functor*) *cf-universal-arrow-fo-is-iso-arr*:

assumes *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$
and *universal-arrow-fo* $\mathfrak{F} \ c \ r' \ u'$
and $f : r' \mapsto_{\mathfrak{A}} r$
and $u' = \text{umap-fo } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f)$
shows $f : r' \mapsto_{\text{iso}\mathfrak{A}} r$
<proof>

lemma (in *is-functor*) *universal-arrow-of-if-universal-arrow-of*:

assumes *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$
and $f : r \mapsto_{\text{iso}\mathfrak{A}} r'$
and $u' = \text{umap-of } \mathfrak{F} \ c \ r \ u \ r' (\text{ArrVal}) (f)$
shows *universal-arrow-of* $\mathfrak{F} \ c \ r' \ u'$
<proof>

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

2.5 Universal natural transformation

2.5.1 Definition and elementary properties

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

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

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

Components.

lemma *ntcf-ua-of-components*:

shows *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTMap}) = (\lambda d \in_{\circ} \mathfrak{F} (\text{HomDom}) (\text{Obj}). \text{umap-of } \mathfrak{F} \ c \ r \ u \ d)$
and *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDom}) = \text{Hom}_{O.C\alpha} \mathfrak{F} (\text{HomDom}) (r, -)$
and *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTCod}) = \text{Hom}_{O.C\alpha} \mathfrak{F} (\text{HomCod}) (c, -) \circ_{CF} \mathfrak{F}$
and *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDGDom}) = \mathfrak{F} (\text{HomDom})$
and *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDGCod}) = \text{cat-Set } \alpha$
<proof>

lemma *ntcf-ua-fo-components*:

shows *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTMap}) = (\lambda d \in_{\circ} \mathfrak{F} (\text{HomDom}) (\text{Obj}). \text{umap-fo } \mathfrak{F} \ c \ r \ u \ d)$
and *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDom}) = \text{Hom}_{O.C\alpha} \text{op-cat } (\mathfrak{F} (\text{HomDom})) (r, -)$
and *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTCod}) =$
 $\text{Hom}_{O.C\alpha} \text{op-cat } (\mathfrak{F} (\text{HomCod})) (c, -) \circ_{CF} \text{op-cf } \mathfrak{F}$
and *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDGDom}) = \text{op-cat } (\mathfrak{F} (\text{HomDom}))$
and *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u (\text{NTDGCod}) = \text{cat-Set } \alpha$
<proof>

context *is-functor*

begin

lemmas *ntcf-ua-of-components'* =

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

```

    ntcf-ua-fo  $\alpha$   $\mathfrak{F}$   $c$   $r$   $u$ ( $NTDom$ ) =  $Hom_{O.C\alpha}\mathfrak{A}(-, r)$ 
  and [cat-cs-simps]:
    ntcf-ua-fo  $\alpha$   $\mathfrak{F}$   $c$   $r$   $u$ ( $NTCod$ ) =  $Hom_{O.C\alpha}\mathfrak{B}(-, c) \circ_{CF} op-cf \mathfrak{F}$ 
  and [cat-cs-simps]: ntcf-ua-fo  $\alpha$   $\mathfrak{F}$   $c$   $r$   $u$ ( $NTDGDom$ ) =  $op-cat \mathfrak{A}$ 
  and [cat-cs-simps]: ntcf-ua-fo  $\alpha$   $\mathfrak{F}$   $c$   $r$   $u$ ( $NTDGCod$ ) =  $cat-Set \alpha$ 
  <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  $\alpha=\alpha$  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  $r: r \in_o \mathfrak{A}(\text{Obj})$  and  $c: c \in_o \mathfrak{B}(\text{Obj})$ 
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  $\alpha \mathfrak{A}$ 
  and category  $\alpha \mathfrak{B}$ 
  and  $r \in_o \mathfrak{A}(\text{Obj})$ 
  and  $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(r)$ 
  shows  $\mathcal{R}_o(\text{ntcf-ua-of } \alpha \mathfrak{F} \text{ } c \text{ } r \text{ } u(\text{NTMap})) \subseteq_o \text{cat-Set } \alpha(\text{Arr})$ 
  <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 $c \in_{\circ} \mathfrak{B}(\text{Obj})$

and $r \in_{\circ} \mathfrak{A}(\text{Obj})$

and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(r)$

and $f : a \mapsto_{\mathfrak{A}} b$

shows

$umap\text{-of } \mathfrak{F} \ c \ r \ u \ b \circ_{A \text{ cat-Set } \alpha} cf\text{-hom } \mathfrak{A} \ [\mathfrak{A}(\text{CIde})(r), f]_{\circ} =$

$cf\text{-hom } \mathfrak{B} \ [\mathfrak{B}(\text{CIde})(c), \mathfrak{F}(\text{ArrMap})(f)]_{\circ} \circ_{A \text{ cat-Set } \alpha} umap\text{-of } \mathfrak{F} \ c \ r \ u \ a$

(is $\langle ?uof\text{-}b \circ_{A \text{ cat-Set } \alpha} ?rf = ?cf \circ_{A \text{ cat-Set } \alpha} ?uof\text{-}a \rangle$)

<proof>

lemma *cf-umap-of-cf-hom-unit-commute*:

assumes *category* α \mathfrak{C}

and *category* α \mathfrak{D}

and $\mathfrak{F} : \mathfrak{C} \mapsto_{\mapsto} C\alpha \ \mathfrak{D}$

and $\mathfrak{G} : \mathfrak{D} \mapsto_{\mapsto} C\alpha \ \mathfrak{C}$

and $\eta : cf\text{-id } \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_{\mapsto} C\alpha \ \mathfrak{C}$

and $g : c' \mapsto_{\mathfrak{C}} c$

and $f : d \mapsto_{\mathfrak{D}} d'$

shows

$umap\text{-of } \mathfrak{G} \ c' \ (\mathfrak{F}(\text{ObjMap})(c')) \ (\eta(\text{NTMap})(c')) \ d' \circ_{A \text{ cat-Set } \alpha}$

$cf\text{-hom } \mathfrak{D} \ [\mathfrak{F}(\text{ArrMap})(g), f]_{\circ} =$

$cf\text{-hom } \mathfrak{C} \ [g, \mathfrak{G}(\text{ArrMap})(f)]_{\circ} \circ_{A \text{ cat-Set } \alpha}$

$umap\text{-of } \mathfrak{G} \ c \ (\mathfrak{F}(\text{ObjMap})(c)) \ (\eta(\text{NTMap})(c)) \ d$

(is $\langle ?uof\text{-}c'd' \circ_{A \text{ cat-Set } \alpha} ?\mathfrak{F}gf = ?g\mathfrak{G}f \circ_{A \text{ cat-Set } \alpha} ?uof\text{-}cd \rangle$)

<proof>

2.5.4 Universal natural transformation is a natural transformation

lemma (in *is-functor*) *cf-ntcf-ua-of-is-ntcf*:

assumes $r \in_{\circ} \mathfrak{A}(\text{Obj})$

and $u : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(r)$

shows *ntcf-ua-of* α $\mathfrak{F} \ c \ r \ u$:

$Hom_{O.C\alpha} \mathfrak{A}(r, -) \mapsto_{CF} Hom_{O.C\alpha} \mathfrak{B}(c, -) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto_{\mapsto} C\alpha \ \text{cat-Set } \alpha$

<proof>

lemma (in *is-functor*) *cf-ntcf-ua-fo-is-ntcf*:

assumes $r \in_{\circ} \mathfrak{A}(\text{Obj})$ **and** $u : \mathfrak{F}(\text{ObjMap})(r) \mapsto_{\mathfrak{B}} c$

shows *ntcf-ua-fo* α $\mathfrak{F} \ c \ r \ u$:

$Hom_{O.C\alpha} \mathfrak{A}(-, r) \mapsto_{CF} Hom_{O.C\alpha} \mathfrak{B}(-, c) \circ_{CF} op\text{-cf } \mathfrak{F} :$

$op\text{-cat } \mathfrak{A} \mapsto_{\mapsto} C\alpha \ \text{cat-Set } \alpha$

<proof>

2.5.5 Universal natural transformation and universal arrow

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

lemma (in *is-functor*) *cf-ntcf-ua-of-is-iso-ntcf*:

assumes *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$

shows *ntcf-ua-of* α $\mathfrak{F} \ c \ r \ u$:

$Hom_{O.C\alpha} \mathfrak{A}(r, -) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(c, -) \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto_{\mapsto} C\alpha \ \text{cat-Set } \alpha$

<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* $\alpha \ \mathfrak{F} \ c \ r \ u$:

$$\begin{aligned} & Hom_{O.C\alpha} \mathfrak{A}(-, r) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(-, c) \circ_{CF} op\text{-}cf \ \mathfrak{F} : \\ & op\text{-}cat \ \mathfrak{A} \mapsto_{C\alpha} cat\text{-}Set \ \alpha \end{aligned}$$

<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 $r \in_o \ \mathfrak{A}(\text{Obj})$

and $u : c \mapsto_{\mathfrak{B}} \ \mathfrak{F}(\text{ObjMap})(|r|)$

and *ntcf-ua-of* $\alpha \ \mathfrak{F} \ c \ r \ u$:

$$Hom_{O.C\alpha} \mathfrak{A}(r, -) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(c, -) \circ_{CF} \ \mathfrak{F} : \mathfrak{A} \mapsto_{C\alpha} cat\text{-}Set \ \alpha$$

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 $r \in_o \ \mathfrak{A}(\text{Obj})$

and $u : \mathfrak{F}(\text{ObjMap})(|r|) \mapsto_{\mathfrak{B}} \ c$

and *ntcf-ua-fo* $\alpha \ \mathfrak{F} \ c \ r \ u$:

$$\begin{aligned} & Hom_{O.C\alpha} \mathfrak{A}(-, r) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(-, c) \circ_{CF} op\text{-}cf \ \mathfrak{F} : \\ & op\text{-}cat \ \mathfrak{A} \mapsto_{C\alpha} cat\text{-}Set \ \alpha \end{aligned}$$

shows *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$

<proof>

lemma (in *is-functor*) *cf-universal-arrow-of-if-is-iso-ntcf*:

assumes $r \in_o \ \mathfrak{A}(\text{Obj})$

and $c \in_o \ \mathfrak{B}(\text{Obj})$

and φ :

$$Hom_{O.C\alpha} \mathfrak{A}(r, -) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(c, -) \circ_{CF} \ \mathfrak{F} : \mathfrak{A} \mapsto_{C\alpha} cat\text{-}Set \ \alpha$$

shows *universal-arrow-of* $\mathfrak{F} \ c \ r \ (\varphi(|NTMap|)(|r|)(|ArrVal|)(\mathfrak{A}(|CIId|)(|r|)))$

(**is** *<universal-arrow-of* $\mathfrak{F} \ c \ r \ ?u$)

<proof>

lemma (in *is-functor*) *cf-universal-arrow-fo-if-is-iso-ntcf*:

assumes $r \in_o \ \mathfrak{A}(\text{Obj})$

and $c \in_o \ \mathfrak{B}(\text{Obj})$

and φ :

$$Hom_{O.C\alpha} \mathfrak{A}(-, r) \mapsto_{CF.iso} Hom_{O.C\alpha} \mathfrak{B}(-, c) \circ_{CF} op\text{-}cf \ \mathfrak{F} :$$

$$op\text{-}cat \ \mathfrak{A} \mapsto_{C\alpha} cat\text{-}Set \ \alpha$$

shows *universal-arrow-fo* $\mathfrak{F} \ c \ r \ (\varphi(|NTMap|)(|r|)(|ArrVal|)(\mathfrak{A}(|CIId|)(|r|)))$

<proof>

3 Limits and colimits

3.1 Background

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

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

3.2 Limit and colimit

3.2.1 Definition and elementary properties

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

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

syntax *-is-cat-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $(\langle \langle - : / - <_{CF.lim} - : / - \mapsto_{C1} - \rangle \rangle [51, 51, 51, 51, 51] 51)$

syntax-consts *-is-cat-limit* \equiv *is-cat-limit*

translations $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \equiv$
 $CONST \textit{is-cat-limit} \alpha \mathfrak{J} \mathfrak{C} \mathfrak{F} r u$

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*: $\bigwedge u' r'. u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \implies$
 $\exists ! f'. f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = ntcf-const \mathfrak{J} \mathfrak{C} f' \cdot_{NTCF} u$

syntax *-is-cat-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $(\langle \langle - : / - >_{CF.colim} - : / - \mapsto_{C1} - \rangle \rangle [51, 51, 51, 51, 51] 51)$

syntax-consts *-is-cat-colimit* \equiv *is-cat-colimit*

translations $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \equiv$
 $CONST \textit{is-cat-colimit} \alpha \mathfrak{J} \mathfrak{C} \mathfrak{F} r u$

Rules.

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

mk-ide rf *is-cat-limit-def*[*unfolded is-cat-limit-axioms-def*]
 $|intro \textit{is-cat-limit}I|$
 $|dest \textit{is-cat-limit}D[dest]|$
 $|elim \textit{is-cat-limit}E[elim]|$

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 $u : \mathfrak{F}' >_{CF.colim} r' : \mathfrak{J}' \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

mk-ide rf *is-cat-colimit-def*[*unfolded is-cat-colimit-axioms-def*]
 $|intro \textit{is-cat-colimit}I|$
 $|dest \textit{is-cat-colimit}D[dest]|$
 $|elim \textit{is-cat-colimit}E[elim]|$

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

Limits, colimits and universal arrows.

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

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

Duality.

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

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

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

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

3.2.2 Universal property

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

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

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

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

lemma (in *is-cat-colimit*) *cat-colim-unique-cocone*:

assumes $u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
shows $\exists! f'. f' : r \mapsto_{\mathcal{C}} r' \wedge u' = \text{ntcf-const } \mathfrak{J} \mathcal{C} f' \cdot_{NTCF} u$
 ⟨proof⟩

lemma (in *is-cat-colimit*) *cat-colim-unique-cocone'*:

assumes $u' : \mathfrak{F} >_{CF.cocone} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
shows
 $\exists! f'. f' : r \mapsto_{\mathcal{C}} r' \wedge (\forall j \in \mathfrak{J}(\text{Obj}). u'(\text{NTMap})(j) = f' \circ_{A\mathcal{C}} u(\text{NTMap})(j))$
 ⟨proof⟩

lemma (in *is-cat-colimit*) *cat-colim-unique*:

assumes $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
shows $\exists! f'. f' : r \mapsto_{\mathcal{C}} r' \wedge u' = \text{ntcf-const } \mathfrak{J} \mathcal{C} f' \cdot_{NTCF} u$
 ⟨proof⟩

lemma (in *is-cat-colimit*) *cat-colim-unique'*:

assumes $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
shows
 $\exists! f'. f' : r \mapsto_{\mathcal{C}} r' \wedge (\forall j \in \mathfrak{J}(\text{Obj}). u'(\text{NTMap})(j) = f' \circ_{A\mathcal{C}} u(\text{NTMap})(j))$
 ⟨proof⟩

lemma *cat-lim-ex-is-iso-arr*:

assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$ **and** $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
obtains f **where** $f : r' \mapsto_{iso\mathcal{C}} r$ **and** $u' = u \cdot_{NTCF} \text{ntcf-const } \mathfrak{J} \mathcal{C} f$
 ⟨proof⟩

lemma *cat-lim-ex-is-iso-arr'*:

assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$ **and** $u' : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
obtains f **where** $f : r' \mapsto_{iso\mathcal{C}} r$
and $\wedge j. j \in \mathfrak{J}(\text{Obj}) \implies u'(\text{NTMap})(j) = u(\text{NTMap})(j) \circ_{A\mathcal{C}} f$
 ⟨proof⟩

lemma *cat-colim-ex-is-iso-arr*:

assumes $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
and $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
obtains f **where** $f : r \mapsto_{iso\mathcal{C}} r'$ **and** $u' = \text{ntcf-const } \mathfrak{J} \mathcal{C} f \cdot_{NTCF} u$
 ⟨proof⟩

lemma *cat-colim-ex-is-iso-arr'*:

assumes $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
and $u' : \mathfrak{F} >_{CF.colim} r' : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
obtains f **where** $f : r \mapsto_{iso\mathcal{C}} r'$
and $\wedge j. j \in \mathfrak{J}(\text{Obj}) \implies u'(\text{NTMap})(j) = f \circ_{A\mathcal{C}} u(\text{NTMap})(j)$
 ⟨proof⟩

3.2.3 Further properties

lemma (in *is-cat-limit*) *cat-lim-is-cat-limit-if-is-iso-arr*:

assumes $f : r' \mapsto_{iso\mathcal{C}} r$
shows $u \cdot_{NTCF} \text{ntcf-const } \mathfrak{J} \mathcal{C} f : r' <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{\mathcal{C}} C\alpha \mathcal{C}$
 ⟨proof⟩

lemma (in *is-cat-colimit*) *cat-colim-is-cat-colimit-if-is-iso-arr*:

assumes $f : r \mapsto_{iso\mathcal{C}} r'$

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

lemma $ntcf\text{-}cf\text{-}comp\text{-}is\text{-}cat\text{-}limit\text{-}if\text{-}is\text{-}iso\text{-}functor$:

assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto_{C.iso\alpha} \mathfrak{B}$

shows $u \circ_{NTCF-CF} \mathfrak{G} : r <_{CF.lim} \mathfrak{F} \circ_{CF} \mathfrak{G} : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{C}$

⟨proof⟩

lemma $ntcf\text{-}cf\text{-}comp\text{-}is\text{-}cat\text{-}limit\text{-}if\text{-}is\text{-}iso\text{-}functor'$ [cat-lim-cs-intros]:

assumes $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$

and $\mathfrak{G} : \mathfrak{A} \mapsto_{C.iso\alpha} \mathfrak{B}$

and $\mathfrak{A}' = \mathfrak{F} \circ_{CF} \mathfrak{G}$

shows $u \circ_{NTCF-CF} \mathfrak{G} : r <_{CF.lim} \mathfrak{A}' : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{C}$

⟨proof⟩

3.3 Small limit and small colimit

3.3.1 Definition and elementary properties

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

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

assumes $tm\text{-}cat\text{-}lim\text{-}ua\text{-}fo$:

$\wedge u' r'. u' : r' <_{CF.cone} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \implies$

$\exists ! f'. f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \cdot_{NTCF} ntcf\text{-}const \mathfrak{J} \mathfrak{C} f'$

syntax $-is\text{-}tm\text{-}cat\text{-}limit :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle (- : / - <_{CF.tm.lim} - : / - \mapsto_{C.tm1} -) \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts $-is\text{-}tm\text{-}cat\text{-}limit \hat{=} is\text{-}tm\text{-}cat\text{-}limit$

translations $u : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \hat{=}$

$CONST is\text{-}tm\text{-}cat\text{-}limit \alpha \mathfrak{J} \mathfrak{C} \mathfrak{F} r u$

locale $is\text{-}tm\text{-}cat\text{-}colimit = is\text{-}tm\text{-}cat\text{-}cocone \alpha r \mathfrak{J} \mathfrak{C} \mathfrak{F} u$ **for** $\alpha \mathfrak{J} \mathfrak{C} \mathfrak{F} r u +$

assumes $tm\text{-}cat\text{-}colim\text{-}ua\text{-}of$:

$\wedge u' r'. u' : \mathfrak{F} >_{CF.cococone} r' : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \implies$

$\exists ! f'. f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = ntcf\text{-}const \mathfrak{J} \mathfrak{C} f' \cdot_{NTCF} u$

syntax $-is\text{-}tm\text{-}cat\text{-}colimit :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle (- : / - >_{CF.tm.colim} - : / - \mapsto_{C.tm1} -) \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts $-is\text{-}tm\text{-}cat\text{-}colimit \hat{=} is\text{-}tm\text{-}cat\text{-}colimit$

translations $u : \mathfrak{F} >_{CF.tm.colim} r : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \hat{=}$

$CONST is\text{-}tm\text{-}cat\text{-}colimit \alpha \mathfrak{J} \mathfrak{C} \mathfrak{F} r u$

Rules.

lemma (in $is\text{-}tm\text{-}cat\text{-}limit$) $is\text{-}tm\text{-}cat\text{-}limit\text{-}axioms'$ [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 $u : r' <_{CF.tm.lim} \mathfrak{F}' : \mathfrak{J}' \mapsto_{C.tm\alpha'} \mathfrak{C}'$

⟨proof⟩

mk-ide rf $is\text{-}tm\text{-}cat\text{-}limit\text{-}def$ [unfolded $is\text{-}tm\text{-}cat\text{-}limit\text{-}axioms\text{-}def$]

|intro $is\text{-}tm\text{-}cat\text{-}limitI$ |

|dest $is\text{-}tm\text{-}cat\text{-}limitD$ [dest]|

|elim $is\text{-}tm\text{-}cat\text{-}limitE$ [elim]|

lemmas [cat-lim-cs-intros] = $is\text{-}tm\text{-}cat\text{-}limitD(1)$

lemma (in $is\text{-}tm\text{-}cat\text{-}colimit$) $is\text{-}tm\text{-}cat\text{-}colimit\text{-}axioms'$ [cat-lim-cs-intros]:

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

mk-ide rf *is-tm-cat-colimit-def*[*unfolded is-tm-cat-colimit-axioms-def*]
 $|intro\ is\ tm\ cat\ colimit\ I|$
 $|dest\ is\ tm\ cat\ colimit\ D[dest]|$
 $|elim\ is\ tm\ cat\ colimit\ E[elim]|$

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

lemma *is-tm-cat-limitI'*:

assumes $u : r <_{CF.tm.cone} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge u' r'. u' : r' <_{CF.tm.cone} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \implies$
 $\exists ! f'. f' : r' \mapsto_{\mathfrak{C}} r \wedge u' = u \cdot_{NTCF} ntcf\text{-const } \mathfrak{J} \mathfrak{C} f'$
shows $u : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma *is-tm-cat-colimitI'*:

assumes $u : \mathfrak{F} >_{CF.tm.cocone} r : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
and $\bigwedge u' r'. u' : \mathfrak{F} >_{CF.tm.cocone} r' : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \implies$
 $\exists ! f'. f' : r \mapsto_{\mathfrak{C}} r' \wedge u' = ntcf\text{-const } \mathfrak{J} \mathfrak{C} f' \cdot_{NTCF} u$
shows $u : \mathfrak{F} >_{CF.tm.colim} r : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
 $\langle proof \rangle$

Elementary properties.

sublocale *is-tm-cat-limit* \subseteq *is-cat-limit*
 $\langle proof \rangle$

sublocale *is-tm-cat-colimit* \subseteq *is-cat-colimit*
 $\langle proof \rangle$

lemma (in *is-cat-limit*) *cat-lim-is-tm-cat-limit*:

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
shows $u : r <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma (in *is-cat-colimit*) *cat-colim-is-tm-cat-colimit*:

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
shows $u : \mathfrak{F} >_{CF.tm.colim} r : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
 $\langle proof \rangle$

Limits, colimits and universal arrows.

lemma (in *is-tm-cat-limit*) *tm-cat-lim-is-universal-arrow-fo*:

universal-arrow-fo ($\Delta_{CF.tm} \alpha \mathfrak{J} \mathfrak{C}$) (*cf-map* \mathfrak{F}) r (*ntcf-arrow* u)
 $\langle proof \rangle$

lemma (in *is-tm-cat-cone*) *tm-cat-cone-is-tm-cat-limit*:

assumes *universal-arrow-fo* ($\Delta_{CF.tm} \alpha \mathfrak{J} \mathfrak{C}$) (*cf-map* \mathfrak{F}) c (*ntcf-arrow* \mathfrak{N})
shows $\mathfrak{N} : c <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma (in *is-tm-cat-colimit*) *tm-cat-colim-is-universal-arrow-of*:

universal-arrow-of ($\Delta_{CF.tm} \alpha \mathfrak{J} \mathfrak{C}$) (*cf-map* \mathfrak{F}) r (*ntcf-arrow* u)
 $\langle proof \rangle$

lemma (in *is-tm-cat-cocone*) *tm-cat-cocone-is-tm-cat-colimit*:

assumes *universal-arrow-of* ($\Delta_{CF.tm} \alpha \mathfrak{J} \mathfrak{C}$) (*cf-map* \mathfrak{F}) c (*ntcf-arrow* \mathfrak{N})

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

Duality.

lemma (in is-tm-cat-limit) is-tm-cat-colimit-op:

$op-ntcf\ u : op-cf\ \mathfrak{F} >_{CF.tm.colim} r : op-cat\ \mathfrak{J} \mapsto_{C.tm\alpha} op-cat\ \mathcal{C}$
 ⟨proof⟩

lemma (in is-tm-cat-limit) is-tm-cat-colimit-op'[cat-op-intros]:

assumes $\mathfrak{F}' = op-cf\ \mathfrak{F}$ **and** $\mathfrak{J}' = op-cat\ \mathfrak{J}$ **and** $\mathcal{C}' = op-cat\ \mathcal{C}$
shows $op-ntcf\ u : \mathfrak{F}' >_{CF.tm.colim} r : \mathfrak{J}' \mapsto_{C.tm\alpha} \mathcal{C}'$
 ⟨proof⟩

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

lemma (in is-tm-cat-colimit) is-tm-cat-limit-op:

$op-ntcf\ u : r <_{CF.tm.lim} op-cf\ \mathfrak{F} : op-cat\ \mathfrak{J} \mapsto_{C.tm\alpha} op-cat\ \mathcal{C}$
 ⟨proof⟩

lemma (in is-tm-cat-colimit) is-tm-cat-colimit-op'[cat-op-intros]:

assumes $\mathfrak{F}' = op-cf\ \mathfrak{F}$ **and** $\mathfrak{J}' = op-cat\ \mathfrak{J}$ **and** $\mathcal{C}' = op-cat\ \mathcal{C}$
shows $op-ntcf\ u : r <_{CF.tm.lim} \mathfrak{F}' : \mathfrak{J}' \mapsto_{C.tm\alpha} \mathcal{C}'$
 ⟨proof⟩

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

3.3.2 Further properties

lemma (in is-tm-cat-limit) tm-cat-lim-is-tm-cat-limit-if-iso-arr:

assumes $f : r' \mapsto_{iso} \mathcal{C} r$
shows $u \cdot_{NTCF} ntcf-const\ \mathfrak{J}\ \mathcal{C}\ f : r' <_{CF.tm.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathcal{C}$
 ⟨proof⟩

lemma (in is-tm-cat-colimit) tm-cat-colim-is-tm-cat-colimit-if-iso-arr:

assumes $f : r \mapsto_{iso} \mathcal{C} r'$
shows $ntcf-const\ \mathfrak{J}\ \mathcal{C}\ f \cdot_{NTCF} u : \mathfrak{F} >_{CF.tm.colim} r' : \mathfrak{J} \mapsto_{C.tm\alpha} \mathcal{C}$
 ⟨proof⟩

3.4 Finite limit and finite colimit

locale is-cat-finite-limit =

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

syntax -is-cat-finite-limit :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

(⟨(- :/ - <_{CF.lim.fin} - :/ - \mapsto_{C1} -)⟩ [51, 51, 51, 51, 51] 51)

syntax-consts -is-cat-finite-limit \equiv is-cat-finite-limit

translations $u : r <_{CF.lim.fin} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathcal{C} \equiv$

CONST is-cat-finite-limit $\alpha\ \mathfrak{J}\ \mathcal{C}\ \mathfrak{F}\ r\ u$

locale is-cat-finite-colimit =

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

syntax -is-cat-finite-colimit :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

(⟨(- :/ - >_{CF.colim.fin} - :/ - \mapsto_{C1} -)⟩ [51, 51, 51, 51, 51] 51)

syntax-consts -is-cat-finite-colimit \equiv is-cat-finite-colimit

translations $u : \mathfrak{F} >_{CF.colim.fin} r : \mathfrak{J} \mapsto_{C\alpha} \mathcal{C} \equiv$

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

Rules.

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

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

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

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

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

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

Duality.

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

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

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

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

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

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

Elementary properties.

sublocale *is-cat-finite-limit* \subseteq *is-tm-cat-limit*
 ⟨*proof*⟩

sublocale *is-cat-finite-colimit* \subseteq *is-tm-cat-colimit*
 ⟨*proof*⟩

3.5 Creation of limits

See Chapter V-1 in [9].

definition *cf-creates-limits* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *cf-creates-limits* $\alpha \mathfrak{G} \mathfrak{F} =$
 ($\forall \tau b.$
 $\tau : b <_{CF.lim} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod) \longrightarrow$
 ($\mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod) \longrightarrow$
 ($\exists ! \sigma a. \exists \sigma a. \sigma a = \langle \sigma, a \rangle \wedge$
 $\sigma : a <_{CF.cone} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod) \wedge$
 $\tau = \mathfrak{G} \circ_{CF-NTCF} \sigma \wedge$
 $b = \mathfrak{G}(ObjMap)(a)$
) \wedge
 ($\forall \sigma a.$
 $\sigma : a <_{CF.cone} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod) \longrightarrow$
 $\tau = \mathfrak{G} \circ_{CF-NTCF} \sigma \longrightarrow$
 $b = \mathfrak{G}(ObjMap)(a) \longrightarrow$
 $\sigma : a <_{CF.lim} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod)$
)
)
)

Rules.

context

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

begin

interpretation \mathfrak{F} : *is-functor* $\alpha \mathfrak{J} \mathfrak{A} \mathfrak{F}$ *{proof}*

interpretation \mathfrak{G} : *is-functor* $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{G}$ *{proof}*

mk-ide rf *cf-creates-limits-def*[

where $\alpha = \alpha$ **and** $\mathfrak{F} = \mathfrak{F}$ **and** $\mathfrak{G} = \mathfrak{G}$, *unfolded cat-cs-simps*

] *intro cf-creates-limitsI*
dest cf-creates-limitsD'
elim cf-creates-limitsE'

end

lemmas *cf-creates-limitsD[dest!]* = *cf-creates-limitsD'[rotated 2]*

and *cf-creates-limitsE[elim!]* = *cf-creates-limitsE'[rotated 2]*

lemma *cf-creates-limitsE''*:

assumes *cf-creates-limits* $\alpha \mathfrak{G} \mathfrak{F}$
and $\tau : b <_{CF.lim} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G} : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$
obtains σr **where** $\sigma : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{A}$
and $\tau = \mathfrak{G} \circ_{CF-NTCF} \sigma$
and $b = \mathfrak{G}(ObjMap)(r)$

{proof}

3.6 Preservation of limits and colimits

3.6.1 Definitions and elementary properties

See Chapter V-4 in [9].

definition *cf-preserves-limits* :: $V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *cf-preserves-limits* $\alpha \ \mathfrak{G} \ \mathfrak{F} =$

```
(
   $\forall \sigma \ a.$ 
   $\sigma : a <_{CF.lim} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{F}(HomCod) \longrightarrow$ 
   $\mathfrak{G} \circ_{CF-NTCF} \sigma : \mathfrak{G}(ObjMap)(a) <_{CF.lim} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod)$ 
)
```

definition *cf-preserves-colimits* :: $V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *cf-preserves-colimits* $\alpha \ \mathfrak{G} \ \mathfrak{F} =$

```
(
   $\forall \sigma \ a.$ 
   $\sigma : \mathfrak{F} >_{CF.colim} a : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{F}(HomCod) \longrightarrow$ 
   $\mathfrak{G} \circ_{CF-NTCF} \sigma : \mathfrak{G} \circ_{CF} \mathfrak{F} >_{CF.colim} \mathfrak{G}(ObjMap)(a) : \mathfrak{F}(HomDom) \mapsto_{C\alpha} \mathfrak{G}(HomCod)$ 
)
```

Rules.

context

fixes $\alpha \ \mathfrak{J} \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{G} \ \mathfrak{F}$

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{A}$

and $\mathfrak{G} : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$

begin

interpretation \mathfrak{F} : *is-functor* $\alpha \ \mathfrak{J} \ \mathfrak{A} \ \mathfrak{F}$ *<proof>*

interpretation \mathfrak{G} : *is-functor* $\alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{G}$ *<proof>*

mk-ide rf *cf-preserves-limits-def*[

where $\alpha=\alpha$ **and** $\mathfrak{F}=\mathfrak{F}$ **and** $\mathfrak{G}=\mathfrak{G}$, *unfolded cat-cs-simps*

]

|*intro cf-preserves-limitsI*

|*dest cf-preserves-limitsD'*

|*elim cf-preserves-limitsE'*

mk-ide rf *cf-preserves-colimits-def*[

where $\alpha=\alpha$ **and** $\mathfrak{F}=\mathfrak{F}$ **and** $\mathfrak{G}=\mathfrak{G}$, *unfolded cat-cs-simps*

]

|*intro cf-preserves-colimitsI*

|*dest cf-preserves-colimitsD'*

|*elim cf-preserves-colimitsE'*

end

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

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

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

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

Duality.

lemma *cf-preserves-colimits-op*[*cat-op-simps*]:

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$

shows

cf-preserves-colimits α (*op-cf* \mathfrak{G}) (*op-cf* \mathfrak{F}) \longleftrightarrow

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

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

3.6.2 Further properties

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

3.7 Continuous and cocontinuous functor

3.7.1 Definition and elementary properties

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

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

Rules.

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

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

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

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

end

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

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

Duality.

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

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

3.7.2 Category isomorphisms are continuous and cocontinuous

lemma (**in** *is-iso-functor*) *iso-cf-is-cf-continuous*: *is-cf-continuous* α \mathfrak{F}
<proof>

lemma (**in** *is-iso-functor*) *iso-cf-is-cf-cocontinuous*: *is-cf-cocontinuous* α \mathfrak{F}
<proof>

3.8 Tiny-continuous and tiny-cocontinuous functor

3.8.1 Definition and elementary properties

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

Rules.

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

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

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

end

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

Elementary properties.

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

4 Initial and terminal objects as limits and colimits

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

4.1.1 Definition and elementary properties

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

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

syntax *-is-cat-obj-empty-terminal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$
 \langle \langle - :/ - $<_{CF.1}$ 0_{CF} :/ $0_C \mapsto_{C^1}$ - \rangle [51, 51] 51

syntax-consts *-is-cat-obj-empty-terminal* \equiv *is-cat-obj-empty-terminal*

translations $\mathfrak{Z} : z <_{CF.1} 0_{CF} : 0_C \mapsto_{C^\alpha} \mathfrak{C} \equiv$
CONST is-cat-obj-empty-terminal α \mathfrak{C} z \mathfrak{Z}

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

syntax *-is-cat-obj-empty-initial* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$
 \langle \langle - :/ $0_{CF} >_{CF.0}$ - :/ $0_C \mapsto_{C^1}$ - \rangle [51, 51] 51

syntax-consts *-is-cat-obj-empty-initial* \equiv *is-cat-obj-empty-initial*

translations $\mathfrak{Z} : 0_{CF} >_{CF.0} z : 0_C \mapsto_{C^\alpha} \mathfrak{C} \equiv$
CONST is-cat-obj-empty-initial α \mathfrak{C} z \mathfrak{Z}

Rules.

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

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

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

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

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

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

Duality.

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

²<https://ncatlab.org/nlab/show/initial+object>

³<https://ncatlab.org/nlab/show/terminal+object>

$\langle \text{proof} \rangle$

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

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

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

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

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

Elementary properties.

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

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

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

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

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

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

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

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

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

4.2 Initial cone and terminal cocone

4.2.1 Definitions and elementary properties

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

where $ntcf\text{-initial } \mathfrak{C} z =$
 $[$
 $(\lambda b \in_o \mathfrak{C}(\text{Obj}). \text{THE } f. f : z \mapsto_{\mathfrak{C}} b),$
 $cf\text{-const } \mathfrak{C} \mathfrak{C} z,$
 $cf\text{-id } \mathfrak{C},$
 $\mathfrak{C},$
 \mathfrak{C}
 $].$

definition $ntcf\text{-terminal} :: V \Rightarrow V \Rightarrow V$

where $ntcf\text{-terminal } \mathfrak{C} z =$
 $[$
 $(\lambda b \in_o \mathfrak{C}(\text{Obj}). \text{THE } f. f : b \mapsto_{\mathfrak{C}} z),$
 $cf\text{-id } \mathfrak{C},$
 $cf\text{-const } \mathfrak{C} \mathfrak{C} z,$
 $\mathfrak{C},$
 \mathfrak{C}
 $].$

Components.

lemma $ntcf\text{-initial-components}$:

shows $ntcf\text{-initial } \mathfrak{C} z(\text{NTMap}) = (\lambda c \in_o \mathfrak{C}(\text{Obj}). \text{THE } f. f : z \mapsto_{\mathfrak{C}} c)$
and $ntcf\text{-initial } \mathfrak{C} z(\text{NTDom}) = cf\text{-const } \mathfrak{C} \mathfrak{C} z$
and $ntcf\text{-initial } \mathfrak{C} z(\text{NTCod}) = cf\text{-id } \mathfrak{C}$
and $ntcf\text{-initial } \mathfrak{C} z(\text{NTDGDom}) = \mathfrak{C}$
and $ntcf\text{-initial } \mathfrak{C} z(\text{NTDGCod}) = \mathfrak{C}$
 $\langle \text{proof} \rangle$

lemmas $[cat\text{-lim-cs-simps}] = ntcf\text{-initial-components}(2-5)$

lemma $ntcf\text{-terminal-components}$:

shows $ntcf\text{-terminal } \mathfrak{C} z(\text{NTMap}) = (\lambda c \in_o \mathfrak{C}(\text{Obj}). \text{THE } f. f : c \mapsto_{\mathfrak{C}} z)$
and $ntcf\text{-terminal } \mathfrak{C} z(\text{NTDom}) = cf\text{-id } \mathfrak{C}$
and $ntcf\text{-terminal } \mathfrak{C} z(\text{NTCod}) = cf\text{-const } \mathfrak{C} \mathfrak{C} z$
and $ntcf\text{-terminal } \mathfrak{C} z(\text{NTDGDom}) = \mathfrak{C}$
and $ntcf\text{-terminal } \mathfrak{C} z(\text{NTDGCod}) = \mathfrak{C}$
 $\langle \text{proof} \rangle$

lemmas $[cat\text{-lim-cs-simps}] = ntcf\text{-terminal-components}(2-5)$

Duality.

lemma $ntcf\text{-initial-op}[cat\text{-op-simps}]$:

$op\text{-ntcf } (ntcf\text{-initial } \mathfrak{C} z) = ntcf\text{-terminal } (op\text{-cat } \mathfrak{C}) z$
 $\langle \text{proof} \rangle$

lemma $ntcf\text{-cone-terminal-op}[cat\text{-op-simps}]$:

$op\text{-ntcf } (ntcf\text{-terminal } \mathfrak{C} z) = ntcf\text{-initial } (op\text{-cat } \mathfrak{C}) z$
 $\langle \text{proof} \rangle$

4.2.2 Natural transformation map

mk-VLambda $ntcf\text{-initial-components}(1)$

$[vsu\ ntcf\text{-initial-vsuv}[cat\text{-lim-cs-intros]]$
 $[vdomain\ ntcf\text{-initial-vdomain}[cat\text{-lim-cs-simps]]$
 $[app\ ntcf\text{-initial-app}]$

mk-VLambda $ntcf\text{-terminal-components}(1)$

$[vsu\ ntcf\text{-terminal-vsuv}[cat\text{-lim-cs-intros]]$

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

lemma (in category)

assumes *obj-initial* $\mathfrak{C} z$ and $c \in_{\circ} \mathfrak{C}(\text{Obj})$

shows *ntcf-initial-NTMap-app-is-arr*:

$\text{ntcf-initial } \mathfrak{C} z(\text{NTMap})(c) : z \mapsto_{\mathfrak{C}} c$

and *ntcf-initial-NTMap-app-unique*:

$\wedge f'. f' : z \mapsto_{\mathfrak{C}} c \implies f' = \text{ntcf-initial } \mathfrak{C} z(\text{NTMap})(c)$

<proof>

lemma (in category) *ntcf-initial-NTMap-app-is-arr'*[cat-lim-cs-intros]:

assumes *obj-initial* $\mathfrak{C} z$

and $c \in_{\circ} \mathfrak{C}(\text{Obj})$

and $\mathfrak{C}' = \mathfrak{C}$

and $z' = z$

and $c' = c$

shows *ntcf-initial* $\mathfrak{C} z(\text{NTMap})(c) : z' \mapsto_{\mathfrak{C}'} c'$

<proof>

lemma (in category)

assumes *obj-terminal* $\mathfrak{C} z$ and $c \in_{\circ} \mathfrak{C}(\text{Obj})$

shows *ntcf-terminal-NTMap-app-is-arr*:

$\text{ntcf-terminal } \mathfrak{C} z(\text{NTMap})(c) : c \mapsto_{\mathfrak{C}} z$

and *ntcf-terminal-NTMap-app-unique*:

$\wedge f'. f' : c \mapsto_{\mathfrak{C}} z \implies f' = \text{ntcf-terminal } \mathfrak{C} z(\text{NTMap})(c)$

<proof>

lemma (in category) *ntcf-terminal-NTMap-app-is-arr'*[cat-lim-cs-intros]:

assumes *obj-terminal* $\mathfrak{C} z$

and $c \in_{\circ} \mathfrak{C}(\text{Obj})$

and $\mathfrak{C}' = \mathfrak{C}$

and $z' = z$

and $c' = c$

shows *ntcf-terminal* $\mathfrak{C} z(\text{NTMap})(c) : c' \mapsto_{\mathfrak{C}'} z'$

<proof>

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

4.3.1 Definition and elementary properties

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

locale *is-cat-obj-id-initial* = *is-cat-limit* $\alpha \mathfrak{C} \mathfrak{C} \langle \text{cf-id } \mathfrak{C} \rangle z \mathfrak{J}$ for $\alpha \mathfrak{C} z \mathfrak{J}$

syntax *-is-cat-obj-id-initial* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - : / - <_{CF.0} \text{id}_C : / \mapsto_{C1} - \rangle [51, 51, 51] 51 \rangle$

syntax-consts *-is-cat-obj-id-initial* \equiv *is-cat-obj-id-initial*

translations $\mathfrak{J} : z <_{CF.0} \text{id}_C : \mapsto_{C\alpha} \mathfrak{C} \equiv$

CONST is-cat-obj-id-initial $\alpha \mathfrak{C} z \mathfrak{J}$

locale *is-cat-obj-id-terminal* = *is-cat-colimit* $\alpha \mathfrak{C} \mathfrak{C} \langle \text{cf-id } \mathfrak{C} \rangle z \mathfrak{J}$ for $\alpha \mathfrak{C} z \mathfrak{J}$

syntax *-is-cat-obj-id-terminal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - : / \text{id}_C >_{CF.1} - : / \mapsto_{C1} - \rangle [51, 51, 51] 51 \rangle$

syntax-consts *-is-cat-obj-id-terminal* \equiv *is-cat-obj-id-terminal*

⁴<https://ncatlab.org/nlab/show/initial+object>

⁵<https://ncatlab.org/nlab/show/terminal+object>

translations $\mathfrak{J} : id_C \succ_{CF.1} z : \mapsto_{C\alpha} \mathfrak{C} \rightleftharpoons$
 $CONST$ *is-cat-obj-id-terminal* $\alpha \mathfrak{C} z \mathfrak{J}$

Rules.

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

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

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

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

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

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

Duality.

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

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

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

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

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

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

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

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

<proof>

lemma (in *category*) *cat-obj-terminal-is-cat-obj-id-terminal*:

assumes *obj-terminal* $\mathfrak{C} z$

shows *ntcf-terminal* $\mathfrak{C} z : id_C >_{CF.1} z : \mapsto_{C\alpha} \mathfrak{C}$

<proof>

lemma *cat-cone-CId-obj-initial*:

assumes $\exists : z <_{CF.cone} cf-id \mathfrak{C} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{C}$ and $\exists (NTMap)(z) = \mathfrak{C}(CId)(z)$

shows *obj-initial* $\mathfrak{C} z$

<proof>

lemma *cat-cocone-CId-obj-terminal*:

assumes $\exists : cf-id \mathfrak{C} >_{CF.cocone} z : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{C}$ and $\exists (NTMap)(z) = \mathfrak{C}(CId)(z)$

shows *obj-terminal* $\mathfrak{C} z$

<proof>

lemma (in *is-cat-obj-id-initial*) *cat-oi-obj-initial*: *obj-initial* $\mathfrak{C} z$

<proof>

lemma (in *is-cat-obj-id-terminal*) *cat-oi-obj-terminal*: *obj-terminal* $\mathfrak{C} z$

<proof>

lemma (in *category*) *cat-is-cat-obj-id-initial-obj-initial-iff*:

(*ntcf-initial* $\mathfrak{C} z : z <_{CF.0} id_C : \mapsto_{C\alpha} \mathfrak{C}$) \longleftrightarrow *obj-initial* $\mathfrak{C} z$

<proof>

lemma (in *category*) *cat-is-cat-obj-id-terminal-obj-terminal-iff*:

(*ntcf-terminal* $\mathfrak{C} z : id_C >_{CF.1} z : \mapsto_{C\alpha} \mathfrak{C}$) \longleftrightarrow *obj-terminal* $\mathfrak{C} z$

<proof>

5 Products and coproducts as limits and colimits

5.1 Product and coproduct

5.1.1 Definition and elementary properties

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

locale *is-cat-obj-prod* =
is-cat-limit $\alpha \langle :_C I \rangle \mathfrak{C} \langle \cdot \rightarrow : I A \mathfrak{C} \rangle P \pi + \text{cf-discrete } \alpha I A \mathfrak{C}$
for $\alpha I A \mathfrak{C} P \pi$

syntax *-is-cat-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $(\langle (- : / - <_{CF.\Pi} - : / - \mapsto_{C1} -) \rangle [51, 51, 51, 51, 51] 51)$
syntax-consts *-is-cat-obj-prod* $\hat{=} is-cat-obj-prod$
translations $\pi : P <_{CF.\Pi} A : I \mapsto_{C\alpha} \mathfrak{C} \hat{=} \text{CONST } is-cat-obj-prod \alpha I A \mathfrak{C} P \pi$

locale *is-cat-obj-coproduct* =
is-cat-colimit $\alpha \langle :_C I \rangle \mathfrak{C} \langle \cdot \rightarrow : I A \mathfrak{C} \rangle U \pi + \text{cf-discrete } \alpha I A \mathfrak{C}$
for $\alpha I A \mathfrak{C} U \pi$

syntax *-is-cat-obj-coproduct* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $(\langle (- : / - >_{CF.\Pi} - : / - \mapsto_{C1} -) \rangle [51, 51, 51, 51, 51] 51)$
syntax-consts *-is-cat-obj-coproduct* $\hat{=} is-cat-obj-coproduct$
translations $\pi : A >_{CF.\Pi} U : I \mapsto_{C\alpha} \mathfrak{C} \hat{=} \text{CONST } is-cat-obj-coproduct \alpha I A \mathfrak{C} U \pi$

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 $\pi : P' <_{CF.\Pi} A' : I' \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

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-coproduct*) *is-cat-obj-coproduct-axioms'*[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\pi : A' >_{CF.\Pi} U' : I' \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

mk-ide rf *is-cat-obj-coproduct-def*
 $|intro\ is-cat-obj-coproductI|$
 $|dest\ is-cat-obj-coproductD[dest]|$
 $|elim\ is-cat-obj-coproductE[elim]|$

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

Duality.

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

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

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

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

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

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

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

5.1.2 Universal property

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

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

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

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

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

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

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

lemma *cat-obj-prod-ex-is-iso-arr'*:

assumes $\pi : P <_{CF.\Pi} A : I \mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : P' <_{CF.\Pi} A : I \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : P' \mapsto_{iso\mathfrak{C}} P$
and $\bigwedge j. j \in_o I \implies \pi'(NTMap)(j) = \pi(NTMap)(j) \circ_{A\mathfrak{C}} f$

<proof>

lemma *cat-obj-coprod-ex-is-iso-arr*:

assumes $\pi : A >_{CF.\Pi} U : I \mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : A >_{CF.\Pi} U' : I \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : U \mapsto_{iso\mathfrak{C}} U'$ **and** $\pi' = ntcf-const (:_C I) \mathfrak{C} f \cdot_{NTCF} \pi$

<proof>

lemma *cat-obj-coprod-ex-is-iso-arr'*:

assumes $\pi : A >_{CF.\Pi} U : I \mapsto_{C\alpha} \mathfrak{C}$ **and** $\pi' : A >_{CF.\Pi} U' : I \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : U \mapsto_{iso\mathfrak{C}} U'$
and $\bigwedge j. j \in_o I \implies \pi'(NTMap)(j) = f \circ_{A\mathfrak{C}} \pi(NTMap)(j)$

<proof>

5.2 Small product and small coproduct

5.2.1 Definition and elementary properties

locale *is-tm-cat-obj-prod* =

is-cat-limit $\alpha \langle :_C I \rangle \mathfrak{C} \langle : \rightarrow : I A \mathfrak{C} \rangle P \pi + tm-cf-discrete \alpha I A \mathfrak{C}$
for $\alpha I A \mathfrak{C} P \pi$

syntax *-is-tm-cat-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle - : / - <_{CF.tm.\Pi} - : / - \mapsto_{C.tm^1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-tm-cat-obj-prod* $\hat{=} is-tm-cat-obj-prod$

translations $\pi : P <_{CF.tm.\Pi} A : I \mapsto_{C.tm\alpha} \mathfrak{C} \hat{=}$

CONST is-tm-cat-obj-prod $\alpha I A \mathfrak{C} P \pi$

locale *is-tm-cat-obj-coprod* =

is-cat-colimit $\alpha \langle :_C I \rangle \mathfrak{C} \langle : \rightarrow : I A \mathfrak{C} \rangle U \pi + tm-cf-discrete \alpha I A \mathfrak{C}$
for $\alpha I A \mathfrak{C} U \pi$

syntax *-is-tm-cat-obj-coprod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle - : / - >_{CF.tm.\Pi} - : / - \mapsto_{C.tm^1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-tm-cat-obj-coprod* $\hat{=} is-tm-cat-obj-coprod$

translations $\pi : A >_{CF.tm.\Pi} U : I \mapsto_{C.tm\alpha} \mathfrak{C} \hat{=}$

CONST is-tm-cat-obj-coprod $\alpha I A \mathfrak{C} U \pi$

Rules.

lemma (**in** *is-tm-cat-obj-prod*) *is-tm-cat-obj-prod-axioms*^[*cat-lim-cs-intros*]:

assumes $\alpha' = \alpha$ **and** $P' = P$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$

shows $\pi : P' <_{CF.tm.\Pi} A' : I' \mapsto_{C.tm\alpha'} \mathfrak{C}'$

<proof>

mk-ide **rf** *is-tm-cat-obj-prod-def*

[intro is-tm-cat-obj-prodI]

[dest is-tm-cat-obj-prodD[dest]]

[elim is-tm-cat-obj-prodE[elim]]

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

lemma (**in** *is-tm-cat-obj-coprod*)

is-tm-cat-obj-coprod-axioms^[*cat-lim-cs-intros*]:

assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$

shows $\pi : A' >_{CF.tm.\Pi} U' : I' \mapsto_{C.tm\alpha'} \mathfrak{C}'$

$\langle proof \rangle$

mk-ide rf *is-tm-cat-obj-coprod-def*

$|intro\ is\ tm\ cat\ obj\ coprod\ I|$
 $|dest\ is\ tm\ cat\ obj\ coprod\ D[dest]|$
 $|elim\ is\ tm\ cat\ obj\ coprod\ E[elim]|$

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

Elementary properties.

sublocale *is-tm-cat-obj-prod* \subseteq *is-cat-obj-prod*

$\langle proof \rangle$

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

sublocale *is-tm-cat-obj-coprod* \subseteq *is-cat-obj-coprod*

$\langle proof \rangle$

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

sublocale *is-tm-cat-obj-prod* \subseteq *is-tm-cat-limit* $\alpha \langle :_C I \rangle \mathfrak{C} \langle :- \rangle : I A \mathfrak{C} \rangle P \pi$

$\langle proof \rangle$

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

sublocale *is-tm-cat-obj-coprod* \subseteq *is-tm-cat-colimit* $\alpha \langle :_C I \rangle \mathfrak{C} \langle :- \rangle : I A \mathfrak{C} \rangle U \pi$

$\langle proof \rangle$

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

Duality.

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

op-ntcf $\pi : A >_{CF.tm.\Pi} P : I \mapsto \mapsto_{C.tm\alpha} op\text{-}cat\ \mathfrak{C}$

$\langle proof \rangle$

lemma (in *is-tm-cat-obj-prod*) *is-tm-cat-obj-coprod-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$

shows *op-ntcf* $\pi : A >_{CF.tm.\Pi} P : I \mapsto \mapsto_{C.tm\alpha} \mathfrak{C}'$

$\langle proof \rangle$

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

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

op-ntcf $\pi : U <_{CF.tm.\Pi} A : I \mapsto \mapsto_{C.tm\alpha} op\text{-}cat\ \mathfrak{C}$

$\langle proof \rangle$

lemma (in *is-tm-cat-obj-coprod*) *is-tm-cat-obj-prod-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$

shows *op-ntcf* $\pi : U <_{CF.tm.\Pi} A : I \mapsto \mapsto_{C.tm\alpha} \mathfrak{C}'$

$\langle proof \rangle$

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

5.3 Finite product and finite coproduct

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

syntax *-is-cat-finite-obj-prod* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $\langle \langle - : / - <_{CF.\Pi.f\in} - : / - \mapsto_{C^1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$
syntax-consts *-is-cat-finite-obj-prod* \equiv *is-cat-finite-obj-prod*
translations $\pi : P <_{CF.\Pi.f\in} A : I \mapsto_{C\alpha} \mathfrak{C} \equiv$
CONST is-cat-finite-obj-prod α *I A* \mathfrak{C} *P* π

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

syntax *-is-cat-finite-obj-coproduct* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 $\langle \langle - : / - >_{CF.\Pi.f\in} - : / - \mapsto_{C^1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$
syntax-consts *-is-cat-finite-obj-coproduct* \equiv *is-cat-finite-obj-coproduct*
translations $\pi : A >_{CF.\Pi.f\in} U : I \mapsto_{C\alpha} \mathfrak{C} \equiv$
CONST is-cat-finite-obj-coproduct α *I A* \mathfrak{C} *U* π

lemma (in *is-cat-finite-obj-prod*) *cat-fin-obj-prod-index-vfinite*: *vfinite I*
 $\langle proof \rangle$

sublocale *is-cat-finite-obj-prod* $\subseteq I$: *finite-category* α $\langle :_C I \rangle$
 $\langle proof \rangle$

lemma (in *is-cat-finite-obj-coproduct*) *cat-fin-obj-coproduct-index-vfinite*:
vfinite I
 $\langle proof \rangle$

sublocale *is-cat-finite-obj-coproduct* $\subseteq I$: *finite-category* α $\langle :_C I \rangle$
 $\langle proof \rangle$

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 $\pi : P' <_{CF.\Pi.f\in} A' : I' \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

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-coproduct*)
is-cat-finite-obj-coproduct-axioms'[*cat-lim-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $U' = U$ **and** $A' = A$ **and** $I' = I$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\pi : A' >_{CF.\Pi.f\in} U' : I' \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

mk-ide rf
is-cat-finite-obj-coproduct-def[*unfolded is-cat-finite-obj-coproduct-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 $\pi : A >_{CF.\Pi.f\text{in}} P : I \mapsto_{C\alpha} \text{op-cat } \mathfrak{C}$
⟨proof⟩

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

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

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

5.4 Product and coproduct of two objects

5.4.1 Definition and elementary properties

locale is-cat-obj-prod-2 = is-cat-obj-prod $\alpha \langle 2_{\mathbb{N}} \rangle \langle \text{if}2 \ a \ b \rangle \mathfrak{C} \ P \ \pi$
for $\alpha \ a \ b \ \mathfrak{C} \ P \ \pi$

syntax -is-cat-obj-prod-2 :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$
(⟨(- :/ - <_{CF.x} {-,-} :/ 2_C \mapsto_{C1} -)⟩ [51, 51, 51, 51, 51] 51)

syntax-consts -is-cat-obj-prod-2 \cong is-cat-obj-prod-2

translations $\pi : P <_{CF.x} \{a,b\} : 2_C \mapsto_{C\alpha} \mathfrak{C} \cong$
CONST is-cat-obj-prod-2 $\alpha \ a \ b \ \mathfrak{C} \ P \ \pi$

locale is-cat-obj-coprod-2 = is-cat-obj-coprod $\alpha \langle 2_{\mathbb{N}} \rangle \langle \text{if}2 \ a \ b \rangle \mathfrak{C} \ P \ \pi$
for $\alpha \ a \ b \ \mathfrak{C} \ P \ \pi$

syntax -is-cat-obj-coprod-2 :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$
(⟨(- :/ {-,-} >_{CF.w} - :/ 2_C \mapsto_{C1} -)⟩ [51, 51, 51, 51, 51] 51)

syntax-consts -is-cat-obj-coprod-2 \cong is-cat-obj-coprod-2

translations $\pi : \{a,b\} >_{CF.w} U : 2_C \mapsto_{C\alpha} \mathfrak{C} \cong$
CONST is-cat-obj-coprod-2 $\alpha \ a \ b \ \mathfrak{C} \ U \ \pi$

abbreviation proj-fst **where** proj-fst $\pi \equiv \text{vpfst } (\pi(\text{NTMap}))$

abbreviation proj-snd **where** proj-snd $\pi \equiv \text{vpsnd } (\pi(\text{NTMap}))$

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 $\pi : P' <_{CF.x} \{a',b'\} : 2_C \mapsto_{C\alpha} \mathfrak{C}'$

$\langle proof \rangle$

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 $\pi : \{a', b'\} >_{CF.\omega} P' : \mathcal{2}_C \mapsto \mapsto_{C\alpha} \mathfrak{C}'$
 $\langle proof \rangle$

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 $\pi : \{a, b\} >_{CF.\omega} P : \mathcal{2}_C \mapsto \mapsto_{C\alpha} \text{op-cat } \mathfrak{C}$
 $\langle proof \rangle$

lemma (**in** *is-cat-obj-prod-2*) *is-cat-obj-coprod-2-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
shows *op-ntcf* $\pi : \{a, b\} >_{CF.\omega} P : \mathcal{2}_C \mapsto \mapsto_{C\alpha} \mathfrak{C}'$
 $\langle proof \rangle$

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 $\pi : P <_{CF.\times} \{a, b\} : \mathcal{2}_C \mapsto \mapsto_{C\alpha} \text{op-cat } \mathfrak{C}$
 $\langle proof \rangle$

lemma (**in** *is-cat-obj-coprod-2*) *is-cat-obj-prod-2-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
shows *op-ntcf* $\pi : P <_{CF.\times} \{a, b\} : \mathcal{2}_C \mapsto \mapsto_{C\alpha} \mathfrak{C}'$
 $\langle proof \rangle$

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* \subseteq *is-cat-finite-obj-prod* $\alpha \langle \mathcal{2}_N \rangle \langle \text{if2 } a \ b \rangle \mathfrak{C} P \pi$
 $\langle proof \rangle$

sublocale *is-cat-obj-coprod-2* \subseteq *is-cat-finite-obj-coprod* $\alpha \langle \mathcal{2}_N \rangle \langle \text{if2 } a \ b \rangle \mathfrak{C} P \pi$
 $\langle proof \rangle$

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*]: $a \in_{\circ} \mathfrak{C}(\text{Obj})$
and *cat-obj-prod-2-right-in-Obj*[*cat-lim-cs-intros*]: $b \in_{\circ} \mathfrak{C}(\text{Obj})$
 $\langle proof \rangle$

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]: $a \in_{\circ} \mathfrak{C}(\text{Obj})$
and cat-obj-coprod-2-right-in-Obj[cat-lim-cs-intros]: $b \in_{\circ} \mathfrak{C}(\text{Obj})$
 ⟨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:

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

lemma helper-I2-proj-fst-proj-snd-iff':

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

5.4.2 Universal property

lemma (in is-cat-obj-prod-2) cat-obj-prod-2-unique-cone':

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

lemma (in is-cat-obj-prod-2) cat-obj-prod-2-unique:

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

lemma (in is-cat-obj-prod-2) cat-obj-prod-2-unique':

assumes $\pi' : P' <_{CF.\times} \{a,b\} : \mathbb{2}_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$
shows
 $\exists ! f'. f' : P' \mapsto_{\mathfrak{C}} P \wedge$
 $\text{proj-fst } \pi' = \text{proj-fst } \pi \circ_{A\mathfrak{C}} f' \wedge$
 $\text{proj-snd } \pi' = \text{proj-snd } \pi \circ_{A\mathfrak{C}} f'$
 ⟨proof⟩

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

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

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

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

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

assumes $\pi' : \{a,b\} >_{CF.\sqcup} P' : \mathbb{2}_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$

shows

$\exists !f'. f' : P \mapsto_{\mathfrak{C}} P' \wedge$
 $proj\text{-fst } \pi' = f' \circ_{A\mathfrak{C}} proj\text{-fst } \pi \wedge$
 $proj\text{-snd } \pi' = f' \circ_{A\mathfrak{C}} proj\text{-snd } \pi$
 $\langle proof \rangle$

lemma *cat-obj-prod-2-ex-is-iso-arr*:

assumes $\pi : P <_{CF.\times} \{a,b\} : 2_C \mapsto_{C\alpha} \mathfrak{C}$
and $\pi' : P' <_{CF.\times} \{a,b\} : 2_C \mapsto_{C\alpha} \mathfrak{C}$
obtains f where $f : P' \mapsto_{iso\mathfrak{C}} P$ **and** $\pi' = \pi \cdot_{NTCF} ntcf\text{-const } (:_C (2_{\mathbb{N}})) \mathfrak{C} f$
 $\langle proof \rangle$

lemma *cat-obj-coprod-2-ex-is-iso-arr*:

assumes $\pi : \{a,b\} >_{CF.\sqcup} U : 2_C \mapsto_{C\alpha} \mathfrak{C}$
and $\pi' : \{a,b\} >_{CF.\sqcup} U' : 2_C \mapsto_{C\alpha} \mathfrak{C}$
obtains f where $f : U \mapsto_{iso\mathfrak{C}} U'$ **and** $\pi' = ntcf\text{-const } (:_C (2_{\mathbb{N}})) \mathfrak{C} f \cdot_{NTCF} \pi$
 $\langle proof \rangle$

5.5 Projection cone

5.5.1 Definition and elementary properties

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

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

Components.

lemma *ntcf-obj-prod-base-components*:

shows *ntcf-obj-prod-base* $\mathfrak{C} I F P f(\text{NTMap}) = (\lambda j \epsilon_{\circ} :_C I(\text{Obj}). f j)$
and *ntcf-obj-prod-base* $\mathfrak{C} I F P f(\text{NTDom}) = cf\text{-const } (:_C I) \mathfrak{C} P$
and *ntcf-obj-prod-base* $\mathfrak{C} I F P f(\text{NTCod}) = \text{:}\rightarrow : I F \mathfrak{C}$
and *ntcf-obj-prod-base* $\mathfrak{C} I F P f(\text{NTDGDom}) = :_C I$
and *ntcf-obj-prod-base* $\mathfrak{C} I F P f(\text{NTDGCod}) = \mathfrak{C}$
 $\langle proof \rangle$

lemma *ntcf-obj-coprod-base-components*:

shows *ntcf-obj-coprod-base* $\mathfrak{C} I F P f(\text{NTMap}) = (\lambda j \epsilon_{\circ} :_C I(\text{Obj}). f j)$
and *ntcf-obj-coprod-base* $\mathfrak{C} I F P f(\text{NTDom}) = \text{:}\rightarrow : I F \mathfrak{C}$
and *ntcf-obj-coprod-base* $\mathfrak{C} I F P f(\text{NTCod}) = cf\text{-const } (:_C I) \mathfrak{C} P$
and *ntcf-obj-coprod-base* $\mathfrak{C} I F P f(\text{NTDGDom}) = :_C I$
and *ntcf-obj-coprod-base* $\mathfrak{C} I F P f(\text{NTDGCod}) = \mathfrak{C}$
 $\langle proof \rangle$

Duality.

lemma (in *cf-discrete*) *op-ntcf-ntcf-obj-coprod-base[cat-op-simps]*:

op-ntcf (*ntcf-obj-coprod-base* $\mathfrak{C} I F P f$) =
ntcf-obj-prod-base (*op-cat* \mathfrak{C}) *I F P f*
 $\langle proof \rangle$

lemma (in *cf-discrete*) *op-ntcf-ntcf-obj-prod-base[cat-op-simps]*:

op-ntcf (*ntcf-obj-prod-base* $\mathfrak{C} I F P f$) =
ntcf-obj-coprod-base (*op-cat* \mathfrak{C}) *I F P f*
 $\langle proof \rangle$

5.5.2 Natural transformation map

mk-VLambda *ntcf-obj-prod-base-components*(1)
 $|vsv\ ntcf\text{-obj-prod-base-NTMap-vs}[cat\text{-cs-intros}]|$
 $|vdomain\ ntcf\text{-obj-prod-base-NTMap-vdomain}[cat\text{-cs-simps}]|$
 $|app\ ntcf\text{-obj-prod-base-NTMap-app}[cat\text{-cs-simps}]|$

mk-VLambda *ntcf-obj-coprod-base-components*(1)
 $|vsv\ ntcf\text{-obj-coprod-base-NTMap-vs}[cat\text{-cs-intros}]|$
 $|vdomain\ ntcf\text{-obj-coprod-base-NTMap-vdomain}[cat\text{-cs-simps}]|$
 $|app\ ntcf\text{-obj-coprod-base-NTMap-app}[cat\text{-cs-simps}]|$

5.5.3 Projection natural transformation is a cone

lemma (in *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-cone*:

assumes $P \in_{\circ} \mathfrak{C}(\text{Obj})$ **and** $\bigwedge a. a \in_{\circ} I \implies f\ a : P \mapsto_{\mathfrak{C}} F\ a$
shows $ntcf\text{-obj-prod-base } \mathfrak{C}\ I\ F\ P\ f : P <_{CF.cone} \implies I\ F\ \mathfrak{C} : :_C\ I \mapsto_{C\alpha} \mathfrak{C}$
<proof>

lemma (in *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-coprod-base-is-cat-cocone*:

assumes $P \in_{\circ} \mathfrak{C}(\text{Obj})$ **and** $\bigwedge a. a \in_{\circ} I \implies f\ a : F\ a \mapsto_{\mathfrak{C}} P$
shows $ntcf\text{-obj-coprod-base } \mathfrak{C}\ I\ F\ P\ f :$
 $\implies I\ F\ \mathfrak{C} >_{CF.cocone} P : :_C\ I \mapsto_{C\alpha} \mathfrak{C}$
<proof>

lemma (in *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-obj-prod*:

assumes $P \in_{\circ} \mathfrak{C}(\text{Obj})$
and $\bigwedge a. a \in_{\circ} I \implies f\ a : P \mapsto_{\mathfrak{C}} F\ a$
and $\bigwedge u'\ r'.$
 $\llbracket u' : r' <_{CF.cone} \implies I\ F\ \mathfrak{C} : :_C\ I \mapsto_{C\alpha} \mathfrak{C} \rrbracket \implies$
 $\exists ! f'.$
 $f' : r' \mapsto_{\mathfrak{C}} P \wedge$
 $u' = ntcf\text{-obj-prod-base } \mathfrak{C}\ I\ F\ P\ f \cdot_{NTCF} ntcf\text{-const } (:_C\ I)\ \mathfrak{C}\ f'$
shows $ntcf\text{-obj-prod-base } \mathfrak{C}\ I\ F\ P\ f : P <_{CF.\Pi} F : I \mapsto_{C\alpha} \mathfrak{C}$
<proof>

lemma (in *tm-cf-discrete*) *tm-cf-discrete-ntcf-obj-coprod-base-is-cat-obj-coprod*:

assumes $P \in_{\circ} \mathfrak{C}(\text{Obj})$
and $\bigwedge a. a \in_{\circ} I \implies f\ a : F\ a \mapsto_{\mathfrak{C}} P$
and $\bigwedge u'\ r'. \llbracket u' : \implies I\ F\ \mathfrak{C} >_{CF.cocone} r' : :_C\ I \mapsto_{C\alpha} \mathfrak{C} \rrbracket \implies$
 $\exists ! f'.$
 $f' : P \mapsto_{\mathfrak{C}} r' \wedge$
 $u' = ntcf\text{-const } (:_C\ I)\ \mathfrak{C}\ f' \cdot_{NTCF} ntcf\text{-obj-coprod-base } \mathfrak{C}\ I\ F\ P\ f$
shows $ntcf\text{-obj-coprod-base } \mathfrak{C}\ I\ F\ P\ f : F >_{CF.\Pi} P : I \mapsto_{C\alpha} \mathfrak{C}$
(is <?nc : F >_{CF.\Pi} P : I \mapsto_{C\alpha} \mathfrak{C})
<proof>

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

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

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

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

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

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

Elementary properties.

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

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

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

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

shows $x(\text{NTMap})(\mathbf{a}_{SS}) = x(\text{NTMap})(\mathbf{a}_{SS}) \circ_{A\mathfrak{C}} \mathfrak{g}$
and $x(\text{NTMap})(\mathbf{b}_{SS}) = x(\text{NTMap})(\mathbf{b}_{SS}) \circ_{A\mathfrak{C}} \mathfrak{f}$
and $x(\text{NTMap})(\mathbf{a}_{SS}) \circ_{A\mathfrak{C}} \mathfrak{g} = x(\text{NTMap})(\mathbf{b}_{SS}) \circ_{A\mathfrak{C}} \mathfrak{f}$
⟨proof⟩

6.1.2 Universal property

lemma *is-cat-pullbackI'*:

assumes $x : X <_{CF.cocone} \langle \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \rangle_{CF\mathfrak{C}} : \rightarrow \leftarrow C \mapsto C\alpha \mathfrak{C}$
and *cf-scospan* $\alpha \mathbf{a} \mathfrak{g} \mathbf{0} \mathfrak{f} \mathbf{b} \mathfrak{C}$
and $\wedge x' X'. x' : X' <_{CF.cocone} \langle \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \rangle_{CF\mathfrak{C}} : \rightarrow \leftarrow C \mapsto C\alpha \mathfrak{C} \implies$
 $\exists ! f'$.
 $f' : X' \mapsto_{\mathfrak{C}} X \wedge$
 $x'(\text{NTMap})(\mathbf{a}_{SS}) = x(\text{NTMap})(\mathbf{a}_{SS}) \circ_{A\mathfrak{C}} f' \wedge$
 $x'(\text{NTMap})(\mathbf{b}_{SS}) = x(\text{NTMap})(\mathbf{b}_{SS}) \circ_{A\mathfrak{C}} f'$
shows $x : X <_{CF.pb} \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \mapsto C\alpha \mathfrak{C}$
⟨proof⟩

lemma *is-cat-pushoutI'*:

assumes $x : \langle \mathbf{a} \leftarrow \mathfrak{g} \leftarrow \mathbf{0} \rightarrow \mathfrak{f} \rightarrow \mathbf{b} \rangle_{CF\mathfrak{C}} >_{CF.cocone} X : \leftarrow \rightarrow C \mapsto C\alpha \mathfrak{C}$
and *cf-sspan* $\alpha \mathbf{a} \mathfrak{g} \mathbf{0} \mathfrak{f} \mathbf{b} \mathfrak{C}$
and $\wedge x' X'. x' : \langle \mathbf{a} \leftarrow \mathfrak{g} \leftarrow \mathbf{0} \rightarrow \mathfrak{f} \rightarrow \mathbf{b} \rangle_{CF\mathfrak{C}} >_{CF.cocone} X' : \leftarrow \rightarrow C \mapsto C\alpha \mathfrak{C} \implies$
 $\exists ! f'$.
 $f' : X \mapsto_{\mathfrak{C}} X' \wedge$
 $x'(\text{NTMap})(\mathbf{a}_{SS}) = f' \circ_{A\mathfrak{C}} x(\text{NTMap})(\mathbf{a}_{SS}) \wedge$
 $x'(\text{NTMap})(\mathbf{b}_{SS}) = f' \circ_{A\mathfrak{C}} x(\text{NTMap})(\mathbf{b}_{SS})$
shows $x : \mathbf{a} \leftarrow \mathfrak{g} \leftarrow \mathbf{0} \rightarrow \mathfrak{f} \rightarrow \mathbf{b} >_{CF.po} X \mapsto C\alpha \mathfrak{C}$
⟨proof⟩

lemma (in *is-cat-pullback*) *cat-pb-unique-cone*:

assumes $x' : X' <_{CF.cocone} \langle \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \rangle_{CF\mathfrak{C}} : \rightarrow \leftarrow C \mapsto C\alpha \mathfrak{C}$
shows $\exists ! f'$.
 $f' : X' \mapsto_{\mathfrak{C}} X \wedge$
 $x'(\text{NTMap})(\mathbf{a}_{SS}) = x(\text{NTMap})(\mathbf{a}_{SS}) \circ_{A\mathfrak{C}} f' \wedge$
 $x'(\text{NTMap})(\mathbf{b}_{SS}) = x(\text{NTMap})(\mathbf{b}_{SS}) \circ_{A\mathfrak{C}} f'$
⟨proof⟩

lemma (in *is-cat-pullback*) *cat-pb-unique*:

assumes $x' : X' <_{CF.pb} \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \mapsto C\alpha \mathfrak{C}$
shows $\exists ! f'. f' : X' \mapsto_{\mathfrak{C}} X \wedge x' = x \cdot_{NTCF} \text{ntcf-const} \rightarrow \leftarrow C \mathfrak{C} f'$
⟨proof⟩

lemma (in *is-cat-pullback*) *cat-pb-unique'*:

assumes $x' : X' <_{CF.pb} \mathbf{a} \rightarrow \mathfrak{g} \rightarrow \mathbf{0} \leftarrow \mathfrak{f} \leftarrow \mathbf{b} \mapsto C\alpha \mathfrak{C}$
shows $\exists ! f'$.
 $f' : X' \mapsto_{\mathfrak{C}} X \wedge$
 $x'(\text{NTMap})(\mathbf{a}_{SS}) = x(\text{NTMap})(\mathbf{a}_{SS}) \circ_{A\mathfrak{C}} f' \wedge$
 $x'(\text{NTMap})(\mathbf{b}_{SS}) = x(\text{NTMap})(\mathbf{b}_{SS}) \circ_{A\mathfrak{C}} f'$
⟨proof⟩

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

assumes $x' : \langle \mathbf{a} \leftarrow \mathfrak{g} \leftarrow \mathbf{0} \rightarrow \mathfrak{f} \rightarrow \mathbf{b} \rangle_{CF\mathfrak{C}} >_{CF.cocone} X' : \leftarrow \rightarrow C \mapsto C\alpha \mathfrak{C}$
shows $\exists ! f'$.
 $f' : X \mapsto_{\mathfrak{C}} X' \wedge$
 $x'(\text{NTMap})(\mathbf{a}_{SS}) = f' \circ_{A\mathfrak{C}} x(\text{NTMap})(\mathbf{a}_{SS}) \wedge$
 $x'(\text{NTMap})(\mathbf{b}_{SS}) = f' \circ_{A\mathfrak{C}} x(\text{NTMap})(\mathbf{b}_{SS})$
⟨proof⟩

lemma (in *is-cat-pushout*) *cat-po-unique*:

assumes $x' : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X' \mapsto \mapsto_{C\alpha} \mathfrak{C}$
shows $\exists ! f'. f' : X \mapsto_{\mathfrak{C}} X' \wedge x' = ntcf-const \leftarrow \mapsto_C \mathfrak{C} f' \cdot_{NTCF} x$
<proof>

lemma (in *is-cat-pushout*) *cat-po-unique'*:

assumes $x' : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X' \mapsto \mapsto_{C\alpha} \mathfrak{C}$
shows $\exists ! f'$.

$f' : X \mapsto_{\mathfrak{C}} X' \wedge$
 $x'(\downarrow NTMap)(\downarrow \mathbf{a}_{SS}) = f' \circ_{A\mathfrak{C}} x(\downarrow NTMap)(\downarrow \mathbf{a}_{SS}) \wedge$
 $x'(\downarrow NTMap)(\downarrow \mathbf{b}_{SS}) = f' \circ_{A\mathfrak{C}} x(\downarrow NTMap)(\downarrow \mathbf{b}_{SS})$

<proof>

lemma *cat-pullback-ex-is-iso-arr*:

assumes $x : X <_{CF.pb} \mathbf{a} \rightarrow \mathbf{g} \rightarrow \mathbf{o} \leftarrow \mathbf{f} \leftarrow \mathbf{b} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $x' : X' <_{CF.pb} \mathbf{a} \rightarrow \mathbf{g} \rightarrow \mathbf{o} \leftarrow \mathbf{f} \leftarrow \mathbf{b} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : X' \mapsto_{iso\mathfrak{C}} X$
and $x' = x \cdot_{NTCF} ntcf-const \rightarrow \leftarrow_C \mathfrak{C} f$

<proof>

lemma *cat-pullback-ex-is-iso-arr'*:

assumes $x : X <_{CF.pb} \mathbf{a} \rightarrow \mathbf{g} \rightarrow \mathbf{o} \leftarrow \mathbf{f} \leftarrow \mathbf{b} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $x' : X' <_{CF.pb} \mathbf{a} \rightarrow \mathbf{g} \rightarrow \mathbf{o} \leftarrow \mathbf{f} \leftarrow \mathbf{b} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : X' \mapsto_{iso\mathfrak{C}} X$
and $x'(\downarrow NTMap)(\downarrow \mathbf{a}_{SS}) = x(\downarrow NTMap)(\downarrow \mathbf{a}_{SS}) \circ_{A\mathfrak{C}} f$
and $x'(\downarrow NTMap)(\downarrow \mathbf{b}_{SS}) = x(\downarrow NTMap)(\downarrow \mathbf{b}_{SS}) \circ_{A\mathfrak{C}} f$

<proof>

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

assumes $x : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $x' : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X' \mapsto \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : X \mapsto_{iso\mathfrak{C}} X'$
and $x' = ntcf-const \leftarrow \mapsto_C \mathfrak{C} f \cdot_{NTCF} x$

<proof>

lemma *cat-pushout-ex-is-iso-arr'*:

assumes $x : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $x' : \mathbf{a} \leftarrow \mathbf{g} \leftarrow \mathbf{o} \rightarrow \mathbf{f} \rightarrow \mathbf{b} >_{CF.po} X' \mapsto \mapsto_{C\alpha} \mathfrak{C}$
obtains f **where** $f : X \mapsto_{iso\mathfrak{C}} X'$
and $x'(\downarrow NTMap)(\downarrow \mathbf{a}_{SS}) = f \circ_{A\mathfrak{C}} x(\downarrow NTMap)(\downarrow \mathbf{a}_{SS})$
and $x'(\downarrow NTMap)(\downarrow \mathbf{b}_{SS}) = f \circ_{A\mathfrak{C}} x(\downarrow NTMap)(\downarrow \mathbf{b}_{SS})$

<proof>

7 Equalizers and coequalizers as limits and colimits

7.1 Equalizer and coequalizer

7.1.1 Definition and elementary properties

See [2]⁶.

locale *is-cat-equalizer* =

is-cat-limit $\alpha \langle \uparrow_C (\mathbf{a}_{PL} F) (\mathbf{b}_{PL} F) F \rangle \mathfrak{C} \langle \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathbf{a}_{PL} F) (\mathbf{b}_{PL} F) F \mathbf{a} \mathbf{b} F' \rangle E \varepsilon +$
 $F': vsv F'$

for $\alpha \mathbf{a} \mathbf{b} F F' \mathfrak{C} E \varepsilon +$

assumes *cat-eq-F-in-Vset*[*cat-lim-cs-intros*]: $F \in_0 Vset \alpha$

and *cat-eq-F-ne*[*cat-lim-cs-intros*]: $F \neq 0$

and *cat-eq-F'-vdomain*[*cat-lim-cs-simps*]: $\mathcal{D}_0 F' = F$

and *cat-eq-F'-app-is-arr*[*cat-lim-cs-intros*]: $\mathfrak{f} \in_0 F \implies F'(\mathfrak{f}) : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$

syntax *-is-cat-equalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle - : / \langle - \rangle_{CF.eq} '(-,-,-)' : / \uparrow_C \mapsto_{C1} - \rangle [51, 51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-cat-equalizer* $\hat{=} is-cat-equalizer$

translations $\varepsilon : E \langle_{CF.eq} (\mathbf{a}, \mathbf{b}, F, F') : \uparrow_C \mapsto_{C\alpha} \mathfrak{C} \hat{=}$

CONST is-cat-equalizer $\alpha \mathbf{a} \mathbf{b} F F' \mathfrak{C} E \varepsilon$

locale *is-cat-coequalizer* =

is-cat-colimit $\alpha \langle \uparrow_C (\mathbf{b}_{PL} F) (\mathbf{a}_{PL} F) F \rangle \mathfrak{C} \langle \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathbf{b}_{PL} F) (\mathbf{a}_{PL} F) F \mathbf{b} \mathbf{a} F' \rangle E \varepsilon +$
 $F': vsv F'$

for $\alpha \mathbf{a} \mathbf{b} F F' \mathfrak{C} E \varepsilon +$

assumes *cat-coeq-F-in-Vset*[*cat-lim-cs-intros*]: $F \in_0 Vset \alpha$

and *cat-coeq-F-ne*[*cat-lim-cs-intros*]: $F \neq 0$

and *cat-coeq-F'-vdomain*[*cat-lim-cs-simps*]: $\mathcal{D}_0 F' = F$

and *cat-coeq-F'-app-is-arr*[*cat-lim-cs-intros*]: $\mathfrak{f} \in_0 F \implies F'(\mathfrak{f}) : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$

syntax *-is-cat-coequalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle - : / '(-,-,-)' \rangle_{CF.coeq} - : / \uparrow_C \mapsto_{C1} - \rangle [51, 51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-cat-coequalizer* $\hat{=} is-cat-coequalizer$

translations $\varepsilon : (\mathbf{a}, \mathbf{b}, F, F') \rangle_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C} \hat{=}$

CONST is-cat-coequalizer $\alpha \mathbf{a} \mathbf{b} F F' \mathfrak{C} E \varepsilon$

Rules.

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

assumes $\alpha' = \alpha$

and $E' = E$

and $\mathbf{a}' = \mathbf{a}$

and $\mathbf{b}' = \mathbf{b}$

and $F'' = F$

and $F''' = F'$

and $\mathfrak{C}' = \mathfrak{C}$

shows $\varepsilon : E' \langle_{CF.eq} (\mathbf{a}', \mathbf{b}', F'', F''') : \uparrow_C \mapsto_{C\alpha'} \mathfrak{C}'$

<proof>

mk-ide rf *is-cat-equalizer-def*[*unfolded is-cat-equalizer-axioms-def*]

[intro is-cat-equalizerI]

[dest is-cat-equalizerD[dest]]

[elim is-cat-equalizerE[elim]]

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

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

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

assumes $\alpha' = \alpha$
 and $E' = E$
 and $\mathbf{a}' = \mathbf{a}$
 and $\mathbf{b}' = \mathbf{b}$
 and $F'' = F$
 and $F''' = F'$
 and $\mathfrak{C}' = \mathfrak{C}$
 shows $\varepsilon : (\mathbf{a}', \mathbf{b}', F'', F''') >_{CF.coeq} E' : \uparrow_C \mapsto \mapsto_{C\alpha'} \mathfrak{C}'$
 <proof>

mk-ide rf *is-cat-coequalizer-def*[*unfolded is-cat-coequalizer-axioms-def*]

|*intro is-cat-coequalizerI*||
 |*dest is-cat-coequalizerD*[*dest*]|
 |*elim is-cat-coequalizerE*[*elim*]|

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

Elementary properties.

lemma (in *is-cat-equalizer*)

cat-eq-a[*cat-lim-cs-intros*]: $\mathbf{a} \in_{\circ} \mathfrak{C}(\text{Obj})$
 and *cat-eq-b*[*cat-lim-cs-intros*]: $\mathbf{b} \in_{\circ} \mathfrak{C}(\text{Obj})$
 <proof>

lemma (in *is-cat-coequalizer*)

cat-coeq-a[*cat-lim-cs-intros*]: $\mathbf{a} \in_{\circ} \mathfrak{C}(\text{Obj})$
 and *cat-coeq-b*[*cat-lim-cs-intros*]: $\mathbf{b} \in_{\circ} \mathfrak{C}(\text{Obj})$
 <proof>

sublocale *is-cat-equalizer* \subseteq *cf-parallel* $\alpha \langle \mathbf{a}_{PL} F \rangle \langle \mathbf{b}_{PL} F \rangle F \mathbf{a} \mathbf{b} F' \mathfrak{C}$
 <proof>

sublocale *is-cat-coequalizer* \subseteq *cf-parallel* $\alpha \langle \mathbf{b}_{PL} F \rangle \langle \mathbf{a}_{PL} F \rangle F \mathbf{b} \mathbf{a} F' \mathfrak{C}$
 <proof>

Duality.

lemma (in *is-cat-equalizer*) *is-cat-equalizer-op*:

op-ntcf $\varepsilon : (\mathbf{a}, \mathbf{b}, F, F') >_{CF.coeq} E : \uparrow_C \mapsto \mapsto_{C\alpha} \text{op-cat } \mathfrak{C}$
 <proof>

lemma (in *is-cat-equalizer*) *is-cat-coequalizer-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
 shows *op-ntcf* $\varepsilon : (\mathbf{a}, \mathbf{b}, F, F') >_{CF.coeq} E : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}'$
 <proof>

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

lemma (in *is-cat-coequalizer*) *is-cat-equalizer-op*:

op-ntcf $\varepsilon : E <_{CF.eq} (\mathbf{a}, \mathbf{b}, F, F') : \uparrow_C \mapsto \mapsto_{C\alpha} \text{op-cat } \mathfrak{C}$
 <proof>

lemma (in *is-cat-coequalizer*) *is-cat-equalizer-op'*[*cat-op-intros*]:

assumes $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
 shows *op-ntcf* $\varepsilon : E <_{CF.eq} (\mathbf{a}, \mathbf{b}, F, F') : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}'$
 <proof>

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

Further properties.

lemma (in *category*) *cat-cf-parallel-ab*:

assumes $vsv F'$
and $F \in_0 Vset \alpha$
and $\mathcal{D}_0 F' = F$
and $\wedge f. f \in_0 F \implies F'(\downarrow f) : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$
and $\mathbf{a} \in_0 \mathfrak{C}(\downarrow Obj)$
and $\mathbf{b} \in_0 \mathfrak{C}(\downarrow Obj)$
shows *cf-parallel* α $(\mathbf{a}_{PL} F)$ $(\mathbf{b}_{PL} F)$ F \mathbf{a} \mathbf{b} $F' \mathfrak{C}$
<proof>

lemma (in *category*) *cat-cf-parallel-ba*:

assumes $vsv F'$
and $F \in_0 Vset \alpha$
and $\mathcal{D}_0 F' = F$
and $\wedge f. f \in_0 F \implies F'(\downarrow f) : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$
and $\mathbf{a} \in_0 \mathfrak{C}(\downarrow Obj)$
and $\mathbf{b} \in_0 \mathfrak{C}(\downarrow Obj)$
shows *cf-parallel* α $(\mathbf{b}_{PL} F)$ $(\mathbf{a}_{PL} F)$ F \mathbf{b} \mathbf{a} $F' \mathfrak{C}$
<proof>

lemma *cat-cone-cf-par-eps-NTMap-app*:

assumes $\varepsilon :$
 $E <_{CF.cone} \uparrow \mapsto \uparrow_{CF} \mathfrak{C} (\mathbf{a}_{PL} F) (\mathbf{b}_{PL} F) F \mathbf{a} \mathbf{b} F' :$
 $\uparrow_C (\mathbf{a}_{PL} F) (\mathbf{b}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $vsv F'$
and $F \in_0 Vset \alpha$
and $\mathcal{D}_0 F' = F$
and $\wedge f. f \in_0 F \implies F'(\downarrow f) : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$
and $f \in_0 F$
shows $\varepsilon(\downarrow NTMap)(\downarrow \mathbf{b}_{PL} F) = F'(\downarrow f) \circ_{A\mathfrak{C}} \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F)$
<proof>

lemma *cat-cocone-cf-par-eps-NTMap-app*:

assumes $\varepsilon :$
 $\uparrow \mapsto \uparrow_{CF} \mathfrak{C} (\mathbf{b}_{PL} F) (\mathbf{a}_{PL} F) F \mathbf{b} \mathbf{a} F' >_{CF.cocone} E :$
 $\uparrow_C (\mathbf{b}_{PL} F) (\mathbf{a}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $vsv F'$
and $F \in_0 Vset \alpha$
and $\mathcal{D}_0 F' = F$
and $\wedge f. f \in_0 F \implies F'(\downarrow f) : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$
and $f \in_0 F$
shows $\varepsilon(\downarrow NTMap)(\downarrow \mathbf{b}_{PL} F) = \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F) \circ_{A\mathfrak{C}} F'(\downarrow f)$
<proof>

lemma (in *is-cat-equalizer*) *cat-eq-eps-NTMap-app*:

assumes $f \in_0 F$
shows $\varepsilon(\downarrow NTMap)(\downarrow \mathbf{b}_{PL} F) = F'(\downarrow f) \circ_{A\mathfrak{C}} \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F)$
<proof>

lemma (in *is-cat-coequalizer*) *cat-coeq-eps-NTMap-app*:

assumes $f \in_0 F$
shows $\varepsilon(\downarrow NTMap)(\downarrow \mathbf{b}_{PL} F) = \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F) \circ_{A\mathfrak{C}} F'(\downarrow f)$
<proof>

lemma (in *is-cat-equalizer*) *cat-eq-Comp-eq*:

assumes $g \in_0 F$ **and** $f \in_0 F$
shows $F'(\downarrow g) \circ_{A\mathfrak{C}} \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F) = F'(\downarrow f) \circ_{A\mathfrak{C}} \varepsilon(\downarrow NTMap)(\downarrow \mathbf{a}_{PL} F)$

$\langle \text{proof} \rangle$

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

assumes $\mathfrak{g} \in_{\circ} F$ and $\mathfrak{f} \in_{\circ} F$

shows $\varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) \circ_{A\mathfrak{C}} F'(\ulcorner \mathfrak{g} \urcorner) = \varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) \circ_{A\mathfrak{C}} F'(\ulcorner \mathfrak{f} \urcorner)$

$\langle \text{proof} \rangle$

7.1.2 Universal property

lemma *is-cat-equalizerI'*:

assumes ε :

$E <_{CF.cone} \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mathfrak{a} \mathfrak{b} F'$:

$\uparrow_C (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C}$

and $vsv F'$

and $F \in_{\circ} Vset \alpha$

and $\mathcal{D}_{\circ} F' = F$

and $\wedge \mathfrak{f}. \mathfrak{f} \in_{\circ} F \implies F'(\ulcorner \mathfrak{f} \urcorner) : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{b}$

and $\mathfrak{f} \in_{\circ} F$

and $\wedge \varepsilon' E'. \varepsilon'$:

$E' <_{CF.cone} \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mathfrak{a} \mathfrak{b} F'$:

$\uparrow_C (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C} \implies$

$\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) = \varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) \circ_{A\mathfrak{C}} f'$

shows $\varepsilon : E <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$

$\langle \text{proof} \rangle$

lemma *is-cat-coequalizerI'*:

assumes ε :

$\uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mathfrak{b} \mathfrak{a} F' >_{CF.cocone} E$:

$\uparrow_C (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C}$

and $vsv F'$

and $F \in_{\circ} Vset \alpha$

and $\mathcal{D}_{\circ} F' = F$

and $\wedge \mathfrak{f}. \mathfrak{f} \in_{\circ} F \implies F'(\ulcorner \mathfrak{f} \urcorner) : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{a}$

and $\mathfrak{f} \in_{\circ} F$

and $\wedge \varepsilon' E'. \varepsilon'$:

$\uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mathfrak{b} \mathfrak{a} F' >_{CF.cocone} E'$:

$\uparrow_C (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C} \implies$

$\exists ! f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) = f' \circ_{A\mathfrak{C}} \varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner)$

shows $\varepsilon : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$

$\langle \text{proof} \rangle$

lemma (in *is-cat-equalizer*) *cat-eq-unique-cone*:

assumes ε' :

$E' <_{CF.cone} \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mathfrak{a} \mathfrak{b} F' : \uparrow_C (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F \mapsto \mapsto_{C\alpha} \mathfrak{C}$

(**is** $\langle \varepsilon' : E' <_{CF.cone} ?II-II : ?II \mapsto \mapsto_{C\alpha} \mathfrak{C} \rangle$)

shows $\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) = \varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) \circ_{A\mathfrak{C}} f'$

$\langle \text{proof} \rangle$

lemma (in *is-cat-equalizer*) *cat-eq-unique*:

assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$

shows

$\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon' = \varepsilon \cdot_{NTCF} ntcf\text{-const} (\uparrow_C (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F) \mathfrak{C} f'$

$\langle \text{proof} \rangle$

lemma (in *is-cat-equalizer*) *cat-eq-unique'*:

assumes $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto \mapsto_{C\alpha} \mathfrak{C}$

shows $\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) = \varepsilon(\ulcorner NTMap \urcorner)(\ulcorner \mathfrak{a}_{PL} F \urcorner) \circ_{A\mathfrak{C}} f'$

$\langle \text{proof} \rangle$

lemma (in *is-cat-coequalizer*) *cat-coeq-unique-cocone*:

assumes ε' :

$$\uparrow \rightarrow \uparrow_{CF} \mathfrak{C} (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mathfrak{b} \mathfrak{a} F' >_{CF.cocone} E' :$$

$$\uparrow_C (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F \mapsto_{C\alpha} \mathfrak{C}$$

$$(\text{is } \langle \varepsilon' : ?II-II \rangle >_{CF.cocone} E' : ?II \mapsto_{C\alpha} \mathfrak{C})$$

shows $\exists ! f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon' (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) = f' \circ_A \varepsilon (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F)$

<proof>

lemma (in *is-cat-coequalizer*) *cat-coeq-unique*:

assumes $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E' : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

shows $\exists ! f'$.

$$f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon' = \text{ntcf-const} (\uparrow_C (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F) \mathfrak{C} f' \cdot_{NTCF} \varepsilon$$

<proof>

lemma (in *is-cat-coequalizer*) *cat-coeq-unique'*:

assumes $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E' : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

shows $\exists ! f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon' (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) = f' \circ_A \varepsilon (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F)$

<proof>

lemma *cat-equalizer-ex-is-iso-arr*:

assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

obtains f where $f : E' \mapsto_{iso\mathfrak{C}} E$

and $\varepsilon' = \varepsilon \cdot_{NTCF} \text{ntcf-const} (\uparrow_C (\mathfrak{a}_{PL} F) (\mathfrak{b}_{PL} F) F) \mathfrak{C} f$

<proof>

lemma *cat-equalizer-ex-is-iso-arr'*:

assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, F, F') : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

obtains f where $f : E' \mapsto_{iso\mathfrak{C}} E$

and $\varepsilon' (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) = \varepsilon (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) \circ_A \mathfrak{C} f$

and $\varepsilon' (\downarrow_{NTMap}) (\mathfrak{b}_{PL} F) = \varepsilon (\downarrow_{NTMap}) (\mathfrak{b}_{PL} F) \circ_A \mathfrak{C} f$

<proof>

lemma *cat-coequalizer-ex-is-iso-arr*:

assumes $\varepsilon : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E' : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

obtains f where $f : E \mapsto_{iso\mathfrak{C}} E'$

and $\varepsilon' = \text{ntcf-const} (\uparrow_C (\mathfrak{b}_{PL} F) (\mathfrak{a}_{PL} F) F) \mathfrak{C} f \cdot_{NTCF} \varepsilon$

<proof>

lemma *cat-coequalizer-ex-is-iso-arr'*:

assumes $\varepsilon : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, F, F') >_{CF.coeq} E' : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

obtains f where $f : E \mapsto_{iso\mathfrak{C}} E'$

and $\varepsilon' (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) = f \circ_A \varepsilon (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F)$

and $\varepsilon' (\downarrow_{NTMap}) (\mathfrak{b}_{PL} F) = f \circ_A \varepsilon (\downarrow_{NTMap}) (\mathfrak{b}_{PL} F)$

<proof>

7.1.3 Further properties

lemma (in *is-cat-equalizer*) *cat-eq-is-monic-arr*:

— See subsection 3.3 in [3].

$$\varepsilon (\downarrow_{NTMap}) (\mathfrak{a}_{PL} F) : E \mapsto_{\text{monic}} \mathfrak{a}$$

<proof>

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

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

7.2 Equalizer and coequalizer for two arrows

7.2.1 Definition and elementary properties

See [2]⁷.

locale *is-cat-equalizer-2* =
is-cat-limit $\alpha \langle \uparrow_C \mathbf{a}_{PL2} \mathbf{b}_{PL2} \mathbf{g}_{PL} \mathbf{f}_{PL} \rangle \mathfrak{C} \langle \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} \mathbf{a}_{PL2} \mathbf{b}_{PL2} \mathbf{g}_{PL} \mathbf{f}_{PL} \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \rangle E \varepsilon$
for $\alpha \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \mathfrak{C} E \varepsilon +$
assumes *cat-eq-g*[*cat-lim-cs-intros*]: $\mathbf{g} : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$
and *cat-eq-f*[*cat-lim-cs-intros*]: $\mathbf{f} : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$

syntax *is-cat-equalizer-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 (⟨(-) / - <_{CF.eq} '(-,-,-)' / $\uparrow_C \mapsto_{C1}$ -)⟩ [51, 51, 51, 51, 51, 51] 51)
syntax-consts *is-cat-equalizer-2* \equiv *is-cat-equalizer-2*
translations $\varepsilon : E <_{CF.eq} (\mathbf{a}, \mathbf{b}, \mathbf{g}, \mathbf{f}) : \uparrow_C \mapsto_{C\alpha} \mathfrak{C} \equiv$
 CONST *is-cat-equalizer-2* $\alpha \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \mathfrak{C} E \varepsilon$

locale *is-cat-coequalizer-2* =
is-cat-colimit
 $\alpha \langle \uparrow_C \mathbf{b}_{PL2} \mathbf{a}_{PL2} \mathbf{f}_{PL} \mathbf{g}_{PL} \rangle \mathfrak{C} \langle \uparrow \rightarrow \uparrow_{CF} \mathfrak{C} \mathbf{b}_{PL2} \mathbf{a}_{PL2} \mathbf{f}_{PL} \mathbf{g}_{PL} \mathbf{b} \mathbf{a} \mathbf{f} \mathbf{g} \rangle E \varepsilon$
for $\alpha \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \mathfrak{C} E \varepsilon +$
assumes *cat-coeq-g*[*cat-lim-cs-intros*]: $\mathbf{g} : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$
and *cat-coeq-f*[*cat-lim-cs-intros*]: $\mathbf{f} : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$

syntax *is-cat-coequalizer-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 (⟨(-) / '(-,-,-)' >_{CF.coeq} - / $\uparrow_C \mapsto_{C1}$ -)⟩ [51, 51, 51, 51, 51, 51] 51)
syntax-consts *is-cat-coequalizer-2* \equiv *is-cat-coequalizer-2*
translations $\varepsilon : (\mathbf{a}, \mathbf{b}, \mathbf{g}, \mathbf{f}) >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C} \equiv$
 CONST *is-cat-coequalizer-2* $\alpha \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \mathfrak{C} E \varepsilon$

Rules.

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

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

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

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

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

and $\mathbf{b}' = \mathbf{b}$
and $\mathbf{g}' = \mathbf{g}$
and $\mathbf{f}' = \mathbf{f}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : (\mathbf{a}', \mathbf{b}', \mathbf{g}', \mathbf{f}') >_{CF.coeq} E' : \uparrow_C \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle proof \rangle$

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

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

Helper lemmas.

lemma *cat-eq-F'-helper*:
 $(\lambda f \in_o set \{f_{PL}, g_{PL}\}. (f = g_{PL} ? g : f)) =$
 $(\lambda f \in_o set \{f_{PL}, g_{PL}\}. (f = f_{PL} ? f : g))$
 $\langle proof \rangle$

Elementary properties.

sublocale *is-cat-equalizer-2* \subseteq *cf-parallel-2* α \mathbf{a}_{PL2} \mathbf{b}_{PL2} \mathbf{g}_{PL} \mathbf{f}_{PL} \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} \mathfrak{C}
 $\langle proof \rangle$

sublocale *is-cat-coequalizer-2* \subseteq *cf-parallel-2* α \mathbf{b}_{PL2} \mathbf{a}_{PL2} \mathbf{f}_{PL} \mathbf{g}_{PL} \mathbf{b} \mathbf{a} \mathbf{f} \mathbf{g} \mathfrak{C}
 $\langle proof \rangle$

lemma (**in** *is-cat-equalizer-2*) *cat-equalizer-2-is-cat-equalizer*:
 $\varepsilon :$
 $E <_{CF.eq} (\mathbf{a}, \mathbf{b}, set \{g_{PL}, f_{PL}\}, (\lambda f \in_o set \{g_{PL}, f_{PL}\}. (f = f_{PL} ? f : g))) :$
 $\uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma (**in** *is-cat-coequalizer-2*) *cat-coequalizer-2-is-cat-coequalizer*:
 $\varepsilon :$
 $(\mathbf{a}, \mathbf{b}, set \{g_{PL}, f_{PL}\}, (\lambda f \in_o set \{g_{PL}, f_{PL}\}. (f = f_{PL} ? f : g))) >_{CF.coeq} E :$
 $\uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma *cat-equalizer-is-cat-equalizer-2*:
assumes $\varepsilon :$
 $E <_{CF.eq} (\mathbf{a}, \mathbf{b}, set \{g_{PL}, f_{PL}\}, (\lambda f \in_o set \{g_{PL}, f_{PL}\}. (f = f_{PL} ? f : g))) :$
 $\uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
shows $\varepsilon : E <_{CF.eq} (\mathbf{a}, \mathbf{b}, \mathbf{g}, \mathbf{f}) : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
 $\langle proof \rangle$

lemma *cat-coequalizer-is-cat-coequalizer-2*:
assumes $\varepsilon :$
 $(\mathbf{a}, \mathbf{b}, set \{g_{PL}, f_{PL}\}, (\lambda f \in_o set \{g_{PL}, f_{PL}\}. (f = f_{PL} ? f : g))) >_{CF.coeq} E :$
 $\uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
shows $\varepsilon : (\mathbf{a}, \mathbf{b}, \mathbf{g}, \mathbf{f}) >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$
 $\langle proof \rangle$

Duality.

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

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

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

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

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

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

Further properties.

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

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

lemma *cat-cone-cf-par-2-eps-NTMap-app*:

assumes $\varepsilon :$
 $E <_{CF.cone} \uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathbf{a}_{PL2} \mathbf{b}_{PL2} \mathbf{g}_{PL} \mathbf{f}_{PL} \mathbf{a} \mathbf{b} \mathbf{g} \mathbf{f} : \uparrow\uparrow_C \mathbf{a}_{PL2} \mathbf{b}_{PL2} \mathbf{g}_{PL} \mathbf{f}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathbf{g} : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$
and $\mathbf{f} : \mathbf{a} \mapsto_{\mathfrak{C}} \mathbf{b}$
shows
 $\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \mathbf{g} \circ_{A\mathfrak{C}} \varepsilon(\text{NTMap})(\mathbf{a}_{PL2})$
 $\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \mathbf{f} \circ_{A\mathfrak{C}} \varepsilon(\text{NTMap})(\mathbf{a}_{PL2})$
<proof>

lemma *cat-cocone-cf-par-2-eps-NTMap-app*:

assumes $\varepsilon :$
 $\uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathbf{b}_{PL2} \mathbf{a}_{PL2} \mathbf{f}_{PL} \mathbf{g}_{PL} \mathbf{b} \mathbf{a} \mathbf{f} \mathbf{g} >_{CF.cocone} E :$
 $\uparrow\uparrow_C \mathbf{b}_{PL2} \mathbf{a}_{PL2} \mathbf{f}_{PL} \mathbf{g}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathbf{g} : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$
and $\mathbf{f} : \mathbf{b} \mapsto_{\mathfrak{C}} \mathbf{a}$
shows
 $\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \varepsilon(\text{NTMap})(\mathbf{a}_{PL2}) \circ_{A\mathfrak{C}} \mathbf{g}$
 $\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \varepsilon(\text{NTMap})(\mathbf{a}_{PL2}) \circ_{A\mathfrak{C}} \mathbf{f}$
<proof>

lemma (in *is-cat-equalizer-2*) *cat-eq-2-eps-NTMap-app*:

$\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \mathbf{g} \circ_{A\mathfrak{C}} \varepsilon(\text{NTMap})(\mathbf{a}_{PL2})$
 $\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \mathbf{f} \circ_{A\mathfrak{C}} \varepsilon(\text{NTMap})(\mathbf{a}_{PL2})$
<proof>

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-eps-NTMap-app*:

$\varepsilon(\text{NTMap})(\mathbf{b}_{PL2}) = \varepsilon(\text{NTMap})(\mathbf{a}_{PL2}) \circ_{A\mathfrak{C}} \mathbf{g}$

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

lemma (in *is-cat-equalizer-2*) *cat-eq-2-Comp-eq*:

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

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-Comp-eq*:

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

7.2.2 Universal property

lemma *is-cat-equalizer-2I'*:

assumes ε :

$E <_{CF.cone} \uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mathit{a} \mathit{b} \mathit{g} \mathit{f} : \uparrow\uparrow_C \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mapsto \rightarrow_{C\alpha} \mathfrak{C}$
 and $g : \mathit{a} \mapsto_{\mathfrak{C}} \mathit{b}$
 and $f : \mathit{a} \mapsto_{\mathfrak{C}} \mathit{b}$
 and $\wedge \varepsilon' E'. \varepsilon'$:

$E' <_{CF.cone} \uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mathit{a} \mathit{b} \mathit{g} \mathit{f} :$
 $\uparrow\uparrow_C \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mapsto \rightarrow_{C\alpha} \mathfrak{C} \implies$
 $\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\mathit{NTMap})(\mathit{a}_{PL2}) = \varepsilon(\mathit{NTMap})(\mathit{a}_{PL2}) \circ_{A\mathfrak{C}} f'$

shows $\varepsilon : E <_{CF.eq} (\mathit{a}, \mathit{b}, \mathit{g}, \mathit{f}) : \uparrow\uparrow_C \mapsto \rightarrow_{C\alpha} \mathfrak{C}$

⟨proof⟩

lemma *is-cat-coequalizer-2I'*:

assumes ε :

$\uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathit{b}_{PL2} \mathit{a}_{PL2} \mathit{f}_{PL} \mathit{g}_{PL} \mathit{b} \mathit{a} \mathit{f} \mathit{g} >_{CF.cocone} E :$
 $\uparrow\uparrow_C \mathit{b}_{PL2} \mathit{a}_{PL2} \mathit{f}_{PL} \mathit{g}_{PL} \mapsto \rightarrow_{C\alpha} \mathfrak{C}$
 and $g : \mathit{b} \mapsto_{\mathfrak{C}} \mathit{a}$
 and $f : \mathit{b} \mapsto_{\mathfrak{C}} \mathit{a}$
 and $\wedge \varepsilon' E'. \varepsilon'$:

$\uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathit{b}_{PL2} \mathit{a}_{PL2} \mathit{f}_{PL} \mathit{g}_{PL} \mathit{b} \mathit{a} \mathit{f} \mathit{g} >_{CF.cocone} E' :$
 $\uparrow\uparrow_C \mathit{b}_{PL2} \mathit{a}_{PL2} \mathit{f}_{PL} \mathit{g}_{PL} \mapsto \rightarrow_{C\alpha} \mathfrak{C} \implies$
 $\exists ! f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(\mathit{NTMap})(\mathit{a}_{PL2}) = f' \circ_{A\mathfrak{C}} \varepsilon(\mathit{NTMap})(\mathit{a}_{PL2})$

shows $\varepsilon : (\mathit{a}, \mathit{b}, \mathit{g}, \mathit{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto \rightarrow_{C\alpha} \mathfrak{C}$

⟨proof⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique-cone*:

assumes ε' :

$E' <_{CF.cone} \uparrow\uparrow \rightarrow \uparrow\uparrow_{CF} \mathfrak{C} \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mathit{a} \mathit{b} \mathit{g} \mathit{f} :$
 $\uparrow\uparrow_C \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL} \mapsto \rightarrow_{C\alpha} \mathfrak{C}$

shows $\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\mathit{NTMap})(\mathit{a}_{PL2}) = \varepsilon(\mathit{NTMap})(\mathit{a}_{PL2}) \circ_{A\mathfrak{C}} f'$

⟨proof⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique*:

assumes $\varepsilon' : E' <_{CF.eq} (\mathit{a}, \mathit{b}, \mathit{g}, \mathit{f}) : \uparrow\uparrow_C \mapsto \rightarrow_{C\alpha} \mathfrak{C}$

shows

$\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon' = \varepsilon \cdot_{NTCF} \mathit{ntcf-const} (\uparrow\uparrow_C \mathit{a}_{PL2} \mathit{b}_{PL2} \mathit{g}_{PL} \mathit{f}_{PL}) \mathfrak{C} f'$

⟨proof⟩

lemma (in *is-cat-equalizer-2*) *cat-eq-2-unique'*:

assumes $\varepsilon' : E' <_{CF.eq} (\mathit{a}, \mathit{b}, \mathit{g}, \mathit{f}) : \uparrow\uparrow_C \mapsto \rightarrow_{C\alpha} \mathfrak{C}$

shows $\exists ! f'. f' : E' \mapsto_{\mathfrak{C}} E \wedge \varepsilon'(\mathit{NTMap})(\mathit{a}_{PL2}) = \varepsilon(\mathit{NTMap})(\mathit{a}_{PL2}) \circ_{A\mathfrak{C}} f'$

⟨proof⟩

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-unique-cocone*:

assumes ε' :

$\uparrow\uparrow\rightarrow\uparrow\uparrow_{CF} \mathfrak{C} \mathfrak{b}_{PL2} \mathfrak{a}_{PL2} \mathfrak{f}_{PL} \mathfrak{g}_{PL} \mathfrak{b} \mathfrak{a} \mathfrak{f} \mathfrak{g} >_{CF.cocone} E'$:

$\uparrow\uparrow_C \mathfrak{b}_{PL2} \mathfrak{a}_{PL2} \mathfrak{f}_{PL} \mathfrak{g}_{PL} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows $\exists!f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) = f' \circ_{A\mathfrak{C}} \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle)$

<proof>

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-unique*:

assumes $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows $\exists!f'$.

$f' : E \mapsto_{\mathfrak{C}} E' \wedge$

$\varepsilon' = ntcf-const (\uparrow\uparrow_C \mathfrak{b}_{PL2} \mathfrak{a}_{PL2} \mathfrak{f}_{PL} \mathfrak{g}_{PL}) \mathfrak{C} f' \cdot_{NTCF} \varepsilon$

<proof>

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-unique'*:

assumes $\varepsilon' : (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) >_{CF.coeq} E' : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows $\exists!f'. f' : E \mapsto_{\mathfrak{C}} E' \wedge \varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) = f' \circ_{A\mathfrak{C}} \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle)$

<proof>

lemma *cat-equalizer-2-ex-is-iso-arr*:

assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$

and $\varepsilon' = \varepsilon \cdot_{NTCF} ntcf-const (\uparrow\uparrow_C \mathfrak{a}_{PL2} \mathfrak{b}_{PL2} \mathfrak{g}_{PL} \mathfrak{f}_{PL}) \mathfrak{C} f$

<proof>

lemma *cat-equalizer-2-ex-is-iso-arr'*:

assumes $\varepsilon : E <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon' : E' <_{CF.eq} (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

obtains f **where** $f : E' \mapsto_{iso\mathfrak{C}} E$

and $\varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) = \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) \circ_{A\mathfrak{C}} f$

and $\varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{b}_{PL2} \rangle) = \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{b}_{PL2} \rangle) \circ_{A\mathfrak{C}} f$

<proof>

lemma *cat-coequalizer-2-ex-is-iso-arr*:

assumes $\varepsilon : (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

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

obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$

and $\varepsilon' = ntcf-const (\uparrow\uparrow_C \mathfrak{b}_{PL2} \mathfrak{a}_{PL2} \mathfrak{f}_{PL} \mathfrak{g}_{PL}) \mathfrak{C} f \cdot_{NTCF} \varepsilon$

<proof>

lemma *cat-coequalizer-2-ex-is-iso-arr'*:

assumes $\varepsilon : (\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}) >_{CF.coeq} E : \uparrow\uparrow_C \mapsto\mapsto_{C\alpha} \mathfrak{C}$

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

obtains f **where** $f : E \mapsto_{iso\mathfrak{C}} E'$

and $\varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) = f \circ_{A\mathfrak{C}} \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle)$

and $\varepsilon'(\langle NTMap \rangle)(\langle \mathfrak{b}_{PL2} \rangle) = f \circ_{A\mathfrak{C}} \varepsilon(\langle NTMap \rangle)(\langle \mathfrak{b}_{PL2} \rangle)$

<proof>

7.2.3 Further properties

lemma (in *is-cat-equalizer-2*) *cat-eq-2-is-monic-arr*:

$\varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) : E \mapsto_{mon\mathfrak{C}} \mathfrak{a}$

<proof>

lemma (in *is-cat-coequalizer-2*) *cat-coeq-2-is-epic-arr*:

$\varepsilon(\langle NTMap \rangle)(\langle \mathfrak{a}_{PL2} \rangle) : \mathfrak{a} \mapsto_{epi\mathfrak{C}} E$

<proof>

7.3 Equalizer cone

7.3.1 Definition and elementary properties

definition *ntcf-equalizer-base* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where *ntcf-equalizer-base* $\mathfrak{C} \ a \ b \ g \ f \ E \ e =$

[
 $(\lambda x \in \circ \uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} (Obj). \ e \ x),$
cf-const $(\uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL}) \ \mathfrak{C} \ E,$
 $\uparrow \uparrow \rightarrow \uparrow \uparrow_{CF} \ \mathfrak{C} \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} \ a \ b \ g \ f,$
 $\uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL},$
 \mathfrak{C}
]_o

Components.

lemma *ntcf-equalizer-base-components*:

shows *ntcf-equalizer-base* $\mathfrak{C} \ a \ b \ g \ f \ E \ e (NTMap) =$
 $(\lambda x \in \circ \uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} (Obj). \ e \ x)$
and [*cat-lim-cs-simps*]: *ntcf-equalizer-base* $\mathfrak{C} \ a \ b \ g \ f \ E \ e (NTDom) =$
 $cf\text{-const} \ (\uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL}) \ \mathfrak{C} \ E$
and [*cat-lim-cs-simps*]: *ntcf-equalizer-base* $\mathfrak{C} \ a \ b \ g \ f \ E \ e (NTCod) =$
 $\uparrow \uparrow \rightarrow \uparrow \uparrow_{CF} \ \mathfrak{C} \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} \ a \ b \ g \ f$
and [*cat-lim-cs-simps*]:
ntcf-equalizer-base $\mathfrak{C} \ a \ b \ g \ f \ E \ e (NTDGDom) = \uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL}$
and [*cat-lim-cs-simps*]:
ntcf-equalizer-base $\mathfrak{C} \ a \ b \ g \ f \ E \ e (NTDGCod) = \mathfrak{C}$
 {*proof*}

7.3.2 Natural transformation map

mk-VLambda *ntcf-equalizer-base-components*(1)

|*vsu ntcf-equalizer-base-NTMap-vsuv*[*cat-lim-cs-intros*]|
 |*vdomain ntcf-equalizer-base-NTMap-vdomain*[*cat-lim-cs-simps*]|
 |*app ntcf-equalizer-base-NTMap-app*[*cat-lim-cs-simps*]|

7.3.3 Equalizer cone is a cone

lemma (in *category*) *cat-ntcf-equalizer-base-is-cat-cone*:

assumes $e \ a_{PL2} : E \mapsto_{\mathfrak{C}} \ a$

and $e \ b_{PL2} : E \mapsto_{\mathfrak{C}} \ b$

and $e \ b_{PL2} = g \circ_{A\mathfrak{C}} \ e \ a_{PL2}$

and $e \ b_{PL2} = f \circ_{A\mathfrak{C}} \ e \ a_{PL2}$

and $g : a \mapsto_{\mathfrak{C}} \ b$

and $f : a \mapsto_{\mathfrak{C}} \ b$

shows *ntcf-equalizer-base* $\mathfrak{C} \ a \ b \ g \ f \ E \ e :$

$E <_{CF.cone} \ \uparrow \uparrow \rightarrow \uparrow \uparrow_{CF} \ \mathfrak{C} \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} \ a \ b \ g \ f :$

$\uparrow \uparrow_C \ a_{PL2} \ b_{PL2} \ g_{PL} \ f_{PL} \mapsto \mapsto_{C\alpha} \ \mathfrak{C}$

{*proof*}

8 Pointed arrows and natural transformations

8.1 Pointed arrow

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

8.1.1 Definition and elementary properties

See Chapter III-2 in [9].

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

Components.

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

8.1.2 Arrow value

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

8.1.3 Pointed arrow is an arrow in *Set*

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

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

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

8.1.4 Further properties

lemma $ntcf-paa-injective[cat-cs-simps]$:
 $ntcf-paa \ a \ A \ b = ntcf-paa \ a \ A \ c \longleftrightarrow b = c$
 $\langle proof \rangle$

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

lemma (in \mathcal{Z}) *ntcf-paa-ArrVal*:
assumes $F : \text{set } \{\mathbf{a}\} \mapsto_{\text{cat-Set } \alpha} X$ **and** $a = \mathbf{a}$
shows *ntcf-paa* \mathbf{a} X ($F(\text{ArrVal})(\mathbf{a})$) = F
<proof>

lemma (in \mathcal{Z}) *ntcf-paa-Comp-right*[*cat-cs-simps*]:
assumes $F : A \mapsto_{\text{cat-Set } \alpha} B$
and $\mathbf{a} \in_{\circ} \text{cat-Set } \alpha(\text{Obj})$
and $a \in_{\circ} A$
shows $F \circ_A \text{cat-Set } \alpha$ *ntcf-paa* \mathbf{a} A $a = \text{ntcf-paa } \mathbf{a}$ B ($F(\text{ArrVal})(\mathbf{a})$)
<proof>

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

8.2 Pointed natural transformation

8.2.1 Definition and elementary properties

See Chapter III-2 in [9].

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

where *ntcf-pointed* α $\mathbf{a} =$

[
 (
 $\lambda x \in_{\circ} \text{cat-Set } \alpha(\text{Obj}).$
 [
 $(\lambda f \in_{\circ} \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) x. f(\text{ArrVal})(\mathbf{a})),$
 $\text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) x,$
 x
]_o
),
 $\text{Hom}_{O.C\alpha} \text{cat-Set } \alpha(\text{set } \{\mathbf{a}\}, -),$
 $\text{cf-id } (\text{cat-Set } \alpha),$
 $\text{cat-Set } \alpha,$
 $\text{cat-Set } \alpha$
]_o

Components.

lemma *ntcf-pointed-components*:

shows *ntcf-pointed* α $\mathbf{a}(\text{NTMap}) =$

(
 $\lambda x \in_{\circ} \text{cat-Set } \alpha(\text{Obj}).$
 [
 $(\lambda f \in_{\circ} \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) x. f(\text{ArrVal})(\mathbf{a})),$
 $\text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) x,$
 x
]_o
)

and [*cat-cs-simps*]: *ntcf-pointed* α $\mathbf{a}(\text{NTDom}) = \text{Hom}_{O.C\alpha} \text{cat-Set } \alpha(\text{set } \{\mathbf{a}\}, -)$

and [*cat-cs-simps*]: *ntcf-pointed* α $\mathbf{a}(\text{NTCod}) = \text{cf-id } (\text{cat-Set } \alpha)$

and [*cat-cs-simps*]: *ntcf-pointed* α $\mathbf{a}(\text{NTDGDom}) = \text{cat-Set } \alpha$

and [*cat-cs-simps*]: *ntcf-pointed* α $\mathbf{a}(\text{NTDGCod}) = \text{cat-Set } \alpha$

<proof>

8.2.2 Natural transformation map

mk-VLambda *ntcf-pointed-components*(I)

|*vsu ntcf-pointed-NTMap-vsuv*[*cat-cs-intros*]|

|*vdomain ntcf-pointed-NTMap-vdomain*[*cat-cs-simps*]|

$[app\ ntcf\text{-pointed}\text{-}NTMap\text{-}app']$

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

lemma (in \mathcal{Z}) *ntcf-pointed-NTMap-app-is-iso-arr*:
assumes $\mathbf{a} \in_{\circ} cat\text{-}Set\ \alpha(\mathit{Obj})$ **and** $X \in_{\circ} cat\text{-}Set\ \alpha(\mathit{Obj})$
shows *ntcf-pointed* $\alpha\ \mathbf{a}(NTMap)(X) :$
 $Hom\ (cat\text{-}Set\ \alpha)\ (set\ \{\mathbf{a}\})\ X \mapsto_{iso} cat\text{-}Set\ \alpha\ X$
 $\langle proof \rangle$

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

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

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

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

8.2.3 Pointed natural transformation is a natural isomorphism

lemma (in \mathcal{Z}) *ntcf-pointed-is-iso-ntcf*:
assumes $\mathbf{a} \in_{\circ} cat\text{-}Set\ \alpha(\mathit{Obj})$
shows *ntcf-pointed* $\alpha\ \mathbf{a} :$
 $Hom_{O.C\alpha}\ cat\text{-}Set\ \alpha(set\ \{\mathbf{a}\}, -) \mapsto_{CF.iso}\ cf\text{-}id\ (cat\text{-}Set\ \alpha) :$
 $cat\text{-}Set\ \alpha \mapsto_{C\alpha}\ cat\text{-}Set\ \alpha$
 $\langle proof \rangle$

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

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

8.3 Inverse pointed natural transformation

8.3.1 Definition and elementary properties

See Chapter III-2 in [9].

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

(
 $\lambda X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$.
 $[(\lambda x \in_{\circ} X. \text{ntcf-paa } \mathbf{a} \ X \ x), X, \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) \ X]$.
),
 $\text{cf-id } (\text{cat-Set } \alpha)$,
 $\text{Hom}_{O.C} \alpha \text{cat-Set } \alpha (\text{set } \{\mathbf{a}\}, -)$,
 $\text{cat-Set } \alpha$,
 $\text{cat-Set } \alpha$
] $_{\circ}$

Components.

lemma *ntcf-pointed-inv-components*:

shows $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTMap} \rangle =$

(
 $\lambda X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$.
 $[(\lambda x \in_{\circ} X. \text{ntcf-paa } \mathbf{a} \ X \ x), X, \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) \ X]$.
)

and $[\text{cat-cs-simps}]$: $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTDom} \rangle = \text{cf-id } (\text{cat-Set } \alpha)$

and $[\text{cat-cs-simps}]$:

$\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTCod} \rangle = \text{Hom}_{O.C} \alpha \text{cat-Set } \alpha (\text{set } \{\mathbf{a}\}, -)$

and $[\text{cat-cs-simps}]$: $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTDGDom} \rangle = \text{cat-Set } \alpha$

and $[\text{cat-cs-simps}]$: $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTDGCod} \rangle = \text{cat-Set } \alpha$

$\langle \text{proof} \rangle$

8.3.2 Natural transformation map

mk-VLambda *ntcf-pointed-inv-components(1)*

$[\text{vsu } \text{ntcf-pointed-inv-NTMap-vsuv} [\text{cat-cs-intros}]]$

$[\text{vdomain } \text{ntcf-pointed-inv-NTMap-vdomain} [\text{cat-cs-simps}]]$

$[\text{app } \text{ntcf-pointed-inv-NTMap-app}']$

lemma (in \mathcal{Z}) *ntcf-pointed-inv-NTMap-app-ArrVal-app* $[\text{cat-cs-simps}]$:

assumes $X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$ **and** $x \in_{\circ} X$

shows $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTMap} \rangle \langle X \rangle \langle \text{ArrVal} \rangle \langle x \rangle = \text{ntcf-paa } \mathbf{a} \ X \ x$

$\langle \text{proof} \rangle$

lemma (in \mathcal{Z}) *ntcf-pointed-inv-NTMap-app-is-arr*:

assumes $\mathbf{a} \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$ **and** $X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$

shows $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTMap} \rangle \langle X \rangle$:

$X \mapsto_{\text{cat-Set } \alpha} \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) \ X$

$\langle \text{proof} \rangle$

lemma (in \mathcal{Z}) *ntcf-pointed-inv-NTMap-app-is-arr'* $[\text{cat-cs-intros}]$:

assumes $\mathbf{a} \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$

and $X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$

and $A' = X$

and $B' = \text{Hom } (\text{cat-Set } \alpha) (\text{set } \{\mathbf{a}\}) \ X$

and $\mathcal{C}' = \text{cat-Set } \alpha$

shows $\text{ntcf-pointed-inv } \alpha \ \mathbf{a} \langle \text{NTMap} \rangle \langle X \rangle : A' \mapsto_{\mathcal{C}'} B'$

$\langle \text{proof} \rangle$

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

lemma (in \mathcal{Z}) *is-inverse-ntcf-pointed-inv-NTMap-app*:

assumes $\mathbf{a} \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$ **and** $X \in_{\circ} \text{cat-Set } \alpha \langle \text{Obj} \rangle$

shows

is-inverse

$(\text{cat-Set } \alpha)$

$(ntcf\text{-pointed}\text{-inv } \alpha \mathbf{a} \langle NTMap \rangle \langle X \rangle)$
 $(ntcf\text{-pointed } \alpha \mathbf{a} \langle NTMap \rangle \langle X \rangle)$
 $(is \text{ is-inverse } (cat\text{-Set } \alpha) \text{ ?bwd } \text{ ?fwd})$
 $\langle proof \rangle$

8.3.3 Inverse pointed natural transformation is a natural isomorphism

lemma (in \mathcal{Z}) *ntcf-pointed-inv-is-ntcf*:
assumes $\mathbf{a} \in_{\circ} cat\text{-Set } \alpha \langle Obj \rangle$
shows *ntcf-pointed-inv* $\alpha \mathbf{a}$:
 $cf\text{-id } (cat\text{-Set } \alpha) \mapsto_{CF} Hom_{O.C\alpha} cat\text{-Set } \alpha (set \{ \mathbf{a} \}, -) :$
 $cat\text{-Set } \alpha \mapsto_{C\alpha} cat\text{-Set } \alpha$
 $\langle proof \rangle$

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

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

lemma (in \mathcal{Z}) *inv-ntcf-ntcf-pointed*[*cat-cs-simps*]:
assumes $\mathbf{a} \in_{\circ} cat\text{-Set } \alpha \langle Obj \rangle$
shows *inv-ntcf* (*ntcf-pointed* $\alpha \mathbf{a}$) = *ntcf-pointed-inv* $\alpha \mathbf{a}$
 $\langle proof \rangle$

lemma (in \mathcal{Z}) *inv-ntcf-ntcf-pointed-inv*[*cat-cs-simps*]:
assumes $\mathbf{a} \in_{\circ} cat\text{-Set } \alpha \langle Obj \rangle$
shows *inv-ntcf* (*ntcf-pointed-inv* $\alpha \mathbf{a}$) = *ntcf-pointed* $\alpha \mathbf{a}$
 $\langle proof \rangle$

lemma (in \mathcal{Z}) *ntcf-pointed-inv-is-iso-ntcf*:
assumes $\mathbf{a} \in_{\circ} cat\text{-Set } \alpha \langle Obj \rangle$
shows *ntcf-pointed-inv* $\alpha \mathbf{a}$:
 $cf\text{-id } (cat\text{-Set } \alpha) \mapsto_{CF.is\circ} Hom_{O.C\alpha} cat\text{-Set } \alpha (set \{ \mathbf{a} \}, -) :$
 $cat\text{-Set } \alpha \mapsto_{C\alpha} cat\text{-Set } \alpha$
 $\langle proof \rangle$

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

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

9 Representable and corepresentable functors

9.1 Representable and corepresentable functors

9.1.1 Definitions and elementary properties

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

definition *cat-representation* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *cat-representation* $\alpha \mathfrak{F} c \psi \leftrightarrow$

$c \in_{\circ} \mathfrak{F}(\text{HomDom})(\text{Obj}) \wedge$

$\psi : \text{Hom}_{O.C\alpha} \mathfrak{F}(\text{HomDom})(c, -) \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{F}(\text{HomDom}) \mapsto_{C\alpha} \text{cat-Set } \alpha$

definition *cat-corepresentation* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *cat-corepresentation* $\alpha \mathfrak{F} c \psi \leftrightarrow$

$c \in_{\circ} \mathfrak{F}(\text{HomDom})(\text{Obj}) \wedge$

$\psi : \text{Hom}_{O.C\alpha} \text{op-cat } (\mathfrak{F}(\text{HomDom}))(-, c) \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{F}(\text{HomDom}) \mapsto_{C\alpha} \text{cat-Set } \alpha$

Rules.

context

fixes $\alpha \mathfrak{C} \mathfrak{F}$

assumes $\mathfrak{F} : \mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

begin

interpretation $\mathfrak{F} : \text{is-functor } \alpha \mathfrak{C} \langle \text{cat-Set } \alpha \rangle \mathfrak{F} \langle \text{proof} \rangle$

mk-ide rf *cat-representation-def*[**where** $\alpha = \alpha$ **and** $\mathfrak{F} = \mathfrak{F}$, *unfolded cat-cs-simps*]

|*intro cat-representationI*|

|*dest cat-representationD'*|

|*elim cat-representationE'*|

end

lemmas *cat-representationD*[*dest*] = *cat-representationD'*[*rotated*]

and *cat-representationE*[*elim*] = *cat-representationE'*[*rotated*]

lemma *cat-corepresentationI*:

assumes *category* $\alpha \mathfrak{C}$

and $\mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

and $c \in_{\circ} \mathfrak{C}(\text{Obj})$

and $\psi : \text{Hom}_{O.C\alpha} \mathfrak{C}(-, c) \mapsto_{CF.iso} \mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

shows *cat-corepresentation* $\alpha \mathfrak{F} c \psi$

<proof>

lemma *cat-corepresentationD*:

assumes *cat-corepresentation* $\alpha \mathfrak{F} c \psi$

and *category* $\alpha \mathfrak{C}$

and $\mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

shows $c \in_{\circ} \mathfrak{C}(\text{Obj})$

and $\psi : \text{Hom}_{O.C\alpha} \mathfrak{C}(-, c) \mapsto_{CF.iso} \mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

<proof>

lemma *cat-corepresentationE*:

assumes *cat-corepresentation* $\alpha \mathfrak{F} c \psi$

and *category* $\alpha \mathfrak{C}$

and $\mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

obtains $c \in_{\circ} \mathfrak{C}(\text{Obj})$

and $\psi : \text{Hom}_{O.C\alpha} \mathfrak{C}(-, c) \mapsto_{CF.iso} \mathfrak{F} : \text{op-cat } \mathfrak{C} \mapsto_{C\alpha} \text{cat-Set } \alpha$

<proof>

9.1.2 Representable functors and universal arrows

lemma *universal-arrow-of-if-cat-representation:*

— See Proposition 2 in Chapter III-2 in [9].

assumes $\mathfrak{K} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$

and *cat-representation* $\alpha \ \mathfrak{K} \ r \ \psi$

and $\mathfrak{a} \in_{\circ} \text{cat-Set } \alpha(\text{Obj})$

shows *universal-arrow-of*

$\mathfrak{K}(\text{set } \{\mathfrak{a}\}) \ r \ (\text{ntcf-paa } \mathfrak{a} \ (\mathfrak{K}(\text{ObjMap})(\downarrow r)) \ (\psi(\downarrow \text{NTMap})(\downarrow r)(\downarrow \text{ArrVal})(\downarrow \mathfrak{C}(\downarrow \text{CId})(\downarrow r))))$

<proof>

lemma *universal-arrow-of-if-cat-corepresentation:*

— See Proposition 2 in Chapter III-2 in [9].

assumes *category* $\alpha \ \mathfrak{C}$

and $\mathfrak{K} : \text{op-cat } \mathfrak{C} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$

and *cat-corepresentation* $\alpha \ \mathfrak{K} \ r \ \psi$

and $\mathfrak{a} \in_{\circ} \text{cat-Set } \alpha(\text{Obj})$

shows *universal-arrow-of*

$\mathfrak{K}(\text{set } \{\mathfrak{a}\}) \ r \ (\text{ntcf-paa } \mathfrak{a} \ (\mathfrak{K}(\text{ObjMap})(\downarrow r)) \ (\psi(\downarrow \text{NTMap})(\downarrow r)(\downarrow \text{ArrVal})(\downarrow \mathfrak{C}(\downarrow \text{CId})(\downarrow r))))$

<proof>

lemma *cat-representation-if-universal-arrow-of:*

— See Proposition 2 in Chapter III-2 in [9].

assumes $\mathfrak{K} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$

and $\mathfrak{a} \in_{\circ} \text{cat-Set } \alpha(\text{Obj})$

and *universal-arrow-of* $\mathfrak{K}(\text{set } \{\mathfrak{a}\}) \ r \ u$

shows *cat-representation* $\alpha \ \mathfrak{K} \ r \ (\text{Yoneda-arrow } \alpha \ \mathfrak{K} \ r \ (u(\downarrow \text{ArrVal})(\downarrow \mathfrak{a})))$

<proof>

lemma *cat-corepresentation-if-universal-arrow-of:*

— See Proposition 2 in Chapter III-2 in [9].

assumes *category* $\alpha \ \mathfrak{C}$

and $\mathfrak{K} : \text{op-cat } \mathfrak{C} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$

and $\mathfrak{a} \in_{\circ} \text{cat-Set } \alpha(\text{Obj})$

and *universal-arrow-of* $\mathfrak{K}(\text{set } \{\mathfrak{a}\}) \ r \ u$

shows *cat-corepresentation* $\alpha \ \mathfrak{K} \ r \ (\text{Yoneda-arrow } \alpha \ \mathfrak{K} \ r \ (u(\downarrow \text{ArrVal})(\downarrow \mathfrak{a})))$

<proof>

9.2 Limits and colimits as universal cones

lemma *is-tm-cat-limit-if-cat-corepresentation:*

— See Definition 3.1.5 in Section 3.1 in [14].

assumes $\mathfrak{F} : \mathfrak{J} \mapsto \mapsto_{C.\text{tm}\alpha} \mathfrak{C}$

and *cat-corepresentation* $\alpha \ (\text{tm-cf-Cone } \alpha \ \mathfrak{F}) \ r \ \psi$

(is *cat-corepresentation* $\alpha \ ?\text{Cone } r \ \psi)$

shows *ntcf-of-ntcf-arrow* $\mathfrak{J} \ \mathfrak{C} \ (\psi(\downarrow \text{NTMap})(\downarrow r)(\downarrow \text{ArrVal})(\downarrow \mathfrak{C}(\downarrow \text{CId})(\downarrow r))) :$

$r <_{CF.\text{tm.lim}} \mathfrak{F} : \mathfrak{J} \mapsto \mapsto_{C.\text{tm}\alpha} \mathfrak{C}$

(is *ntcf-of-ntcf-arrow* $\mathfrak{J} \ \mathfrak{C} \ ?\psi r 1r : r <_{CF.\text{tm.lim}} \mathfrak{F} : \mathfrak{J} \mapsto \mapsto_{C.\text{tm}\alpha} \mathfrak{C}$)

<proof>

lemma *cat-corepresentation-if-is-tm-cat-limit:*

— See Definition 3.1.5 in Section 3.1 in [14].

assumes $\psi : r <_{CF.\text{tm.lim}} \mathfrak{F} : \mathfrak{J} \mapsto \mapsto_{C.\text{tm}\alpha} \mathfrak{C}$

shows *cat-corepresentation*

$\alpha \ (\text{tm-cf-Cone } \alpha \ \mathfrak{F}) \ r \ (\text{Yoneda-arrow } \alpha \ (\text{tm-cf-Cone } \alpha \ \mathfrak{F}) \ r \ (\text{ntcf-arrow } \psi))$

(is *cat-corepresentation* $\alpha \ ?\text{Cone } r \ ?Y\psi)$

<proof>

10 Completeness and cocompleteness

10.1 Limits by products and equalizers

lemma *cat-limit-of-cat-prod-obj-and-cat-equalizer*:

— See Theorem 1 in Chapter V-2 in [9].

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$

and $\bigwedge a\ b\ g\ f. \llbracket f : a \mapsto_{\mathfrak{C}} b; g : a \mapsto_{\mathfrak{C}} b \rrbracket \implies$

$\exists E\ \varepsilon. \varepsilon : E <_{CF.eq} (a, b, g, f) : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\bigwedge A. tm\text{-}cf\text{-discrete}\ \alpha\ (\mathfrak{J}(Obj))\ A\ \mathfrak{C} \implies$

$\exists P\ \pi. \pi : P <_{CF.\Pi} A : \mathfrak{J}(Obj) \mapsto_{C\alpha} \mathfrak{C}$

and $\bigwedge A. tm\text{-}cf\text{-discrete}\ \alpha\ (\mathfrak{J}(Arr))\ A\ \mathfrak{C} \implies$

$\exists P\ \pi. \pi : P <_{CF.\Pi} A : \mathfrak{J}(Arr) \mapsto_{C\alpha} \mathfrak{C}$

obtains $r\ u$ **where** $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

<proof>

lemma *cat-colimit-of-cat-prod-obj-and-cat-coequalizer*:

— See Theorem 1 in Chapter V-2 in [9].

assumes $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$

and $\bigwedge a\ b\ g\ f. \llbracket f : b \mapsto_{\mathfrak{C}} a; g : b \mapsto_{\mathfrak{C}} a \rrbracket \implies$

$\exists E\ \varepsilon. \varepsilon : (a, b, g, f) >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\bigwedge A. tm\text{-}cf\text{-discrete}\ \alpha\ (\mathfrak{J}(Obj))\ A\ \mathfrak{C} \implies$

$\exists P\ \pi. \pi : A >_{CF.\Pi} P : \mathfrak{J}(Obj) \mapsto_{C\alpha} \mathfrak{C}$

and $\bigwedge A. tm\text{-}cf\text{-discrete}\ \alpha\ (\mathfrak{J}(Arr))\ A\ \mathfrak{C} \implies$

$\exists P\ \pi. \pi : A >_{CF.\Pi} P : \mathfrak{J}(Arr) \mapsto_{C\alpha} \mathfrak{C}$

obtains $r\ u$ **where** $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

<proof>

10.2 Small-complete and small-cocomplete category

10.2.1 Definition and elementary properties

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

assumes *cat-small-complete*:

$\bigwedge \mathfrak{F}\ \mathfrak{J}. \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \implies \exists u\ r. u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

locale *cat-small-cocomplete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ \mathfrak{C} +$

assumes *cat-small-cocomplete*:

$\bigwedge \mathfrak{F}\ \mathfrak{J}. \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C} \implies \exists u\ r. u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

Rules.

mk-ide rf *cat-small-complete-def*[*unfolded cat-small-complete-axioms-def*]

|*intro cat-small-completeI*|

|*dest cat-small-completeD*[*dest*]|

|*elim cat-small-completeE*[*elim*]|

lemma *cat-small-completeE'*[*elim*]:

assumes *cat-small-complete* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$

obtains $u\ r$ **where** $u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

<proof>

mk-ide rf *cat-small-cocomplete-def*[*unfolded cat-small-cocomplete-axioms-def*]

|*intro cat-small-cocompleteI*|

|*dest cat-small-cocompleteD*[*dest*]|

|*elim cat-small-cocompleteE*[*elim*]|

lemma *cat-small-cocompleteE'*[*elim*]:

assumes *cat-small-cocomplete* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{C}$

obtains $u\ r$ **where** $u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

<proof>

10.2.2 Duality

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

<proof>

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

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

<proof>

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

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

lemma (in *category*) *cat-small-complete-if-eq-and-obj-prod*:

— See Corollary 2 in Chapter V-2 in [9]

assumes $\wedge a \ b \ g \ f. \llbracket f : a \mapsto_{\mathfrak{C}} b; g : a \mapsto_{\mathfrak{C}} b \rrbracket \Longrightarrow$

$\exists E \ \varepsilon. \ \varepsilon : E <_{CF.eq} (a, b, g, f) : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\wedge A \ I. \ tm\text{-}cf\text{-discrete} \ \alpha \ I \ A \ \mathfrak{C} \Longrightarrow \exists P \ \pi. \ \pi : P <_{CF.\Pi} A : I \mapsto_{C\alpha} \mathfrak{C}$

shows *cat-small-complete* α \mathfrak{C}

<proof>

lemma (in *category*) *cat-small-cocomplete-if-eq-and-obj-prod*:

assumes $\wedge a \ b \ g \ f. \llbracket f : b \mapsto_{\mathfrak{C}} a; g : b \mapsto_{\mathfrak{C}} a \rrbracket \Longrightarrow$

$\exists E \ \varepsilon. \ \varepsilon : (a, b, g, f) >_{CF.coeq} E : \uparrow_C \mapsto_{C\alpha} \mathfrak{C}$

and $\wedge A \ I. \ tm\text{-}cf\text{-discrete} \ \alpha \ I \ A \ \mathfrak{C} \Longrightarrow \exists P \ \pi. \ \pi : A >_{CF.\Pi} P : I \mapsto_{C\alpha} \mathfrak{C}$

shows *cat-small-cocomplete* α \mathfrak{C}

<proof>

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

lemma (in *cat-small-complete*) *cat-sc-ex-obj-initial*:

— See Theorem 1 in Chapter V-6 in [9].

assumes $A \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

and $A \in_{\circ} \text{Vset} \ \alpha$

and $\wedge c. \ c \in_{\circ} \mathfrak{C}(\text{Obj}) \Longrightarrow \exists f \ a. \ a \in_{\circ} A \wedge f : a \mapsto_{\mathfrak{C}} c$

obtains z **where** *obj-initial* $\mathfrak{C} \ z$

<proof>

lemma (in *cat-small-cocomplete*) *cat-sc-ex-obj-terminal*:

— See Theorem 1 in Chapter V-6 in [9].

assumes $A \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

and $A \in_{\circ} \text{Vset} \ \alpha$

and $\wedge c. \ c \in_{\circ} \mathfrak{C}(\text{Obj}) \Longrightarrow \exists f \ a. \ a \in_{\circ} A \wedge f : c \mapsto_{\mathfrak{C}} a$

obtains z **where** *obj-terminal* $\mathfrak{C} \ z$

<proof>

10.2.5 Creation of limits, continuity and completeness

lemma

— See Theorem 2 in Chapter V-4 in [9].

assumes $\mathfrak{G} : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$

and *cat-small-complete* α \mathfrak{B}

and $\wedge \mathfrak{J} \ \mathfrak{J}. \ \mathfrak{J} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{A} \Longrightarrow \mathfrak{G} \circ_{CF} \mathfrak{J} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{B}$

and $\wedge \mathfrak{F} \mathfrak{J}. \mathfrak{F} : \mathfrak{J} \mapsto_{C.tm\alpha} \mathfrak{A} \implies cf\text{-creates-limits } \alpha \ \mathfrak{G} \ \mathfrak{F}$
shows *is-tm-cf-continuous-if-cf-creates-limits*: *is-tm-cf-continuous* $\alpha \ \mathfrak{G}$
and *cat-small-complete-if-cf-creates-limits*: *cat-small-complete* $\alpha \ \mathfrak{A}$
<proof>

10.3 Finite-complete and finite-cocomplete category

locale *cat-finite-complete* = *category* $\alpha \ \mathfrak{C}$ **for** $\alpha \ \mathfrak{C} +$
assumes *cat-finite-complete*:
 $\wedge \mathfrak{F} \mathfrak{J}. \llbracket \textit{finite-category } \alpha \ \mathfrak{J}; \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \rrbracket \implies$
 $\exists u \ r. u : r <_{CF.lim} \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

locale *cat-finite-cocomplete* = *category* $\alpha \ \mathfrak{C}$ **for** $\alpha \ \mathfrak{C} +$
assumes *cat-finite-cocomplete*:
 $\wedge \mathfrak{F} \mathfrak{J}. \llbracket \textit{finite-category } \alpha \ \mathfrak{J}; \mathfrak{F} : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C} \rrbracket \implies$
 $\exists u \ r. u : \mathfrak{F} >_{CF.colim} r : \mathfrak{J} \mapsto_{C\alpha} \mathfrak{C}$

Rules.

mk-ide rf *cat-finite-complete-def*[*unfolded cat-finite-complete-axioms-def*]
 $|intro \textit{cat-finite-complete}I|$
 $|dest \textit{cat-finite-complete}D[dest]|$
 $|elim \textit{cat-finite-complete}E[elim]|$

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

mk-ide rf *cat-finite-cocomplete-def*[*unfolded cat-finite-cocomplete-axioms-def*]
 $|intro \textit{cat-finite-cocomplete}I|$
 $|dest \textit{cat-finite-cocomplete}D[dest]|$
 $|elim \textit{cat-finite-cocomplete}E[elim]|$

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

Elementary properties.

sublocale *cat-small-complete* \subseteq *cat-finite-complete*
<proof>

sublocale *cat-small-cocomplete* \subseteq *cat-finite-cocomplete*
<proof>

11 Comma categories and universal constructions

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

lemma (in *is-functor*) *universal-arrow-of-if-obj-initial*:

— See Chapter III-1 in [9].

assumes $c \in_{\circ} \mathfrak{B}(\text{Obj})$ **and** *obj-initial* $(c \downarrow_{CF} \mathfrak{F}) [0, r, u]_{\circ}$

shows *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$

<proof>

lemma (in *is-functor*) *obj-initial-if-universal-arrow-of*:

— See Chapter III-1 in [9].

assumes *universal-arrow-of* $\mathfrak{F} \ c \ r \ u$

shows *obj-initial* $(c \downarrow_{CF} \mathfrak{F}) [0, r, u]_{\circ}$

<proof>

lemma (in *is-functor*) *universal-arrow-fo-if-obj-terminal*:

— See Chapter III-1 in [9].

assumes $c \in_{\circ} \mathfrak{B}(\text{Obj})$ **and** *obj-terminal* $(\mathfrak{F} \ {}_{CF}\downarrow \ c) [r, 0, u]_{\circ}$

shows *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$

<proof>

lemma (in *is-functor*) *obj-terminal-if-universal-arrow-fo*:

— See Chapter III-1 in [9].

assumes *universal-arrow-fo* $\mathfrak{F} \ c \ r \ u$

shows *obj-terminal* $(\mathfrak{F} \ {}_{CF}\downarrow \ c) [r, 0, u]_{\circ}$

<proof>

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

See Chapter V-6 in [9].

lemma *cf-obj-cf-comma-proj-creates-limits*:

assumes $\mathfrak{G} : \mathfrak{A} \mapsto_{CF} \mathfrak{X}$

and *is-tm-cf-continuous* $\alpha \ \mathfrak{G}$

and $x \in_{\circ} \mathfrak{X}(\text{Obj})$

and $\mathfrak{F} : \mathfrak{J} \mapsto_{CF} x \downarrow_{CF} \mathfrak{G}$

shows *cf-creates-limits* $\alpha \ (x \circ \square_{CF} \ \mathfrak{G}) \ \mathfrak{F}$

<proof>

12 Category *Set* and universal constructions

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

lemma (in \mathcal{Z}) *tm-cf-discrete-cat-Set-if-VLambda-in-Vset*:

assumes $VLambda\ I\ F\ \epsilon_o\ Vset\ \alpha$
shows $tm-cf-discrete\ \alpha\ I\ F\ (cat-Set\ \alpha)$

<proof>

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

12.2.1 Definition and elementary properties

definition *ntcf-Set-obj-prod* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where $ntcf-Set-obj-prod\ \alpha\ I\ F = ntcf-obj-prod-base$
 $(cat-Set\ \alpha)\ I\ F\ (\prod_{\circ} i \in_{\circ} I.\ F\ i)\ (\lambda i.\ vprojection-arrow\ I\ F\ i)$

definition *ntcf-Set-obj-coproduct* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where $ntcf-Set-obj-coproduct\ \alpha\ I\ F = ntcf-obj-coproduct-base$
 $(cat-Set\ \alpha)\ I\ F\ (\coprod_{\circ} i \in_{\circ} I.\ F\ i)\ (\lambda i.\ vcinjection-arrow\ I\ F\ i)$

Components.

lemma *ntcf-Set-obj-prod-components*:

shows $ntcf-Set-obj-prod\ \alpha\ I\ F(\backslash NTMap) =$
 $(\lambda i \in_{\circ} :_C\ I(\backslash Obj).\ vprojection-arrow\ I\ F\ i)$
and $ntcf-Set-obj-prod\ \alpha\ I\ F(\backslash NTDom) =$
 $cf-const\ (:_C\ I)\ (cat-Set\ \alpha)\ (\prod_{\circ} i \in_{\circ} I.\ F\ i)$
and $ntcf-Set-obj-prod\ \alpha\ I\ F(\backslash NTCod) = \text{!} \rightarrow : I\ F\ (cat-Set\ \alpha)$
and $ntcf-Set-obj-prod\ \alpha\ I\ F(\backslash NTGDGDom) = :_C\ I$
and $ntcf-Set-obj-prod\ \alpha\ I\ F(\backslash NTDGCod) = cat-Set\ \alpha$

<proof>

lemma *ntcf-Set-obj-coproduct-components*:

shows $ntcf-Set-obj-coproduct\ \alpha\ I\ F(\backslash NTMap) =$
 $(\lambda i \in_{\circ} :_C\ I(\backslash Obj).\ vcinjection-arrow\ I\ F\ i)$
and $ntcf-Set-obj-coproduct\ \alpha\ I\ F(\backslash NTDom) = \text{!} \rightarrow : I\ F\ (cat-Set\ \alpha)$
and $ntcf-Set-obj-coproduct\ \alpha\ I\ F(\backslash NTCod) =$
 $cf-const\ (:_C\ I)\ (cat-Set\ \alpha)\ (\coprod_{\circ} i \in_{\circ} I.\ F\ i)$
and $ntcf-Set-obj-coproduct\ \alpha\ I\ F(\backslash NTGDGDom) = :_C\ I$
and $ntcf-Set-obj-coproduct\ \alpha\ I\ F(\backslash NTDGCod) = cat-Set\ \alpha$

<proof>

12.2.2 Natural transformation map

mk-VLambda *ntcf-Set-obj-prod-components(1)*

$|vsv\ ntcf-Set-obj-prod-NTMap-vsuv[cat-cs-intros]|$
 $|vdomain\ ntcf-Set-obj-prod-NTMap-vdomain[cat-cs-simps]|$
 $|app\ ntcf-Set-obj-prod-NTMap-app[cat-cs-simps]|$

mk-VLambda *ntcf-Set-obj-coproduct-components(1)*

$|vsv\ ntcf-Set-obj-coproduct-NTMap-vsuv[cat-cs-intros]|$
 $|vdomain\ ntcf-Set-obj-coproduct-NTMap-vdomain[cat-cs-simps]|$
 $|app\ ntcf-Set-obj-coproduct-NTMap-app[cat-cs-simps]|$

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

lemma (in \mathcal{Z}) *tm-cf-discrete-ntcf-obj-prod-base-is-cat-obj-prod*:

— See Theorem 5.2 in Chapter Introduction in [6].

assumes $VLambda\ I\ F\ \epsilon_o\ Vset\ \alpha$
shows $ntcf\text{-}Set\text{-}obj\text{-}prod\ \alpha\ I\ F :$
 $(\prod_o i \in_o I. F\ i) <_{CF.\Pi}\ F : I \mapsto_{C\alpha} cat\text{-}Set\ \alpha$
<proof>

lemma (in \mathcal{Z}) *tm-cf-discrete-ntcf-obj-prod-base-is-tm-cat-obj-prod:*

— See Theorem 5.2 in Chapter Introduction in [6].

assumes $VLambda\ I\ F\ \epsilon_o\ Vset\ \alpha$
shows $ntcf\text{-}Set\text{-}obj\text{-}prod\ \alpha\ I\ F :$
 $(\prod_o i \in_o I. F\ i) <_{CF.tm.\Pi}\ F : I \mapsto_{C.tm\alpha} cat\text{-}Set\ \alpha$
<proof>

lemma (in \mathcal{Z}) *tm-cf-discrete-ntcf-obj-coproduct-base-is-cat-obj-coproduct:*

— See Theorem 5.2 in Chapter Introduction in [6].

assumes $VLambda\ I\ F\ \epsilon_o\ Vset\ \alpha$
shows $ntcf\text{-}Set\text{-}obj\text{-}coprod\ \alpha\ I\ F :$
 $F >_{CF.\Pi}\ (\prod_o i \in_o I. F\ i) : I \mapsto_{C\alpha} cat\text{-}Set\ \alpha$
<proof>

lemma (in \mathcal{Z}) *ntcf-Set-obj-coproduct-is-tm-cat-obj-coproduct:*

— See Theorem 5.2 in Chapter Introduction in [6].

assumes $VLambda\ I\ F\ \epsilon_o\ Vset\ \alpha$
shows $ntcf\text{-}Set\text{-}obj\text{-}coprod\ \alpha\ I\ F :$
 $F >_{CF.tm.\Pi}\ (\prod_o i \in_o I. F\ i) : I \mapsto_{C.tm\alpha} cat\text{-}Set\ \alpha$
<proof>

12.3 Equalizer for the category Set

12.3.1 Definition and elementary properties

abbreviation $ntcf\text{-}Set\text{-}equalizer\text{-}map :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $ntcf\text{-}Set\text{-}equalizer\text{-}map\ \alpha\ a\ g\ f\ i \equiv$

(
 $i = \mathbf{a}_{PL2} ?$
 $incl\text{-}Set\ (vequalizer\ a\ g\ f)\ a :$
 $g \circ_A cat\text{-}Set\ \alpha\ incl\text{-}Set\ (vequalizer\ a\ g\ f)\ a$
)

definition $ntcf\text{-}Set\text{-}equalizer :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $ntcf\text{-}Set\text{-}equalizer\ \alpha\ a\ b\ g\ f = ntcf\text{-}equalizer\text{-}base$

$(cat\text{-}Set\ \alpha)\ a\ b\ g\ f\ (vequalizer\ a\ g\ f)\ (ntcf\text{-}Set\text{-}equalizer\text{-}map\ \alpha\ a\ g\ f)$

Components.

context

fixes $a\ g\ f\ \alpha :: V$

begin

lemmas $ntcf\text{-}Set\text{-}equalizer\text{-}components =$

$ntcf\text{-}equalizer\text{-}base\text{-}components[$

where $\mathcal{C} = \langle cat\text{-}Set\ \alpha \rangle$

and $e = \langle ntcf\text{-}Set\text{-}equalizer\text{-}map\ \alpha\ a\ g\ f \rangle$

and $E = \langle vequalizer\ a\ g\ f \rangle$

and $\mathbf{a} = a$ **and** $\mathbf{g} = g$ **and** $\mathbf{f} = f,$

folded $ntcf\text{-}Set\text{-}equalizer\text{-}def$

$]$

end

12.3.2 Natural transformation map

mk-VLambda *ntcf-Set-equalizer-components(1)*
 $|vsv\ ntcf\text{-Set-equalizer-NTMap-vs}[cat\text{-Set-cs-intros}]|$
 $|vdomain\ ntcf\text{-Set-equalizer-NTMap-vdomain}[cat\text{-Set-cs-simps}]|$
 $|app\ ntcf\text{-Set-equalizer-NTMap-app}|$

lemma *ntcf-Set-equalizer-2-NTMap-app-a*[*cat-Set-cs-simps*]:
assumes $x = a_{PL2}$
shows
 $ntcf\text{-Set-equalizer}\ \alpha\ a\ b\ g\ f\ (NTMap)\ (x) =$
 $incl\text{-Set}\ (vequalizer\ a\ g\ f)\ a$
 $\langle proof \rangle$

lemma *ntcf-Set-equalizer-2-NTMap-app-b*[*cat-Set-cs-simps*]:
assumes $x = b_{PL2}$
shows
 $ntcf\text{-Set-equalizer}\ \alpha\ a\ b\ g\ f\ (NTMap)\ (x) =$
 $g \circ_{A\ cat\text{-Set}\ \alpha}\ incl\text{-Set}\ (vequalizer\ a\ g\ f)\ a$
 $\langle proof \rangle$

12.3.3 Equalizer for the category *Set* is an equalizer

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

12.4 The category *Set* is small-complete

lemma (*in Z*) *cat-small-complete-cat-Set*: *cat-small-complete* $\alpha\ (cat\text{-Set}\ \alpha)$
— This lemma appears as a remark on page 113 in [9].
 $\langle proof \rangle$

13 Adjoints

13.1 Background

named-theorems *adj-cs-simps*

named-theorems *adj-cs-intros*

named-theorems *adj-field-simps*

definition *AdjLeft* :: *V* **where** [*adj-field-simps*]: *AdjLeft* = 0

definition *AdjRight* :: *V* **where** [*adj-field-simps*]: *AdjRight* = 1_N

definition *AdjNT* :: *V* **where** [*adj-field-simps*]: *AdjNT* = 2_N

13.2 Definition and elementary properties

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

locale *is-cf-adjunction* =

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

$\langle \text{op-cat } \mathfrak{C} \times_C \mathfrak{D} \rangle$

$\langle \text{cat-Set } \alpha \rangle$

$\langle \text{Hom}_{O.C\alpha} \mathfrak{D}(\mathfrak{F}-, -) \rangle$

$\langle \text{Hom}_{O.C\alpha} \mathfrak{C}(-, \mathfrak{G}-) \rangle$

$\langle \Phi(\text{AdjNT}) \rangle$

for α \mathfrak{C} \mathfrak{D} \mathfrak{F} \mathfrak{G} Φ +

assumes *cf-adj-length*[*adj-cs-simps*]: *vcard* Φ = 3_N

and *cf-adj-AdjLeft*[*adj-cs-simps*]: $\Phi(\text{AdjLeft}) = \mathfrak{F}$

and *cf-adj-AdjRight*[*adj-cs-simps*]: $\Phi(\text{AdjRight}) = \mathfrak{G}$

syntax *-is-cf-adjunction* :: *V* \Rightarrow *V* \Rightarrow *V* \Rightarrow *V* \Rightarrow *V* \Rightarrow *V* \Rightarrow *bool*

$\langle \langle - : - \Rightarrow_{CF} - : - \Rightarrow_{C1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-cf-adjunction* \Rightarrow *is-cf-adjunction*

translations $\Phi : \mathfrak{F} \Rightarrow_{CF} \mathfrak{G} : \mathfrak{C} \Rightarrow_{C\alpha} \mathfrak{D} \Rightarrow$

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*]:

$[\mathfrak{F}, \mathfrak{G}, \varphi] \circ (\text{AdjLeft}) = \mathfrak{F}$

$[\mathfrak{F}, \mathfrak{G}, \varphi] \circ (\text{AdjRight}) = \mathfrak{G}$

$[\mathfrak{F}, \mathfrak{G}, \varphi] \circ (\text{AdjNT}) = \varphi$

$\langle \text{proof} \rangle$

Rules.

lemma (**in** *is-cf-adjunction*) *is-cf-adjunction-axioms'*[*adj-cs-intros*]:

assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{D}' = \mathfrak{D}$ **and** $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{G}$

shows $\Phi : \mathfrak{F}' \Rightarrow_{CF} \mathfrak{G}' : \mathfrak{C}' \Rightarrow_{C\alpha'} \mathfrak{D}'$

$\langle \text{proof} \rangle$

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 $\mathfrak{F}' = \text{Hom}_{O.C\alpha} \mathfrak{D}(\mathfrak{F}^-, -)$
and $\mathfrak{G}' = \text{Hom}_{O.C\alpha} \mathfrak{C}(-, \mathfrak{G}^-)$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{C} \times_C \mathfrak{D}$
and $\mathfrak{B}' = \text{cat-Set } \alpha$
shows $\Phi(\text{AdjNT}) : \mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto \mapsto_{C\alpha} \mathfrak{B}'$
 ⟨*proof*⟩

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

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

13.3 Opposite adjunction

13.3.1 Definition and elementary properties

See [7] for further information.

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

definition *op-cf-adj* :: $V \Rightarrow V$
where *op-cf-adj* $\Phi =$
 [
 op-cf ($\Phi(\text{AdjRight})$),
 op-cf ($\Phi(\text{AdjLeft})$),
 op-cf-adj-nt ($\Phi(\text{AdjLeft})(\text{HomDom})$) ($\Phi(\text{AdjLeft})(\text{HomCod})$) ($\Phi(\text{AdjNT})$)
]_o

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

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

<proof>

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

The opposite adjunction is an adjunction.

lemma (in is-cf-adjunction) is-cf-adjunction-op:

— See comments in subsection 2.1 in [4].

op-cf-adj $\Phi : \text{op-cf } \mathfrak{G} \rightleftharpoons_{CF} \text{op-cf } \mathfrak{F} : \text{op-cat } \mathfrak{D} \rightleftharpoons_{C\alpha} \text{op-cat } \mathfrak{C}$
<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 $\mathfrak{G}' = \text{op-cf } \mathfrak{G}$

and $\mathfrak{F}' = \text{op-cf } \mathfrak{F}$

and $\mathfrak{D}' = \text{op-cat } \mathfrak{D}$

and $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$

shows $\text{op-cf-adj } \Phi : \mathfrak{G}' \rightleftharpoons_{CF} \mathfrak{F}' : \mathfrak{D}' \rightleftharpoons_{C\alpha} \mathfrak{C}'$

<proof>

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

The operation of taking the opposite adjunction is an involution.

lemma (in is-cf-adjunction) cf-adjunction-op-cf-adj-op-cf-adj[cat-op-simps]:

op-cf-adj (op-cf-adj Φ) = Φ

<proof>

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

13.3.2 Alternative form of the naturality condition

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

lemma (in is-cf-adjunction) cf-adj-Comp-commute-RL:

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$

and $f : \mathfrak{F}(\text{ObjMap})(x) \mapsto_{\mathfrak{D}} a$

and $k : a \mapsto_{\mathfrak{D}} a'$

shows

$\mathfrak{G}(\text{ArrMap})(k) \circ_{A\mathfrak{C}} (\Phi(\text{AdjNT})(\text{NTMap})(x, a) \bullet) (\text{ArrVal})(f) =$
 $(\Phi(\text{AdjNT})(\text{NTMap})(x, a') \bullet) (\text{ArrVal})(k \circ_{A\mathfrak{D}} f)$

<proof>

lemma (in is-cf-adjunction) cf-adj-Comp-commute-LR:

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$

and $f : \mathfrak{F}(\text{ObjMap})(x) \mapsto_{\mathfrak{D}} a$

and $h : x' \mapsto_{\mathfrak{C}} x$

shows

$(\Phi(\text{AdjNT})(\text{NTMap})(x, a) \bullet) (\text{ArrVal})(f) \circ_{A\mathfrak{C}} h =$
 $(\Phi(\text{AdjNT})(\text{NTMap})(x', a) \bullet) (\text{ArrVal})(f \circ_{A\mathfrak{D}} \mathfrak{F}(\text{ArrMap})(h))$

<proof>

13.4 Unit

13.4.1 Definition and elementary properties

See Chapter IV-1 in [9].

definition cf-adjunction-unit :: $V \Rightarrow V$ ($\langle \eta_C \rangle$)

where $\eta_C \Phi =$

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

Components.

lemma *cf-adjunction-unit-components*:

shows $\eta_C \Phi(\text{NTMap}) =$

$$\begin{aligned} & (\\ & \quad \lambda x \in_{\circ} \Phi(\text{AdjLeft})(\text{HomDom})(\text{Obj}). \\ & \quad (\Phi(\text{AdjNT})(\text{NTMap})(x, \Phi(\text{AdjLeft})(\text{ObjMap})(x)) \bullet) (\text{ArrVal})(\\ & \quad \quad \Phi(\text{AdjLeft})(\text{HomCod})(\text{CIId})(\Phi(\text{AdjLeft})(\text{ObjMap})(x)) \\ & \quad) \\ &) \end{aligned}$$

and $\eta_C \Phi(\text{NTDom}) = \text{cf-id } (\Phi(\text{AdjLeft})(\text{HomDom}))$

and $\eta_C \Phi(\text{NTCod}) = (\Phi(\text{AdjRight})) \circ_{CF} (\Phi(\text{AdjLeft}))$

and $\eta_C \Phi(\text{NTDGDom}) = \Phi(\text{AdjLeft})(\text{HomDom})$

and $\eta_C \Phi(\text{NTDGCod}) = \Phi(\text{AdjLeft})(\text{HomDom})$

<proof>

context *is-cf-adjunction*

begin

lemma *cf-adjunction-unit-components'*:

shows $\eta_C \Phi(\text{NTMap}) =$

$$(\lambda x \in_{\circ} \mathfrak{C}(\text{Obj}). (\Phi(\text{AdjNT})(\text{NTMap})(x, \mathfrak{F}(\text{ObjMap})(x)) \bullet) (\text{ArrVal})(\mathfrak{D}(\text{CIId})(\mathfrak{F}(\text{ObjMap})(x))))$$

and $\eta_C \Phi(\text{NTDom}) = \text{cf-id } \mathfrak{C}$

and $\eta_C \Phi(\text{NTCod}) = \mathfrak{G} \circ_{CF} \mathfrak{F}$

and $\eta_C \Phi(\text{NTDGDom}) = \mathfrak{C}$

and $\eta_C \Phi(\text{NTDGCod}) = \mathfrak{C}$

<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)*

|vsu cf-adjunction-unit-NTMap-vsuv[adj-cs-intros]|

lemmas *[adj-cs-simps] =*

is-cf-adjunction.cf-adjunction-unit-NTMap-vdomain

is-cf-adjunction.cf-adjunction-unit-NTMap-app

13.4.2 Natural transformation map

lemma (in *is-cf-adjunction*) *cf-adjunction-unit-NTMap-is-arr*:

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$

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

lemma (in is-cf-adjunction) cf-adjunction-unit-NTMap-is-arr':

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$
and $a = x$
and $b = \mathfrak{G}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(x))$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\eta_C \Phi(\text{NTMap})(x) : x \mapsto_{\mathfrak{C}'} b$
 ⟨proof⟩

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

lemma (in is-cf-adjunction) cf-adjunction-unit-NTMap-vrange:

$\mathcal{R}_{\circ}(\eta_C \Phi(\text{NTMap})) \subseteq_{\circ} \mathfrak{C}(\text{Arr})$
 ⟨proof⟩

13.4.3 Unit is a natural transformation

lemma (in is-cf-adjunction) cf-adjunction-unit-is-ntcf:

$\eta_C \Phi : \text{cf-id } \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{C}$
 ⟨proof⟩

lemma (in is-cf-adjunction) cf-adjunction-unit-is-ntcf':

assumes $\mathfrak{G} = \text{cf-id } \mathfrak{C}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{CF} \mathfrak{F}$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{C}$
shows $\eta_C \Phi : \mathfrak{G} \mapsto_{CF} \mathfrak{G}' : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$
 ⟨proof⟩

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

13.4.4 Every component of a unit is a universal arrow

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

lemma (in is-cf-adjunction) cf-adj-umap-of-unit:

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$ **and** $a \in_{\circ} \mathfrak{D}(\text{Obj})$
shows $\Phi(\text{AdjNT})(\text{NTMap})(x, a)_{\bullet} = \text{umap-of } \mathfrak{G} x (\mathfrak{F}(\text{ObjMap})(x)) (\eta_C \Phi(\text{NTMap})(x)) a$
 (is ⟨ $\Phi(\text{AdjNT})(\text{NTMap})(x, a)_{\bullet} = ?\text{uof-a}$ ⟩)
 ⟨proof⟩

lemma (in is-cf-adjunction) cf-adj-umap-of-unit':

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$
and $a \in_{\circ} \mathfrak{D}(\text{Obj})$
and $\eta = \eta_C \Phi(\text{NTMap})(x)$
and $\mathfrak{F}x = \mathfrak{F}(\text{ObjMap})(x)$
shows $\Phi(\text{AdjNT})(\text{NTMap})(x, a)_{\bullet} = \text{umap-of } \mathfrak{G} x \mathfrak{F}x \eta a$
 ⟨proof⟩

lemma (in is-cf-adjunction) cf-adjunction-unit-component-is-ua-of:

assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$
shows *universal-arrow-of* $\mathfrak{G} x (\mathfrak{F}(\text{ObjMap})(x)) (\eta_C \Phi(\text{NTMap})(x))$
 (is ⟨*universal-arrow-of* $\mathfrak{G} x (\mathfrak{F}(\text{ObjMap})(x)) ?\eta x$ ⟩)
 ⟨proof⟩

13.5 Counit

13.5.1 Definition and elementary properties

definition *cf-adjunction-counit* :: $V \Rightarrow V$ ($\langle \varepsilon_C \rangle$)

where $\varepsilon_C \Phi =$

```
[
  (
     $\lambda x \in \circ \Phi(\text{AdjLeft})(\text{HomCod})(\text{Obj})$ .
     $(\Phi(\text{AdjNT})(\text{NTMap})(\Phi(\text{AdjRight})(\text{ObjMap})(x), x) \bullet)^{-1}_{\text{Set}}(\text{ArrVal})($ 
       $\Phi(\text{AdjLeft})(\text{HomDom})(\text{CId})(\Phi(\text{AdjRight})(\text{ObjMap})(x))$ 
     $)$ 
  ),
   $(\Phi(\text{AdjLeft})) \circ_{CF} (\Phi(\text{AdjRight}))$ ,
  cf-id  $(\Phi(\text{AdjLeft})(\text{HomCod}))$ ,
   $\Phi(\text{AdjLeft})(\text{HomCod})$ ,
   $\Phi(\text{AdjLeft})(\text{HomCod})$ 
]
```

Components.

lemma *cf-adjunction-counit-components*:

shows $\varepsilon_C \Phi(\text{NTMap}) =$

```
(
   $\lambda x \in \circ \Phi(\text{AdjLeft})(\text{HomCod})(\text{Obj})$ .
   $(\Phi(\text{AdjNT})(\text{NTMap})(\Phi(\text{AdjRight})(\text{ObjMap})(x), x) \bullet)^{-1}_{\text{Set}}(\text{ArrVal})($ 
     $\Phi(\text{AdjLeft})(\text{HomDom})(\text{CId})(\Phi(\text{AdjRight})(\text{ObjMap})(x))$ 
   $)$ 
)
```

and $\varepsilon_C \Phi(\text{NTDom}) = (\Phi(\text{AdjLeft})) \circ_{CF} (\Phi(\text{AdjRight}))$

and $\varepsilon_C \Phi(\text{NTCod}) = \text{cf-id } (\Phi(\text{AdjLeft})(\text{HomCod}))$

and $\varepsilon_C \Phi(\text{NTDGDom}) = \Phi(\text{AdjLeft})(\text{HomCod})$

and $\varepsilon_C \Phi(\text{NTDGCod}) = \Phi(\text{AdjLeft})(\text{HomCod})$

<proof>

context *is-cf-adjunction*

begin

lemma *cf-adjunction-counit-components'*:

shows $\varepsilon_C \Phi(\text{NTMap}) =$

```
(
   $\lambda x \in \circ \mathfrak{D}(\text{Obj})$ .
   $(\Phi(\text{AdjNT})(\text{NTMap})(\mathfrak{G}(\text{ObjMap})(x), x) \bullet)^{-1}_{\text{Set}}(\text{ArrVal})(\mathfrak{C}(\text{CId})(\mathfrak{G}(\text{ObjMap})(x)))$ 
)
```

and $\varepsilon_C \Phi(\text{NTDom}) = \mathfrak{F} \circ_{CF} \mathfrak{G}$

and $\varepsilon_C \Phi(\text{NTCod}) = \text{cf-id } \mathfrak{D}$

and $\varepsilon_C \Phi(\text{NTDGDom}) = \mathfrak{D}$

and $\varepsilon_C \Phi(\text{NTDGCod}) = \mathfrak{D}$

<proof>

mk-VLambda *cf-adjunction-counit-components'(1)*

$|vdomain \text{ cf-adjunction-counit-NTMap-vdomain}[adj-cs-simps]|$

$|app \text{ cf-adjunction-counit-NTMap-app}[adj-cs-simps]|$

end

mk-VLambda *cf-adjunction-counit-components(1)*

$|vsu \text{ cf-adjunction-counit-NTMap-vsuv}[adj-cs-intros]|$

lemmas $[adj-cs-simps] =$

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

13.5.2 Duality for the unit and counit

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

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

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

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

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

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

13.5.3 Natural transformation map

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

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

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

lemma (in *is-cf-adjunction*) *cf-adjunction-counit-NTMap-vrange*:
 $\mathcal{R}_o (\varepsilon_C \Phi(\downarrow NTMap)) \subseteq_o \mathfrak{D}(\downarrow Arr)$
 ⟨proof⟩

13.5.4 Counit is a natural transformation

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

lemma (in *is-cf-adjunction*) *cf-adjunction-counit-is-ntcf'*:

assumes $\mathfrak{S} = \mathfrak{F} \circ_{CF} \mathfrak{G}$
 and $\mathfrak{S}' = \text{cf-id } \mathfrak{D}$
 and $\mathfrak{A} = \mathfrak{D}$
 and $\mathfrak{B} = \mathfrak{D}$
 shows $\varepsilon_C \Phi : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto_{C\alpha} \mathfrak{B}$
 $\langle \text{proof} \rangle$

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

13.5.5 Every component of a counit is a universal arrow

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

lemma (in *is-cf-adjunction*) *cf-adj-umap-fo-counit*:

assumes $x \in_{\circ} \mathfrak{D}(\text{Obj})$ and $a \in_{\circ} \mathfrak{C}(\text{Obj})$
 shows *op-cf-adj* $\Phi(\text{AdjNT})(\text{NTMap})(x, a) \bullet =$
umap-fo $\mathfrak{F} x (\mathfrak{G}(\text{ObjMap})(x)) (\varepsilon_C \Phi(\text{NTMap})(x)) a$
 $\langle \text{proof} \rangle$

lemma (in *is-cf-adjunction*) *cf-adjunction-counit-component-is-ua-fo*:

assumes $x \in_{\circ} \mathfrak{D}(\text{Obj})$
 shows *universal-arrow-fo* $\mathfrak{F} x (\mathfrak{G}(\text{ObjMap})(x)) (\varepsilon_C \Phi(\text{NTMap})(x))$
 $\langle \text{proof} \rangle$

13.5.6 Further properties

lemma (in *is-cf-adjunction*) *cf-adj-AdjNT-cf-adjunction-unit*:

— See Chapter IV-1 in [9].
 assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$ and $f : \mathfrak{F}(\text{ObjMap})(x) \mapsto_{\mathfrak{D}} a$
 shows
 $\mathfrak{G}(\text{ArrMap})(f) \circ_{A\mathfrak{C}} \eta_C \Phi(\text{NTMap})(x) =$
 $(\Phi(\text{AdjNT})(\text{NTMap})(x, a) \bullet)(\text{ArrVal})(f)$
 $\langle \text{proof} \rangle$

lemma (in *is-cf-adjunction*) *cf-adj-AdjNT-cf-adjunction-counit*:

— See Chapter IV-1 in [9].
 assumes $x \in_{\circ} \mathfrak{D}(\text{Obj})$ and $g : a \mapsto_{\mathfrak{C}} \mathfrak{G}(\text{ObjMap})(x)$
 shows
 $\varepsilon_C \Phi(\text{NTMap})(x) \circ_{A\mathfrak{D}} \mathfrak{F}(\text{ArrMap})(g) =$
 $(\Phi(\text{AdjNT})(\text{NTMap})(a, x) \bullet)^{-1}_{C \text{ cat-Set } \alpha}(\text{ArrVal})(g)$
 $\langle \text{proof} \rangle$

lemma (in *is-cf-adjunction*) *cf-adj-counit-unit-app[adj-cs-simps]*:

— See Chapter IV-1 in [9].
 assumes $x \in_{\circ} \mathfrak{D}(\text{Obj})$ and $g : a \mapsto_{\mathfrak{C}} \mathfrak{G}(\text{ObjMap})(x)$
 shows $\mathfrak{G}(\text{ArrMap})(\varepsilon_C \Phi(\text{NTMap})(x) \circ_{A\mathfrak{D}} \mathfrak{F}(\text{ArrMap})(g)) \circ_{A\mathfrak{C}} \eta_C \Phi(\text{NTMap})(a) = g$
 $\langle \text{proof} \rangle$

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

lemma (in *is-cf-adjunction*) *cf-adj-unit-counit-app[adj-cs-simps]*:

— See Chapter IV-1 in [9].
 assumes $x \in_{\circ} \mathfrak{C}(\text{Obj})$ and $f : \mathfrak{F}(\text{ObjMap})(x) \mapsto_{\mathfrak{D}} a$
 shows $\varepsilon_C \Phi(\text{NTMap})(a) \circ_{A\mathfrak{D}} \mathfrak{F}(\text{ArrMap})(\mathfrak{G}(\text{ArrMap})(f) \circ_{A\mathfrak{C}} \eta_C \Phi(\text{NTMap})(x)) = f$
 $\langle \text{proof} \rangle$

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

13.6 Counit-unit equations

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

lemma (in *is-cf-adjunction*) *cf-adjunction-counit-unit*:

$$\begin{aligned} & (\mathfrak{G} \circ_{CF-NTCF} \varepsilon_C \Phi) \cdot_{NTCF} (\eta_C \Phi \circ_{NTCF-CF} \mathfrak{G}) = ntcf-id \mathfrak{G} \\ & (\text{is } \langle (\mathfrak{G} \circ_{CF-NTCF} ?\varepsilon) \cdot_{NTCF} (? \eta \circ_{NTCF-CF} \mathfrak{G}) = ntcf-id \mathfrak{G} \rangle) \\ & \langle proof \rangle \end{aligned}$$

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

lemma (in *is-cf-adjunction*) *cf-adjunction-unit-counit*:

$$\begin{aligned} & (\varepsilon_C \Phi \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta_C \Phi) = ntcf-id \mathfrak{F} \\ & (\text{is } \langle (? \varepsilon \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} ? \eta) = ntcf-id \mathfrak{F} \rangle) \\ & \langle proof \rangle \end{aligned}$$

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

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

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

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

definition *cf-adjunction-AdjNT-of-unit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta =$

$$\begin{aligned} & [\\ & \quad (\lambda cd \varepsilon_o (op-cat (\mathfrak{F}(\text{HomDom})) \times_C \mathfrak{F}(\text{HomCod}))(\text{Obj}). \\ & \quad \quad umap-of \mathfrak{G} (cd(\text{Obj})) (\mathfrak{F}(\text{ObjMap})(cd(\text{Obj}))) (\eta(\text{NTMap})(cd(\text{Obj}))) (cd(\text{Id}))), \\ & \quad \text{Hom}_{O.C\alpha} \mathfrak{F}(\text{HomCod})(\mathfrak{F}-, -), \\ & \quad \text{Hom}_{O.C\alpha} \mathfrak{F}(\text{HomDom})(-, \mathfrak{G}-), \\ & \quad op-cat (\mathfrak{F}(\text{HomDom})) \times_C (\mathfrak{F}(\text{HomCod})), \\ & \quad \text{cat-Set } \alpha \\ &]_o \end{aligned}$$

Components.

lemma *cf-adjunction-AdjNT-of-unit-components*:

shows *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta(\text{NTMap}) =$

$$\begin{aligned} & (\\ & \quad \lambda cd \varepsilon_o (op-cat (\mathfrak{F}(\text{HomDom})) \times_C \mathfrak{F}(\text{HomCod}))(\text{Obj}). \\ & \quad \quad umap-of \mathfrak{G} (cd(\text{Obj})) (\mathfrak{F}(\text{ObjMap})(cd(\text{Obj}))) (\eta(\text{NTMap})(cd(\text{Obj}))) (cd(\text{Id})) \\ &) \end{aligned}$$

and *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta(\text{NTDom}) = \text{Hom}_{O.C\alpha} \mathfrak{F}(\text{HomCod})(\mathfrak{F}-, -)$

and *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta(\text{NTCod}) = \text{Hom}_{O.C\alpha} \mathfrak{F}(\text{HomDom})(-, \mathfrak{G}-)$

and *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta(\text{NTDGDom}) =$

$$op-cat (\mathfrak{F}(\text{HomDom})) \times_C (\mathfrak{F}(\text{HomCod}))$$

and *cf-adjunction-AdjNT-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta(\text{NTDGCod}) = \text{cat-Set } \alpha$

$\langle proof \rangle$

13.7.2 Natural transformation map

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

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

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vdomain*[*adj-cs-simps*]:

assumes $\mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{D}$

shows \mathcal{D}_o (*cf-adjunction-AdjNT-of-unit* α \mathfrak{F} \mathfrak{G} η (*NTMap*)) = (*op-cat* $\mathcal{C} \times_C \mathcal{D}$)(*Obj*)

⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-NTMap-app*[*adj-cs-simps*]:

assumes $\mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{D}$ **and** $c \in_o \mathcal{C}$ (*Obj*) **and** $d \in_o \mathcal{D}$ (*Obj*)

shows

cf-adjunction-AdjNT-of-unit α \mathfrak{F} \mathfrak{G} η (*NTMap*)(c , d) \bullet =
umap-of \mathfrak{G} c (\mathfrak{F} (*ObjMap*)(c)) (η (*NTMap*)(c)) d

⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-NTMap-vrange*:

assumes *category* α \mathcal{C}

and *category* α \mathcal{D}

and $\mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{D}$

and $\mathfrak{G} : \mathcal{D} \mapsto \mapsto_{C\alpha} \mathcal{C}$

and $\eta : \text{cf-id } \mathcal{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{C}$

shows \mathcal{R}_o (*cf-adjunction-AdjNT-of-unit* α \mathfrak{F} \mathfrak{G} η (*NTMap*)) \subseteq_o *cat-Set* α (*Arr*)

⟨*proof*⟩

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

lemma *cf-adjunction-AdjNT-of-unit-is-ntcf*:

assumes *category* α \mathcal{C}

and *category* α \mathcal{D}

and $\mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{D}$

and $\mathfrak{G} : \mathcal{D} \mapsto \mapsto_{C\alpha} \mathcal{C}$

and $\eta : \text{cf-id } \mathcal{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{C}$

shows *cf-adjunction-AdjNT-of-unit* α \mathfrak{F} \mathfrak{G} η :

Hom $_{O.C\alpha} \mathcal{D}(\mathfrak{F}-, -) \mapsto_{CF} \text{Hom}_{O.C\alpha} \mathcal{C}(-, \mathfrak{G}-) :$

op-cat $\mathcal{C} \times_C \mathcal{D} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$

⟨*proof*⟩

lemma *cf-adjunction-AdjNT-of-unit-is-ntcf'*[*adj-cs-intros*]:

assumes *category* α \mathcal{C}

and *category* α \mathcal{D}

and $\mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{D}$

and $\mathfrak{G} : \mathcal{D} \mapsto \mapsto_{C\alpha} \mathcal{C}$

and $\eta : \text{cf-id } \mathcal{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathcal{C} \mapsto \mapsto_{C\alpha} \mathcal{C}$

and $\mathfrak{S} = \text{Hom}_{O.C\alpha} \mathcal{D}(\mathfrak{F}-, -)$

and $\mathfrak{S}' = \text{Hom}_{O.C\alpha} \mathcal{C}(-, \mathfrak{G}-)$

and $\mathfrak{A} = \text{op-cat } \mathcal{C} \times_C \mathcal{D}$

and $\mathfrak{B} = \text{cat-Set } \alpha$

shows *cf-adjunction-AdjNT-of-unit* α \mathfrak{F} \mathfrak{G} η : $\mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto \mapsto_{C\alpha} \mathfrak{B}$

⟨*proof*⟩

13.7.4 Adjunction constructed from universal morphisms from objects to functors

definition *cf-adjunction-of-unit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-adjunction-of-unit* α \mathfrak{F} \mathfrak{G} η =

$[\mathfrak{F}, \mathfrak{G}, \text{cf-adjunction-AdjNT-of-unit } \alpha \mathfrak{F} \mathfrak{G} \eta]_o$.

Components.

lemma *cf-adjunction-of-unit-components*:

shows $[adj\text{-}cs\text{-}simps]: cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\downarrow AdjLeft) = \mathfrak{F}$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\downarrow AdjRight) = \mathfrak{G}$
and $cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\downarrow AdjNT) =$
 $cf\text{-}adjunction\text{-}AdjNT\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta$
 $\langle proof \rangle$

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 $\mathcal{D}_\circ (cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\downarrow AdjNT)(\downarrow NTMap)) =$
 $(op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{D})(\downarrow Obj)$
 $\langle proof \rangle$

lemma *cf-adjunction-of-unit-AdjNT-NTMap-app[adj-cs-simps]*:

assumes $\mathfrak{F} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \mathfrak{D}$ **and** $c \in_\circ \mathfrak{C}(\downarrow Obj)$ **and** $d \in_\circ \mathfrak{D}(\downarrow Obj)$
shows
 $cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta(\downarrow AdjNT)(\downarrow NTMap)(\downarrow c, d)_\bullet =$
 $umap\text{-}of\ \mathfrak{G}\ c\ (\mathfrak{F}(\downarrow ObjMap)(\downarrow c))\ (\eta(\downarrow NTMap)(\downarrow c))\ d$
 $\langle proof \rangle$

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

lemma *cf-adjunction-of-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 $\eta : cf\text{-}id\ \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\wedge x. x \in_\circ \mathfrak{C}(\downarrow Obj) \implies universal\text{-}arrow\text{-}of\ \mathfrak{G}\ x\ (\mathfrak{F}(\downarrow ObjMap)(\downarrow x))\ (\eta(\downarrow NTMap)(\downarrow x))$
shows $cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\eta_C (cf\text{-}adjunction\text{-}of\text{-}unit\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \eta) = \eta$
 $\langle proof \rangle$

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

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

13.8.1 Left adjoint

definition *cf-la-of-ra* :: $(V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-la-of-ra* $F\ \mathfrak{G}\ \eta =$

[
 $(\lambda x \in_\circ \mathfrak{G}(\downarrow HomCod)(\downarrow Obj). F\ x),$
(
 $\lambda h \in_\circ \mathfrak{G}(\downarrow HomCod)(\downarrow Arr). THE\ f'$
 $f' : F\ (\mathfrak{G}(\downarrow HomCod)(\downarrow Dom)(\downarrow h)) \mapsto_{\mathfrak{G}(\downarrow HomCod)} F\ (\mathfrak{G}(\downarrow HomCod)(\downarrow Cod)(\downarrow h)) \wedge$
 $\eta(\downarrow \mathfrak{G}(\downarrow HomCod)(\downarrow Cod)(\downarrow h)) \circ_A \mathfrak{G}(\downarrow HomCod)\ h =$
(
 $umap\text{-}of$
 \mathfrak{G}
 $(\mathfrak{G}(\downarrow HomCod)(\downarrow Dom)(\downarrow h))$
 $(F\ (\mathfrak{G}(\downarrow HomCod)(\downarrow Dom)(\downarrow h)))$
 $(\eta(\downarrow \mathfrak{G}(\downarrow HomCod)(\downarrow Dom)(\downarrow h)))$

$$\begin{aligned}
& (F (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Cod})(h))) \\
&)(\mathfrak{ArrVal})(f') \\
&), \\
& \mathfrak{G}(\mathfrak{HomCod}), \\
& \mathfrak{G}(\mathfrak{HomDom}) \\
&]_0
\end{aligned}$$

Components.

lemma *cf-la-of-ra-components*:

shows *cf-la-of-ra* $F \mathfrak{G} \eta(\mathfrak{ObjMap}) = (\lambda x \in_0 \mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Obj}). F x)$

and *cf-la-of-ra* $F \mathfrak{G} \eta(\mathfrak{ArrMap}) =$

$$\begin{aligned}
& (\\
& \lambda h \in_0 \mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Arr}). \text{THE } f' \\
& f' : F (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Dom})(h)) \mapsto_{\mathfrak{G}(\mathfrak{HomDom})} F (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Cod})(h)) \wedge \\
& \eta(\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Cod})(h)) \circ_A \mathfrak{G}(\mathfrak{HomCod}) h = \\
& (\\
& \text{umap-of} \\
& \mathfrak{G} \\
& (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Dom})(h)) \\
& (F (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Dom})(h))) \\
& (\eta(\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Dom})(h))) \\
& (F (\mathfrak{G}(\mathfrak{HomCod})(\mathfrak{Cod})(h))) \\
&)(\mathfrak{ArrVal})(f') \\
&) \\
& \text{and } \textit{cf-la-of-ra} F \mathfrak{G} \eta(\mathfrak{HomDom}) = \mathfrak{G}(\mathfrak{HomCod}) \\
& \text{and } \textit{cf-la-of-ra} F \mathfrak{G} \eta(\mathfrak{HomCod}) = \mathfrak{G}(\mathfrak{HomDom}) \\
& \langle \textit{proof} \rangle
\end{aligned}$$

13.8.2 Object map

mk-VLambda *cf-la-of-ra-components(1)*

$|vsu \textit{cf-la-of-ra-ObjMap-vsuv}[\textit{adj-cs-intros}]|$

mk-VLambda (in *is-functor*)

cf-la-of-ra-components(1) [where $?G = \mathfrak{F}$, *unfolded cf-HomCod*]

$|vdomain \textit{cf-la-of-ra-ObjMap-vdomain}[\textit{adj-cs-simps}]|$

$|app \textit{cf-la-of-ra-ObjMap-app}[\textit{adj-cs-simps}]|$

lemmas [*adj-cs-simps*] =

is-functor.cf-la-of-ra-ObjMap-vdomain

is-functor.cf-la-of-ra-ObjMap-app

13.8.3 Arrow map

mk-VLambda *cf-la-of-ra-components(2)*

$|vsu \textit{cf-la-of-ra-ArrMap-vsuv}[\textit{adj-cs-intros}]|$

mk-VLambda (in *is-functor*)

cf-la-of-ra-components(2) [where $?G = \mathfrak{F}$, *unfolded cf-HomCod cf-HomDom*]

$|vdomain \textit{cf-la-of-ra-ArrMap-vdomain}[\textit{adj-cs-simps}]|$

$|app \textit{cf-la-of-ra-ArrMap-app}|$

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-la-of-ra $F \mathfrak{F} \eta(\mathfrak{ArrMap})(h) =$

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

lemma *cf-la-of-ra-ArrMap-app-unique*:

assumes $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$
and $f : a \mapsto_{\mathfrak{C}} b$
and *universal-arrow-of* $\mathfrak{G} a$ (*cf-la-of-ra* $F \mathfrak{G} \eta(\text{ObjMap})(a)$) ($\eta(a)$)
and *universal-arrow-of* $\mathfrak{G} b$ (*cf-la-of-ra* $F \mathfrak{G} \eta(\text{ObjMap})(b)$) ($\eta(b)$)
shows *cf-la-of-ra* $F \mathfrak{G} \eta(\text{ArrMap})(f) : F a \mapsto_{\mathfrak{D}} F b$
and $\eta(b) \circ_{A\mathfrak{C}} f =$
 $\text{umap-of } \mathfrak{G} a (F a) (\eta(a)) (F b) (\text{ArrVal})(\text{cf-la-of-ra } F \mathfrak{G} \eta(\text{ArrMap})(f))$
and $\wedge f'$.
 [[
 $f' : F a \mapsto_{\mathfrak{D}} F b;$
 $\eta(b) \circ_{A\mathfrak{C}} f = \text{umap-of } \mathfrak{G} a (F a) (\eta(a)) (F b) (\text{ArrVal})(f')$
]]
 $\implies \text{cf-la-of-ra } F \mathfrak{G} \eta(\text{ArrMap})(f) = f'$
 ⟨proof⟩

lemma *cf-la-of-ra-ArrMap-app-is-arr[adj-cs-intros]*:

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

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

lemma *cf-la-of-ra-is-functor*:

assumes $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$
and $\wedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies F c \in_{\circ} \mathfrak{D}(\text{Obj})$
and $\wedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies$
universal-arrow-of $\mathfrak{G} c$ (*cf-la-of-ra* $F \mathfrak{G} \eta(\text{ObjMap})(c)$) ($\eta(c)$)
and $\wedge c c' h. h : c \mapsto_{\mathfrak{C}} c' \implies$
 $\mathfrak{G}(\text{ArrMap})(\text{cf-la-of-ra } F \mathfrak{G} \eta(\text{ArrMap})(h)) \circ_{A\mathfrak{C}} \eta(c) = \eta(c') \circ_{A\mathfrak{C}} h$
shows *cf-la-of-ra* $F \mathfrak{G} \eta : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D}$ (**is** $\langle \mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D} \rangle$)
 ⟨proof⟩

lemma *cf-la-of-ra-is-ntcf*:

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

lemma *cf-la-of-ra-is-unit*:

fixes $F \mathfrak{C} \mathfrak{F} \mathfrak{G} \eta_m \eta$

defines $\mathfrak{F} \equiv \text{cf-la-of-ra } F \mathfrak{G} \eta_m$

and $\eta \equiv [\eta_m, \text{cf-id } \mathfrak{C}, \mathfrak{G} \circ_{CF} \mathfrak{F}, \mathfrak{C}, \mathfrak{C}]_o$

assumes *category* $\alpha \mathfrak{C}$

and *category* $\alpha \mathfrak{D}$

and $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$

and $\bigwedge c. c \in_o \mathfrak{C}(\text{Obj}) \implies F c \in_o \mathfrak{D}(\text{Obj})$

and $\bigwedge c. c \in_o \mathfrak{C}(\text{Obj}) \implies$

universal-arrow-of $\mathfrak{G} c (\mathfrak{F}(\text{ObjMap})(c)) (\eta(\text{NTMap})(c))$

and $\bigwedge c c' h. h : c \mapsto_{\mathfrak{C}} c' \implies$

$\mathfrak{G}(\text{ArrMap})(\mathfrak{F}(\text{ArrMap})(h)) \circ_{A\mathfrak{C}} (\eta(\text{NTMap})(c)) = (\eta(\text{NTMap})(c')) \circ_{A\mathfrak{C}} h$

and *vsv* $(\eta(\text{NTMap}))$

and $\mathfrak{D}_o (\eta(\text{NTMap})) = \mathfrak{C}(\text{Obj})$

shows *cf-adjunction-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta : \mathfrak{F} \rightleftarrows_{CF} \mathfrak{G} : \mathfrak{C} \rightleftarrows_{C\alpha} \mathfrak{D}$

and $\eta_C (\text{cf-adjunction-of-unit } \alpha \mathfrak{F} \mathfrak{G} \eta) = \eta$

<proof>

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

13.9.1 Definition and elementary properties

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

definition *cf-adjunction-of-counit* $:: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon =$

op-cf-adj $(\text{cf-adjunction-of-unit } \alpha (\text{op-cf } \mathfrak{G}) (\text{op-cf } \mathfrak{F}) (\text{op-ntcf } \varepsilon))$

Components.

lemma *cf-adjunction-of-counit-components*:

shows *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon(\text{AdjLeft}) = \text{op-cf } (\text{op-cf } \mathfrak{F})$

and *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon(\text{AdjRight}) = \text{op-cf } (\text{op-cf } \mathfrak{G})$

and *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon(\text{AdjNT}) = \text{op-cf-adj-nt}$

$(\text{op-cf } \mathfrak{G}(\text{HomDom}))$

$(\text{op-cf } \mathfrak{G}(\text{HomCod}))$

$(\text{cf-adjunction-AdjNT-of-unit } \alpha (\text{op-cf } \mathfrak{G}) (\text{op-cf } \mathfrak{F}) (\text{op-ntcf } \varepsilon))$

<proof>

13.9.2 Natural transformation map

lemma *cf-adjunction-of-counit-NTMap-vsuv*:

vsuv $(\text{cf-adjunction-of-counit } \alpha \mathfrak{F} \mathfrak{G} \varepsilon(\text{AdjNT})(\text{NTMap}))$

<proof>

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

lemma *cf-adjunction-of-counit-is-cf-adjunction*:

assumes *category* $\alpha \mathfrak{C}$

and *category* $\alpha \mathfrak{D}$

and $\mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D}$

and $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$

and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} \text{cf-id } \mathfrak{D} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{D}$

and $\bigwedge x. x \in_o \mathfrak{D}(\text{Obj}) \implies \text{universal-arrow-fo } \mathfrak{F} x (\mathfrak{G}(\text{ObjMap})(x)) (\varepsilon(\text{NTMap})(x))$

shows *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon : \mathfrak{F} \Rightarrow_{CF} \mathfrak{G} : \mathfrak{C} \Rightarrow_{C\alpha} \mathfrak{D}$
and ε_C (*cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon$) = ε
and \mathcal{D}_o (*cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon$) (*AdjNT*) (*NTMap*) =
op-cat $\mathfrak{C} \times_C \mathfrak{D}$ (*Obj*)
and $\wedge c d$. $\llbracket c \in_o \mathfrak{C}(\text{Obj}); d \in_o \mathfrak{D}(\text{Obj}) \rrbracket \implies$
cf-adjunction-of-counit $\alpha \mathfrak{F} \mathfrak{G} \varepsilon$ (*AdjNT*) (*NTMap*) (c, d) \bullet =
umap-fo $\mathfrak{F} d$ ($\mathfrak{G}(\text{ObjMap})(d)$) ($\varepsilon(\text{NTMap})(d)$) c^{-1}_{Set}
<proof>

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

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

13.10.1 Definition and elementary properties

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

13.10.2 Object map

lemma *cf-ra-of-la-ObjMap-vsuv* [*adj-cs-intros*]: *vsu* (*cf-ra-of-la* $F \mathfrak{F} \varepsilon$) (*ObjMap*)
<proof>

lemma (**in** *is-functor*) *cf-ra-of-la-ObjMap-vdomain*:
 \mathcal{D}_o (*cf-ra-of-la* $F \mathfrak{F} \varepsilon$) (*ObjMap*) = $\mathfrak{B}(\text{Obj})$
<proof>

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

lemma (**in** *is-functor*) *cf-ra-of-la-ObjMap-app*:
assumes $d \in_o \mathfrak{B}(\text{Obj})$
shows *cf-ra-of-la* $F \mathfrak{F} \varepsilon$ (*ObjMap*) (d) = $F d$
<proof>

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

13.10.3 Arrow map

lemma *cf-ra-of-la-ArrMap-app-unique*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto_{CF} C\alpha \mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} b$
and *universal-arrow-fo* $\mathfrak{F} a$ (*cf-ra-of-la* $F \mathfrak{F} \varepsilon$) (*ObjMap*) (a) ($\varepsilon(a)$)
and *universal-arrow-fo* $\mathfrak{F} b$ (*cf-ra-of-la* $F \mathfrak{F} \varepsilon$) (*ObjMap*) (b) ($\varepsilon(b)$)
shows *cf-ra-of-la* $F \mathfrak{F} \varepsilon$ (*ArrMap*) (f) : $F a \mapsto_{\mathfrak{C}} F b$
and $f \circ_{A\mathfrak{D}} \varepsilon(a)$ =
umap-fo $\mathfrak{F} b$ ($F b$) ($\varepsilon(b)$) ($F a$) (*ArrVal*) (*cf-ra-of-la* $F \mathfrak{F} \varepsilon$) (*ArrMap*) (f)
and $\wedge f'$.
 \llbracket
 $f' : F a \mapsto_{\mathfrak{C}} F b;$
 $f \circ_{A\mathfrak{D}} \varepsilon(a) = \text{umap-fo } \mathfrak{F} b (F b) (\varepsilon(b)) (F a) (\text{ArrVal}) (f')$
 $\rrbracket \implies \text{cf-ra-of-la } F \mathfrak{F} \varepsilon (\text{ArrMap}) (f) = f'$
<proof>

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

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

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

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

lemma *cf-ra-of-la-commute-op*:
assumes $\mathfrak{F} : \mathcal{C} \mapsto_{C\alpha} \mathcal{D}$
and $\wedge d. d \in_{\circ} \mathcal{D}(\text{Obj}) \implies$
universal-arrow-fo $\mathfrak{F} d$ (*cf-ra-of-la* $F \mathfrak{F} \varepsilon(\text{ObjMap})(\downarrow d)$) ($\varepsilon(\downarrow d)$)
and $\wedge d d' h. h : d \mapsto_{\mathcal{D}} d' \implies$
 $\varepsilon(\downarrow d') \circ_{A\mathcal{D}} \mathfrak{F}(\text{ArrMap})(\downarrow \text{cf-ra-of-la } F \mathfrak{F} \varepsilon(\text{ArrMap})(\downarrow h)) =$
 $h \circ_{A\mathcal{D}} \varepsilon(\downarrow d)$
and $h : c' \mapsto_{\mathcal{D}} c$
shows $\mathfrak{F}(\text{ArrMap})(\downarrow \text{cf-ra-of-la } F \mathfrak{F} \varepsilon(\text{ArrMap})(\downarrow h)) \circ_{A \text{ op-cat } \mathcal{D}} \varepsilon(\downarrow c) =$
 $\varepsilon(\downarrow c') \circ_{A \text{ op-cat } \mathcal{D}} h$
<proof>

lemma
assumes $\mathfrak{F} : \mathcal{C} \mapsto_{C\alpha} \mathcal{D}$
and $\wedge d. d \in_{\circ} \mathcal{D}(\text{Obj}) \implies F d \in_{\circ} \mathcal{C}(\text{Obj})$
and $\wedge d. d \in_{\circ} \mathcal{D}(\text{Obj}) \implies$
universal-arrow-fo $\mathfrak{F} d$ (*cf-ra-of-la* $F \mathfrak{F} \varepsilon(\text{ObjMap})(\downarrow d)$) ($\varepsilon(\downarrow d)$)
and $\wedge d d' h. h : d \mapsto_{\mathcal{D}} d' \implies$
 $\varepsilon(\downarrow d') \circ_{A\mathcal{D}} \mathfrak{F}(\text{ArrMap})(\downarrow \text{cf-ra-of-la } F \mathfrak{F} \varepsilon(\text{ArrMap})(\downarrow h)) =$
 $h \circ_{A\mathcal{D}} \varepsilon(\downarrow d)$
shows *cf-ra-of-la-is-functor*: *cf-ra-of-la* $F \mathfrak{F} \varepsilon : \mathcal{D} \mapsto_{C\alpha} \mathcal{C}$
and *cf-la-of-ra-op-is-functor*:
cf-la-of-ra F (*op-cf* \mathfrak{F}) $\varepsilon : \text{op-cat } \mathcal{D} \mapsto_{C\alpha} \text{op-cat } \mathcal{C}$
<proof>

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

lemma *cf-ra-of-la-is-counit*:
fixes $F \mathcal{D} \mathfrak{F} \mathfrak{G} \varepsilon_m \varepsilon$
defines $\mathfrak{G} \equiv \text{cf-ra-of-la } F \mathfrak{F} \varepsilon_m$

and $\varepsilon \equiv [\varepsilon_m, \mathfrak{F} \circ_{CF} \mathfrak{G}, cf-id \mathfrak{D}, \mathfrak{D}, \mathfrak{D}]_o$
assumes *category* $\alpha \mathfrak{C}$
and *category* $\alpha \mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D}$
and $\wedge d. d \in_o \mathfrak{D}(\text{Obj}) \implies F d \in_o \mathfrak{C}(\text{Obj})$
and $\wedge d. d \in_o \mathfrak{D}(\text{Obj}) \implies$
universal-arrow-fo $\mathfrak{F} d (\mathfrak{G}(\text{ObjMap})(d)) (\varepsilon(\text{NTMap})(d))$
and $\wedge d d' h. h : d \mapsto_{\mathfrak{D}} d' \implies$
 $\varepsilon(\text{NTMap})(d') \circ_{A\mathfrak{D}} \mathfrak{F}(\text{ArrMap})(\mathfrak{G}(\text{ArrMap})(h)) = h \circ_{A\mathfrak{D}} \varepsilon(\text{NTMap})(d)$
and *vfsequence* ε
and *vsv* $(\varepsilon(\text{NTMap}))$
and $\mathfrak{D}_o (\varepsilon(\text{NTMap})) = \mathfrak{D}(\text{Obj})$
shows *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\varepsilon_C (\text{cf-adjunction-of-counit } \alpha \mathfrak{F} \mathfrak{G} \varepsilon) = \varepsilon$
{proof}

13.11 Construction of an adjunction from the counit-unit equations

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

lemma *counit-unit-is-cf-adjunction:*

assumes *category* $\alpha \mathfrak{C}$
and *category* $\alpha \mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$
and $\eta : cf-id \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} cf-id \mathfrak{D} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \cdot_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = ntcf-id \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = ntcf-id \mathfrak{F}$
shows *cf-adjunction-of-unit* $\alpha \mathfrak{F} \mathfrak{G} \eta : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\eta_C (\text{cf-adjunction-of-unit } \alpha \mathfrak{F} \mathfrak{G} \eta) = \eta$
and $\varepsilon_C (\text{cf-adjunction-of-unit } \alpha \mathfrak{F} \mathfrak{G} \eta) = \varepsilon$
{proof}

lemma *counit-unit-cf-adjunction-of-counit-is-cf-adjunction:*

assumes *category* $\alpha \mathfrak{C}$
and *category* $\alpha \mathfrak{D}$
and $\mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{C}$
and $\eta : cf-id \mathfrak{C} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{C} \mapsto_{C\alpha} \mathfrak{C}$
and $\varepsilon : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF} cf-id \mathfrak{D} : \mathfrak{D} \mapsto_{C\alpha} \mathfrak{D}$
and $(\mathfrak{G} \circ_{CF-NTCF} \varepsilon) \cdot_{NTCF} (\eta \circ_{NTCF-CF} \mathfrak{G}) = ntcf-id \mathfrak{G}$
and $(\varepsilon \circ_{NTCF-CF} \mathfrak{F}) \cdot_{NTCF} (\mathfrak{F} \circ_{CF-NTCF} \eta) = ntcf-id \mathfrak{F}$
shows *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathfrak{G} \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathfrak{D}$
and $\eta_C (\text{cf-adjunction-of-counit } \alpha \mathfrak{F} \mathfrak{G} \varepsilon) = \eta$
and $\varepsilon_C (\text{cf-adjunction-of-counit } \alpha \mathfrak{F} \mathfrak{G} \varepsilon) = \varepsilon$
{proof}

13.12 Adjoints are unique up to isomorphism

The content of the following subsection is based predominantly on the statement and the proof of Corollary 1 in Chapter IV-1 in [9]. However, similar results can also be found in section 4 in [14] and in subsection 2.1 in [4].

13.12.1 Definitions and elementary properties

definition *cf-adj-LR-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-adj-LR-iso* $\mathfrak{C} \mathfrak{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi =$

[
 (
 $\lambda x \in_{\circ} \mathfrak{C}(\text{Obj}). \text{THE } f'.$
let
 $\eta = \eta_C \Phi;$
 $\eta' = \eta_C \Psi;$
 $\mathfrak{F}x = \mathfrak{F}(\text{ObjMap})(x);$
 $\mathfrak{F}'x = \mathfrak{F}'(\text{ObjMap})(x)$
in
 $f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
 $\eta'(\text{NTMap})(x) = \text{umap-of } \mathfrak{G} \ x \ (\mathfrak{F}x) \ (\eta(\text{NTMap})(x)) \ (\mathfrak{F}'x)(\text{ArrVal})(f')$
),
 $\mathfrak{F},$
 $\mathfrak{F}',$
 $\mathfrak{C},$
 \mathfrak{D}
]_o.

definition *cf-adj-RL-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-adj-RL-iso* $\mathfrak{C} \mathfrak{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi =$

[
 (
 $\lambda x \in_{\circ} \mathfrak{D}(\text{Obj}). \text{THE } f'.$
let
 $\varepsilon = \varepsilon_C \Phi;$
 $\varepsilon' = \varepsilon_C \Psi;$
 $\mathfrak{G}x = \mathfrak{G}(\text{ObjMap})(x);$
 $\mathfrak{G}'x = \mathfrak{G}'(\text{ObjMap})(x)$
in
 $f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x \wedge$
 $\varepsilon'(\text{NTMap})(x) = \text{umap-fo } \mathfrak{F} \ x \ \mathfrak{G}x \ (\varepsilon(\text{NTMap})(x)) \ \mathfrak{G}'x(\text{ArrVal})(f')$
),
 $\mathfrak{G}',$
 $\mathfrak{G},$
 $\mathfrak{D},$
 \mathfrak{C}
]_o.

Components.

lemma *cf-adj-LR-iso-components*:

shows *cf-adj-LR-iso* $\mathfrak{C} \mathfrak{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(\text{NTMap}) =$

(
 $\lambda x \in_{\circ} \mathfrak{C}(\text{Obj}). \text{THE } f'.$
let
 $\eta = \eta_C \Phi;$
 $\eta' = \eta_C \Psi;$
 $\mathfrak{F}x = \mathfrak{F}(\text{ObjMap})(x);$
 $\mathfrak{F}'x = \mathfrak{F}'(\text{ObjMap})(x)$
in
 $f' : \mathfrak{F}x \mapsto_{\mathfrak{D}} \mathfrak{F}'x \wedge$
 $\eta'(\text{NTMap})(x) = \text{umap-of } \mathfrak{G} \ x \ \mathfrak{F}x \ (\eta(\text{NTMap})(x)) \ \mathfrak{F}'x(\text{ArrVal})(f')$
)
and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C} \mathfrak{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(\text{NTDom}) = \mathfrak{F}$
and [*adj-cs-simps*]: *cf-adj-LR-iso* $\mathfrak{C} \mathfrak{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(\text{NTCod}) = \mathfrak{F}'$

and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}LR\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(NTDGDom) = \mathcal{C}$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}LR\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(NTDGCod) = \mathcal{D}$
 $\langle proof \rangle$

lemma *cf-adj-RL-iso-components*:

shows $cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{G}' \Psi(NTMap) =$
 $($
 $\lambda x \in_{\circ} \mathcal{D}(\mathcal{O}bj). \text{ THE } f'.$
let
 $\varepsilon = \varepsilon_C \Phi;$
 $\varepsilon' = \varepsilon_C \Psi;$
 $\mathfrak{G}x = \mathfrak{G}(\mathcal{O}bjMap)(x);$
 $\mathfrak{G}'x = \mathfrak{G}'(\mathcal{O}bjMap)(x)$
in
 $f' : \mathfrak{G}'x \mapsto_{\mathcal{C}} \mathfrak{G}x \wedge$
 $\varepsilon'(\mathcal{N}TMap)(x) = \text{umap-fo } \mathfrak{F} x \mathfrak{G}x (\varepsilon(\mathcal{N}TMap)(x)) \mathfrak{G}'x(\mathcal{A}rrVal)(f')$
 $)$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTDom) = \mathfrak{G}'$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTCod) = \mathfrak{G}$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTDGDom) = \mathcal{D}$
and $[adj\text{-}cs\text{-}simps]: cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTDGCod) = \mathcal{C}$
 $\langle proof \rangle$

13.12.2 Natural transformation map

lemma *cf-adj-LR-iso-vsν[adj-cs-intros]*:

$\text{vs}\nu (cf\text{-}adj\text{-}LR\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(NTMap))$
 $\langle proof \rangle$

lemma *cf-adj-RL-iso-vsν[adj-cs-intros]*:

$\text{vs}\nu (cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTMap))$
 $\langle proof \rangle$

lemma *cf-adj-LR-iso-vdomain[adj-cs-simps]*:

$\mathcal{D}_{\circ} (cf\text{-}adj\text{-}LR\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(NTMap)) = \mathcal{C}(\mathcal{O}bj)$
 $\langle proof \rangle$

lemma *cf-adj-RL-iso-vdomain[adj-cs-simps]*:

$\mathcal{D}_{\circ} (cf\text{-}adj\text{-}RL\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(NTMap)) = \mathcal{D}(\mathcal{O}bj)$
 $\langle proof \rangle$

lemma *cf-adj-LR-iso-app*:

fixes $\mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi$
assumes $x \in_{\circ} \mathcal{C}(\mathcal{O}bj)$
defines $\mathfrak{F}x \equiv \mathfrak{F}(\mathcal{O}bjMap)(x)$
and $\mathfrak{F}'x \equiv \mathfrak{F}'(\mathcal{O}bjMap)(x)$
and $\eta \equiv \eta_C \Phi$
and $\eta' \equiv \eta_C \Psi$
shows $cf\text{-}adj\text{-}LR\text{-}iso \mathcal{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(NTMap)(x) =$
 $($
 $\text{THE } f'.$
 $f' : \mathfrak{F}x \mapsto_{\mathcal{D}} \mathfrak{F}'x \wedge$
 $\eta'(\mathcal{N}TMap)(x) = \text{umap-of } \mathfrak{G} x \mathfrak{F}x (\eta(\mathcal{N}TMap)(x)) \mathfrak{F}'x(\mathcal{A}rrVal)(f')$
 $)$
 $\langle proof \rangle$

lemma *cf-adj-RL-iso-app*:

fixes $\mathcal{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi$

assumes $x \in_{\circ} \mathcal{D}(\text{Obj})$
defines $\mathfrak{G}x \equiv \mathfrak{G}(\text{ObjMap})(x)$
and $\mathfrak{G}'x \equiv \mathfrak{G}'(\text{ObjMap})(x)$
and $\varepsilon \equiv \varepsilon_C \Phi$
and $\varepsilon' \equiv \varepsilon_C \Psi$
shows *cf-adj-RL-iso* $\mathfrak{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi(\text{NTMap})(x) =$
(
 THE f' .
 $f' : \mathfrak{G}'x \mapsto_{\mathfrak{C}} \mathfrak{G}x \wedge$
 $\varepsilon'(\text{NTMap})(x) = \text{umap-fo } \mathfrak{F} x \mathfrak{G}x (\varepsilon(\text{NTMap})(x)) \mathfrak{G}'x(\text{ArrVal})(f')$
)
<proof>

lemma *cf-adj-LR-iso-app-unique*:

fixes $\mathfrak{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi$
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
and $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
and $x \in_{\circ} \mathfrak{C}(\text{Obj})$
defines $\mathfrak{F}x \equiv \mathfrak{F}(\text{ObjMap})(x)$
and $\mathfrak{F}'x \equiv \mathfrak{F}'(\text{ObjMap})(x)$
and $\eta \equiv \eta_C \Phi$
and $\eta' \equiv \eta_C \Psi$
and $f \equiv \text{cf-adj-LR-iso } \mathfrak{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi(\text{NTMap})(x)$
shows
 $\exists ! f'$.
 $f' : \mathfrak{F}'x \mapsto_{\mathcal{D}} \mathfrak{F}'x \wedge$
 $\eta'(\text{NTMap})(x) = \text{umap-of } \mathfrak{G} x \mathfrak{F}x (\eta(\text{NTMap})(x)) \mathfrak{F}'x(\text{ArrVal})(f')$
 $f : \mathfrak{F}x \mapsto_{\text{iso}\mathcal{D}} \mathfrak{F}'x$
 $\eta(\text{NTMap})(x) = \text{umap-of } \mathfrak{G} x \mathfrak{F}x (\eta(\text{NTMap})(x)) \mathfrak{F}'x(\text{ArrVal})(f)$
<proof>

13.12.3 Main results

lemma *cf-adj-LR-iso-is-iso-functor*:

— See Corollary 1 in Chapter IV-1 in [9].
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$ **and** $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
shows $\exists ! \vartheta. \vartheta : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathfrak{C} \mapsto_{C\alpha} \mathcal{D} \wedge \eta_C \Psi = (\mathfrak{G} \circ_{CF-NTCF} \vartheta) \cdot_{NTCF} \eta_C \Phi$
and *cf-adj-LR-iso* $\mathfrak{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi : \mathfrak{F} \mapsto_{CF,iso} \mathfrak{F}' : \mathfrak{C} \mapsto_{C\alpha} \mathcal{D}$
and $\eta_C \Psi = (\mathfrak{G} \circ_{CF-NTCF} \text{cf-adj-LR-iso } \mathfrak{C} \mathcal{D} \mathfrak{G} \mathfrak{F} \Phi \mathfrak{F}' \Psi) \cdot_{NTCF} \eta_C \Phi$
<proof>

lemma *op-ntcf-cf-adj-RL-iso[cat-op-simps]*:

assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
and $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
defines $\text{op-}\mathcal{D} \equiv \text{op-cat } \mathcal{D}$
and $\text{op-}\mathfrak{C} \equiv \text{op-cat } \mathfrak{C}$
and $\text{op-}\mathfrak{F} \equiv \text{op-cf } \mathfrak{F}$
and $\text{op-}\mathfrak{G} \equiv \text{op-cf } \mathfrak{G}$
and $\text{op-}\Phi \equiv \text{op-cf-adj } \Phi$
and $\text{op-}\mathfrak{G}' \equiv \text{op-cf } \mathfrak{G}'$
and $\text{op-}\Psi \equiv \text{op-cf-adj } \Psi$
shows
 $\text{op-ntcf } (\text{cf-adj-RL-iso } \mathfrak{C} \mathcal{D} \mathfrak{F} \mathfrak{G} \Phi \mathfrak{G}' \Psi) =$
 $\text{cf-adj-LR-iso } \text{op-}\mathcal{D} \text{op-}\mathfrak{C} \text{op-}\mathfrak{F} \text{op-}\mathfrak{G} \text{op-}\Phi \text{op-}\mathfrak{G}' \text{op-}\Psi$
<proof>

lemma *op-ntcf-cf-adj-LR-iso[cat-op-simps]*:

assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$ **and** $\Psi : \mathfrak{F}' \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{C} \rightleftharpoons_{C\alpha} \mathcal{D}$

defines $op\text{-}\mathcal{D} \equiv op\text{-}cat \ \mathcal{D}$
and $op\text{-}\mathcal{C} \equiv op\text{-}cat \ \mathcal{C}$
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 $op\text{-}\Psi \equiv op\text{-}cf\text{-}adj \ \Psi$
shows
 $op\text{-}ntcf \ (cf\text{-}adj\text{-}LR\text{-}iso \ \mathcal{C} \ \mathcal{D} \ \mathfrak{F} \ \Phi \ \mathfrak{F}' \ \Psi) =$
 $cf\text{-}adj\text{-}RL\text{-}iso \ op\text{-}\mathcal{D} \ op\text{-}\mathcal{C} \ op\text{-}\mathfrak{G} \ op\text{-}\mathfrak{F} \ op\text{-}\Phi \ op\text{-}\mathfrak{F}' \ op\text{-}\Psi$
(proof)

lemma *cf-adj-RL-iso-app-unique:*
fixes $\mathcal{C} \ \mathcal{D} \ \mathfrak{F} \ \mathfrak{G} \ \Phi \ \mathfrak{G}' \ \Psi$
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathcal{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
and $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathcal{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
and $x \in_{\circ} \mathcal{D} (Obj)$
defines $\mathfrak{G}x \equiv \mathfrak{G} (ObjMap) (x)$
and $\mathfrak{G}'x \equiv \mathfrak{G}' (ObjMap) (x)$
and $\varepsilon \equiv \varepsilon_{\mathcal{C}} \ \Phi$
and $\varepsilon' \equiv \varepsilon_{\mathcal{C}} \ \Psi$
and $f \equiv cf\text{-}adj\text{-}RL\text{-}iso \ \mathcal{C} \ \mathcal{D} \ \mathfrak{F} \ \mathfrak{G} \ \Phi \ \mathfrak{G}' \ \Psi (NTMap) (x)$
shows
 $\exists ! f'$
 $f' : \mathfrak{G}'x \mapsto_{\mathcal{C}} \mathfrak{G}x \wedge$
 $\varepsilon' (NTMap) (x) = umap\text{-}fo \ \mathfrak{F} \ x \ \mathfrak{G}x \ (\varepsilon (NTMap) (x)) \ \mathfrak{G}'x (ArrVal) (f')$
 $f : \mathfrak{G}'x \mapsto_{iso\mathcal{C}} \mathfrak{G}x$
 $\varepsilon' (NTMap) (x) = umap\text{-}fo \ \mathfrak{F} \ x \ \mathfrak{G}x \ (\varepsilon (NTMap) (x)) \ \mathfrak{G}'x (ArrVal) (f)$
(proof)

lemma *cf-adj-RL-iso-is-iso-functor:*
— See Corollary 1 in Chapter IV-1 in [9].
assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathcal{C} \rightleftharpoons_{C\alpha} \mathcal{D}$ **and** $\Psi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G}' : \mathcal{C} \rightleftharpoons_{C\alpha} \mathcal{D}$
shows $\exists ! \vartheta$.
 $\vartheta : \mathfrak{G}' \mapsto_{CF} \mathfrak{G} : \mathcal{D} \mapsto_{C\alpha} \mathcal{C} \wedge$
 $\varepsilon_{\mathcal{C}} \ \Psi = \varepsilon_{\mathcal{C}} \ \Phi \cdot_{NTCF} (\mathfrak{F} \circ_{CF\text{-}NTCF} \vartheta)$
and $cf\text{-}adj\text{-}RL\text{-}iso \ \mathcal{C} \ \mathcal{D} \ \mathfrak{F} \ \mathfrak{G} \ \Phi \ \mathfrak{G}' \ \Psi : \mathfrak{G}' \mapsto_{CF.is} \mathfrak{G} : \mathcal{D} \mapsto_{C\alpha} \mathcal{C}$
and $\varepsilon_{\mathcal{C}} \ \Psi =$
 $\varepsilon_{\mathcal{C}} \ \Phi \cdot_{NTCF} (\mathfrak{F} \circ_{CF\text{-}NTCF} cf\text{-}adj\text{-}RL\text{-}iso \ \mathcal{C} \ \mathcal{D} \ \mathfrak{F} \ \mathfrak{G} \ \Phi \ \mathfrak{G}' \ \Psi)$
(proof)

13.13 Further properties of the adjoint functors

lemma (in *is-cf-adjunction*) *cf-adj-exp-cf-cat:*
— See Proposition 4.4.6 in [14].
assumes $\mathcal{Z} \ \beta$ **and** $\alpha \in_{\circ} \beta$ **and** *category* $\alpha \ \mathfrak{J}$
shows
cf-adjunction-of-unit
 β
 $(exp\text{-}cf\text{-}cat \ \alpha \ \mathfrak{F} \ \mathfrak{J})$
 $(exp\text{-}cf\text{-}cat \ \alpha \ \mathfrak{G} \ \mathfrak{J})$
 $(exp\text{-}ntcf\text{-}cat \ \alpha \ (\eta_{\mathcal{C}} \ \Phi) \ \mathfrak{J}) :$
 $exp\text{-}cf\text{-}cat \ \alpha \ \mathfrak{F} \ \mathfrak{J} \rightleftharpoons_{CF} exp\text{-}cf\text{-}cat \ \alpha \ \mathfrak{G} \ \mathfrak{J} :$
 $cat\text{-}FUNCT \ \alpha \ \mathfrak{J} \ \mathcal{C} \rightleftharpoons_{C\beta} cat\text{-}FUNCT \ \alpha \ \mathfrak{J} \ \mathcal{D}$
(proof)

lemma (in *is-cf-adjunction*) *cf-adj-exp-cf-cat-exp-cf-cat:*
— See Proposition 4.4.6 in [14].

assumes $Z \beta$ and $\alpha \in_o \beta$ and category $\alpha \mathfrak{A}$
shows

cf-adjunction-of-unit

β

(*exp-cat-cf* $\alpha \mathfrak{A} \mathfrak{G}$)

(*exp-cat-cf* $\alpha \mathfrak{A} \mathfrak{F}$)

(*exp-cat-ntcf* $\alpha \mathfrak{A} (\eta_C \Phi)$) :

exp-cat-cf $\alpha \mathfrak{A} \mathfrak{G} \rightleftharpoons_{CF} \textit{exp-cat-cf}$ $\alpha \mathfrak{A} \mathfrak{F}$:

cat-FUNCT $\alpha \mathfrak{C} \mathfrak{A} \rightleftharpoons_{C\beta} \textit{cat-FUNCT}$ $\alpha \mathfrak{D} \mathfrak{A}$

<proof>

13.14 Adjoints on limits

lemma *cf-AdjRight-preserves-limits*:

— See Chapter V-5 in [9].

assumes $\Phi : \mathfrak{F} \rightleftharpoons_{CF} \mathfrak{G} : \mathfrak{X} \rightleftharpoons_{C\alpha} \mathfrak{A}$

shows *is-cf-continuous* $\alpha \mathfrak{G}$

<proof>

14 Simple Kan extensions

14.1 Background

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

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

14.2 Kan extension

14.2.1 Definition and elementary properties

See Chapter X-3 in [9].

locale *is-cat-rKe* =

AG: *is-functor* $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{K} +$
Ran: *is-functor* $\alpha \mathfrak{C} \mathfrak{A} \mathfrak{G} +$
ntcf-rKe: *is-ntcf* $\alpha \mathfrak{B} \mathfrak{A} \langle \mathfrak{G} \circ_{CF} \mathfrak{K} \rangle \mathfrak{T} \varepsilon$
for $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{G} \varepsilon +$
assumes *cat-rKe-ua-fo*:
universal-arrow-fo
 (*exp-cat-cf* $\alpha \mathfrak{A} \mathfrak{K}$)
 (*cf-map* \mathfrak{T})
 (*cf-map* \mathfrak{G})
 (*ntcf-arrow* ε)

syntax *-is-cat-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 ($\langle (- \cdot / - \circ_{CF} - \mapsto_{CF.rKe1} - \cdot / - \mapsto_C - \mapsto_C -) \rangle$ [51, 51, 51, 51, 51, 51, 51] 51)

syntax-consts *-is-cat-rKe* \Leftarrow *is-cat-rKe*

translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A} \Leftarrow$
CONST is-cat-rKe $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{G} \varepsilon$

locale *is-cat-lKe* =

AG: *is-functor* $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{K} +$
Lan: *is-functor* $\alpha \mathfrak{C} \mathfrak{A} \mathfrak{F} +$
ntcf-lKe: *is-ntcf* $\alpha \mathfrak{B} \mathfrak{A} \mathfrak{T} \langle \mathfrak{F} \circ_{CF} \mathfrak{K} \rangle \eta$
for $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{F} \eta +$
assumes *cat-lKe-ua-fo*:
universal-arrow-fo
 (*exp-cat-cf* α (*op-cat* \mathfrak{A}) (*op-cf* \mathfrak{K}))
 (*cf-map* \mathfrak{T})
 (*cf-map* \mathfrak{F})
 (*ntcf-arrow* (*op-ntcf* η))

syntax *-is-cat-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 ($\langle (- \cdot / - \mapsto_{CF.lKe1} - \circ_{CF} - \cdot / - \mapsto_C - \mapsto_C -) \rangle$ [51, 51, 51, 51, 51, 51, 51] 51)

syntax-consts *-is-cat-lKe* \Leftarrow *is-cat-lKe*

translations $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A} \Leftarrow$
CONST is-cat-lKe $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{F} \eta$

Rules.

lemma (**in** *is-cat-rKe*) *is-cat-rKe-axioms'*[*cat-Kan-cs-intros*]:

assumes $\alpha' = \alpha$
and $\mathfrak{G}' = \mathfrak{G}$
and $\mathfrak{K}' = \mathfrak{K}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha'} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$

$\langle \text{proof} \rangle$

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 $\eta : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{F}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
 $\langle \text{proof} \rangle$

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 $\mathfrak{T} \mapsto_{CF.lKe\alpha} \text{op-cf } \mathfrak{G} \circ_{CF} \text{op-cf } \mathfrak{K} :$
op-cat $\mathfrak{B} \mapsto_C \text{op-cat } \mathfrak{C} \mapsto_C \text{op-cat } \mathfrak{A}$
 $\langle \text{proof} \rangle$

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

assumes $\mathfrak{T}' = \text{op-cf } \mathfrak{T}$
and $\mathfrak{G}' = \text{op-cf } \mathfrak{G}$
and $\mathfrak{K}' = \text{op-cf } \mathfrak{K}$
and $\mathfrak{B}' = \text{op-cat } \mathfrak{B}$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$
and $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
shows *op-ntcf* $\varepsilon : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
 $\langle \text{proof} \rangle$

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 $\mathfrak{F} \circ_{CF} \text{op-cf } \mathfrak{K} \mapsto_{CF.rKe\alpha} \text{op-cf } \mathfrak{T} :$
op-cat $\mathfrak{B} \mapsto_C \text{op-cat } \mathfrak{C} \mapsto_C \text{op-cat } \mathfrak{A}$
 $\langle \text{proof} \rangle$

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

assumes $\mathfrak{T}' = \text{op-cf } \mathfrak{T}$
and $\mathfrak{F}' = \text{op-cf } \mathfrak{F}$
and $\mathfrak{K}' = \text{op-cf } \mathfrak{K}$
and $\mathfrak{B}' = \text{op-cat } \mathfrak{B}$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$

and $\mathcal{C}' = \text{op-cat } \mathcal{C}$
 shows $\text{op-ntcf } \eta : \mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathcal{C}' \mapsto_C \mathfrak{A}'$
 ⟨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 $Z \beta$ and $\alpha \in_o \beta$
 shows $\text{exp-cat-cf } \alpha \mathfrak{A} \mathfrak{K} : \text{cat-FUNCT } \alpha \mathcal{C} \mathfrak{A} \mapsto_{C.tiny\beta} \text{cat-FUNCT } \alpha \mathfrak{B} \mathfrak{A}$
 ⟨proof⟩

lemma (in is-cat-lKe) cat-lKe-exp-cat-cf-cat-FUNCT-is-arr:
 assumes $Z \beta$ and $\alpha \in_o \beta$
 shows $\text{exp-cat-cf } \alpha \mathfrak{A} \mathfrak{K} : \text{cat-FUNCT } \alpha \mathcal{C} \mathfrak{A} \mapsto_{C.tiny\beta} \text{cat-FUNCT } \alpha \mathfrak{B} \mathfrak{A}$
 ⟨proof⟩

14.2.2 Universal property

See Chapter X-3 in [9] and [2]⁸.

lemma is-cat-rKeI':
 assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathcal{C}$
 and $\mathcal{G} : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}$
 and $\varepsilon : \mathcal{G} \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 and $\wedge \mathcal{G}' \varepsilon'$.

$$\llbracket \mathcal{G}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}; \varepsilon' : \mathcal{G}' \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A} \rrbracket \implies$$

$$\exists! \sigma. \sigma : \mathcal{G}' \mapsto_{CF} \mathcal{G} : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \cdot_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$$

 shows $\varepsilon : \mathcal{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathcal{C} \mapsto_C \mathfrak{A}$
 ⟨proof⟩

lemma is-cat-lKeI':
 assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathcal{C}$
 and $\mathfrak{F} : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}$
 and $\eta : \mathfrak{T} \mapsto_{CF} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 and $\wedge \mathfrak{F}' \eta'$.

$$\llbracket \mathfrak{F}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}; \eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A} \rrbracket \implies$$

$$\exists! \sigma. \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \cdot_{NTCF} \eta$$

 shows $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathcal{C} \mapsto_C \mathfrak{A}$
 ⟨proof⟩

lemma (in is-cat-rKe) cat-rKe-unique:
 assumes $\mathcal{G}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}$ and $\varepsilon' : \mathcal{G}' \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 shows $\exists! \sigma. \sigma : \mathcal{G}' \mapsto_{CF} \mathcal{G} : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A} \wedge \varepsilon' = \varepsilon \cdot_{NTCF} (\sigma \circ_{NTCF-CF} \mathfrak{K})$
 ⟨proof⟩

lemma (in is-cat-lKe) cat-lKe-unique:
 assumes $\mathfrak{F}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A}$ and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 shows $\exists! \sigma. \sigma : \mathfrak{F} \mapsto_{CF} \mathfrak{F}' : \mathcal{C} \mapsto_{C\alpha} \mathfrak{A} \wedge \eta' = (\sigma \circ_{NTCF-CF} \mathfrak{K}) \cdot_{NTCF} \eta$
 ⟨proof⟩

14.2.3 Further properties

lemma (in is-cat-rKe) cat-rKe-ntcf-ua-fo-is-iso-ntcf-if-ge-Limit:
 assumes $Z \beta$ and $\alpha \in_o \beta$
 shows

⁸https://en.wikipedia.org/wiki/Kan_extension

$ntcf\text{-}ua\text{-}fo \beta (exp\text{-}cat\text{-}cf \alpha \mathfrak{A} \mathfrak{K}) (cf\text{-}map \mathfrak{T}) (cf\text{-}map \mathfrak{G}) (ntcf\text{-}arrow \varepsilon) :$
 $Hom_{O.C\beta} cat\text{-}FUNCT \alpha \mathfrak{C} \mathfrak{A}(-, cf\text{-}map \mathfrak{G}) \mapsto_{CF.iso}$
 $Hom_{O.C\beta} cat\text{-}FUNCT \alpha \mathfrak{B} \mathfrak{A}(-, cf\text{-}map \mathfrak{T}) \circ_{CF} op\text{-}cf (exp\text{-}cat\text{-}cf \alpha \mathfrak{A} \mathfrak{K}) :$
 $op\text{-}cat (cat\text{-}FUNCT \alpha \mathfrak{C} \mathfrak{A}) \mapsto_{\mapsto} C\beta \text{ cat}\text{-}Set \beta$
 ⟨proof⟩

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

assumes $Z \beta$ **and** $\alpha \in_o \beta$

defines $\mathfrak{A}\mathfrak{K} \equiv exp\text{-}cat\text{-}cf \alpha (op\text{-}cat \mathfrak{A}) (op\text{-}cf \mathfrak{K})$

and $\mathfrak{A}\mathfrak{C} \equiv cat\text{-}FUNCT \alpha (op\text{-}cat \mathfrak{C}) (op\text{-}cat \mathfrak{A})$

and $\mathfrak{A}\mathfrak{B} \equiv cat\text{-}FUNCT \alpha (op\text{-}cat \mathfrak{B}) (op\text{-}cat \mathfrak{A})$

shows

$ntcf\text{-}ua\text{-}fo \beta \mathfrak{A}\mathfrak{K} (cf\text{-}map \mathfrak{T}) (cf\text{-}map \mathfrak{G}) (ntcf\text{-}arrow (op\text{-}ntcf \eta)) :$

$Hom_{O.C\beta} \mathfrak{A}\mathfrak{C}(-, cf\text{-}map \mathfrak{G}) \mapsto_{CF.iso} Hom_{O.C\beta} \mathfrak{A}\mathfrak{B}(-, cf\text{-}map \mathfrak{T}) \circ_{CF} op\text{-}cf \mathfrak{A}\mathfrak{K} :$

$op\text{-}cat \mathfrak{A}\mathfrak{C} \mapsto_{\mapsto} C\beta \text{ cat}\text{-}Set \beta$

⟨proof⟩

14.3 Opposite universal arrow for Kan extensions

14.3.1 Definition and elementary properties

The following definition is merely a convenience utility for the exposition of dual results associated with the formula for the right Kan extension and the pointwise right Kan extension.

definition $op\text{-}ua :: (V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$

where $op\text{-}ua \text{ lim-Obj } \mathfrak{K} \ c =$

[
 $\text{lim-Obj } c(\text{UObj}),$
 $op\text{-}ntcf (\text{lim-Obj } c(\text{UArr})) \circ_{NTCF-CF} inv\text{-}cf (op\text{-}cf\text{-}obj\text{-}comma \mathfrak{K} \ c)$
]_o

Components.

lemma *op-ua-components*:

shows [cat-op-simps]: $op\text{-}ua \text{ lim-Obj } \mathfrak{K} \ c(\text{UObj}) = \text{lim-Obj } c(\text{UObj})$

and $op\text{-}ua \text{ lim-Obj } \mathfrak{K} \ c(\text{UArr}) =$

$op\text{-}ntcf (\text{lim-Obj } c(\text{UArr})) \circ_{NTCF-CF} inv\text{-}cf (op\text{-}cf\text{-}obj\text{-}comma \mathfrak{K} \ c)$

⟨proof⟩

14.3.2 Opposite universal arrow for Kan extensions is a limit

lemma *op-ua-UArr-is-cat-limit*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{\mapsto} C\alpha \mathfrak{C}$

and $\mathfrak{T} : \mathfrak{B} \mapsto_{\mapsto} C\alpha \mathfrak{A}$

and $c \in_o \mathfrak{C}(\text{Obj})$

and $u : \mathfrak{T} \circ_{CF} \mathfrak{K} \text{ CF} \sqcap_O c >_{CF.colim} r : \mathfrak{K} \text{ CF} \downarrow c \mapsto_{\mapsto} C\alpha \mathfrak{A}$

shows $op\text{-}ntcf \ u \circ_{NTCF-CF} inv\text{-}cf (op\text{-}cf\text{-}obj\text{-}comma \mathfrak{K} \ c) :$

$r <_{CF.lim} op\text{-}cf \ \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} (op\text{-}cf \ \mathfrak{K}) : c \downarrow_{CF} (op\text{-}cf \ \mathfrak{K}) \mapsto_{\mapsto} C\alpha \text{ op-cat } \mathfrak{A}$

⟨proof⟩

context

fixes $\text{lim-Obj} :: V \Rightarrow V$ **and** $c :: V$

begin

lemmas $op\text{-}ua\text{-}UArr\text{-}is\text{-}cat\text{-}limit' = op\text{-}ua\text{-}UArr\text{-}is\text{-}cat\text{-}limit$

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

]

end

14.4 The Kan extension

The following subsection is based on the statement and proof of Theorem 1 in Chapter X-3 in [9].

14.4.1 Definition and elementary properties

definition *the-cf-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where *the-cf-rKe* α \mathfrak{I} \mathfrak{K} *lim-Obj* =

[
 ($\lambda c \in_{\circ} \mathfrak{K}(\text{HomCod})(\text{Obj})$). *lim-Obj* $c(\text{UObj})$),
 (
 $\lambda g \in_{\circ} \mathfrak{K}(\text{HomCod})(\text{Arr})$. THE f .
 f :
 $\text{lim-Obj } (\mathfrak{K}(\text{HomCod})(\text{Dom})(g))(\text{UObj}) \mapsto_{\mathfrak{I}}(\text{HomCod})$
 $\text{lim-Obj } (\mathfrak{K}(\text{HomCod})(\text{Cod})(g))(\text{UObj}) \wedge$
 $\text{lim-Obj } (\mathfrak{K}(\text{HomCod})(\text{Dom})(g))(\text{UArr}) \circ_{NTCF-CF} g \downarrow_{CF} \mathfrak{K} =$
 $\text{lim-Obj } (\mathfrak{K}(\text{HomCod})(\text{Cod})(g))(\text{UArr}) \cdot_{NTCF}$
 $\text{ntcf-const } ((\mathfrak{K}(\text{HomCod})(\text{Cod})(g)) \downarrow_{CF} \mathfrak{K}) (\mathfrak{I}(\text{HomCod})) f$
),
 $\mathfrak{K}(\text{HomCod})$,
 $\mathfrak{I}(\text{HomCod})$
] \circ .

definition *the-ntcf-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where *the-ntcf-rKe* α \mathfrak{I} \mathfrak{K} *lim-Obj* =

[
 (
 $\lambda c \in_{\circ} \mathfrak{I}(\text{HomDom})(\text{Obj})$.
 $\text{lim-Obj } (\mathfrak{K}(\text{ObjMap})(c))(\text{UArr})(\text{NTMap})(\theta, c, \mathfrak{K}(\text{HomCod})(\text{CId})(\mathfrak{K}(\text{ObjMap})(c)))$).
),
 $\text{the-cf-rKe } \alpha \mathfrak{I} \mathfrak{K} \text{lim-Obj } \circ_{CF} \mathfrak{K}$,
 \mathfrak{I} ,
 $\mathfrak{I}(\text{HomDom})$,
 $\mathfrak{I}(\text{HomCod})$
] \circ .

definition *the-cf-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where *the-cf-lKe* α \mathfrak{I} \mathfrak{K} *lim-Obj* =

$\text{op-cf } (\text{the-cf-rKe } \alpha \text{ (op-cf } \mathfrak{I}) \text{ (op-cf } \mathfrak{K}) \text{ (op-ua lim-Obj } \mathfrak{K}))$

definition *the-ntcf-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

where *the-ntcf-lKe* α \mathfrak{I} \mathfrak{K} *lim-Obj* =

$\text{op-ntcf } (\text{the-ntcf-rKe } \alpha \text{ (op-cf } \mathfrak{I}) \text{ (op-cf } \mathfrak{K}) \text{ (op-ua lim-Obj } \mathfrak{K}))$

Components.

lemma *the-cf-rKe-components*:

shows *the-cf-rKe* α \mathfrak{I} \mathfrak{K} *lim-Obj*(*ObjMap*) =

$(\lambda c \in_{\circ} \mathfrak{K}(\text{HomCod})(\text{Obj})$. *lim-Obj* $c(\text{UObj})$)

and *the-cf-rKe* α \mathfrak{I} \mathfrak{K} *lim-Obj*(*ArrMap*) =

(
 $\lambda g \in_{\circ} \mathfrak{K}(\text{HomCod})(\text{Arr})$. THE f .
 f :

$lim-Obj (\mathfrak{K}(HomCod)(Dom)(g))(UObj) \mapsto_{\mathfrak{T}}(HomCod)$
 $lim-Obj (\mathfrak{K}(HomCod)(Cod)(g))(UObj) \wedge$
 $lim-Obj (\mathfrak{K}(HomCod)(Dom)(g))(UArr) \circ_{NTCF-CF} g \downarrow_{CF} \mathfrak{K} =$
 $lim-Obj (\mathfrak{K}(HomCod)(Cod)(g))(UArr) \cdot_{NTCF}$
 $ntcf-const ((\mathfrak{K}(HomCod)(Cod)(g)) \downarrow_{CF} \mathfrak{K}) (\mathfrak{T}(HomCod)) f$
)

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* $\alpha \mathfrak{T} \mathfrak{K} lim-Obj(NTMap) =$
(
 $\lambda c \in_{\circ} \mathfrak{T}(HomDom)(Obj).$
 $lim-Obj (\mathfrak{K}(ObjMap)(c))(UArr)(NTMap)(\theta, c, \mathfrak{K}(HomCod)(CId)(\mathfrak{K}(ObjMap)(c)))$
)

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 $\mathfrak{K}: \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

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** $\alpha=\alpha$, *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** $\alpha=\alpha$, *unfolded* $\mathfrak{K}.cf-HomCod$ $\mathfrak{T}.cf-HomCod$ $\mathfrak{T}.cf-HomDom$
]

lemmas [*cat-Kan-cs-simps*] = *the-ntcf-rKe-components'*(2-5)

end

14.4.2 Functor: object map

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

context

fixes $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{C} \mathfrak{K} \mathfrak{T}$
assumes $\mathfrak{K}: \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation \mathfrak{K} : *is-functor* $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{K}$ *<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 $\wedge c. c \in_0 \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UObj}) \in_0 \mathfrak{A}(\text{Obj})$
 shows \mathcal{R}_0 (*the-cf-rKe* α \mathfrak{T} \mathfrak{K} *lim-Obj*(*ObjMap*)) $\subseteq_0 \mathfrak{A}(\text{Obj})$
 ⟨*proof*⟩

end

14.4.3 Functor: arrow map

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

context

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

begin

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

mk-VLambda *the-cf-rKe-components(2)*[**where** $\alpha=\alpha$ **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 $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 and $g: g : c \mapsto_{\mathfrak{C}} c'$

begin

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

lemma $g': g \in_0 \mathfrak{C}(\text{Arr})$ ⟨*proof*⟩

mk-VLambda *the-cf-rKe-components(2)*[
 where $\alpha=\alpha$ **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 $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
 and $g : c \mapsto_{\mathfrak{C}} c'$
 and $u : r <_{CF.lim} \mathfrak{T} \circ_{CF} c \text{ o } \prod_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto_{C\alpha} \mathfrak{A}$
 and $u' : r' <_{CF.lim} \mathfrak{T} \circ_{CF} c' \text{ o } \prod_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto_{C\alpha} \mathfrak{A}$

shows $\exists! f$.

$f : r \mapsto_{\mathfrak{A}} r' \wedge$
 $u \circ_{NTCF-CF} g \text{ A } \downarrow_{CF} \mathfrak{K} = u' \cdot_{NTCF} \text{ntcf-const } (c' \downarrow_{CF} \mathfrak{K}) \mathfrak{A} f$

<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 $\text{lim-Obj } c(\text{UArr}) :$

$\text{lim-Obj } c(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $\text{lim-Obj } c'(\text{UArr}) :$

$\text{lim-Obj } c'(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c' \circ \sqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

shows *the-cf-rKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj}(\text{ArrMap})(g) :$

$\text{lim-Obj } c(\text{UObj}) \mapsto_{\mathfrak{A}} \text{lim-Obj } c'(\text{UObj})$

and

$\text{lim-Obj } c(\text{UArr}) \circ_{NTCF-CF} g \downarrow_{CF} \mathfrak{K} =$

$\text{lim-Obj } c'(\text{UArr}) \cdot_{NTCF}$

$\text{ntcf-const } (c' \downarrow_{CF} \mathfrak{K}) \mathfrak{A} (\text{the-cf-rKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj}(\text{ArrMap})(g))$

and

$f : \text{lim-Obj } c(\text{UObj}) \mapsto_{\mathfrak{A}} \text{lim-Obj } c'(\text{UObj});$

$\text{lim-Obj } c(\text{UArr}) \circ_{NTCF-CF} g \downarrow_{CF} \mathfrak{K} =$

$\text{lim-Obj } c'(\text{UArr}) \cdot_{NTCF} \text{ntcf-const } (c' \downarrow_{CF} \mathfrak{K}) \mathfrak{A} f$

$\implies f = \text{the-cf-rKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj}(\text{ArrMap})(g)$

<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 $\text{lim-Obj } c(\text{UArr}) :$

$\text{lim-Obj } c(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $\text{lim-Obj } c'(\text{UArr}) :$

$\text{lim-Obj } c'(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c' \circ \sqcap_{CF} \mathfrak{K} : c' \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $a = \text{lim-Obj } c(\text{UObj})$

and $b = \text{lim-Obj } c'(\text{UObj})$

shows *the-cf-rKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj}(\text{ArrMap})(g) : a \mapsto_{\mathfrak{A}} b$

<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 $\text{lim-Obj } a(\text{UArr}) :$

$\text{lim-Obj } a(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} a \circ \sqcap_{CF} \mathfrak{K} : a \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $\text{lim-Obj } b(\text{UArr}) :$

$\text{lim-Obj } b(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} b \circ \sqcap_{CF} \mathfrak{K} : b \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $f : a \mapsto_{\mathfrak{C}} b$

and $[a', b', f']_{\circ} \in_{\circ} b \downarrow_{CF} \mathfrak{K}(\text{Obj})$

shows

$\text{lim-Obj } a(\text{UArr})(\text{NTMap})(a', b', f' \circ_{A\mathfrak{C}} f)_{\bullet} =$

$\text{lim-Obj } b(\text{UArr})(\text{NTMap})(a', b', f')_{\bullet} \circ_{A\mathfrak{A}}$

$\text{the-cf-rKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj}(\text{ArrMap})(f)$

<proof>

14.4.4 Natural transformation: natural transformation map

mk-VLambda *the-ntcf-rKe-components(1)*

$[\text{vsu the-ntcf-rKe-NTMap-vsuv}[\text{cat-Kan-cs-intros}]]$

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* $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{K}$ *<proof>*

interpretation \mathfrak{T} : *is-functor* $\alpha \mathfrak{B} \mathfrak{A} \mathfrak{T}$ *<proof>*

mk-VLambda *the-ntcf-rKe-components'(1)[OF $\mathfrak{K} \mathfrak{T}$]*
[vdomain the-ntcf-rKe-ObjMap-vdomain[cat-Kan-cs-simps]]
[app the-ntcf-rKe-ObjMap-impl-app[cat-Kan-cs-simps]]

end

14.4.5 The Kan extension is a Kan extension

lemma *the-cf-rKe-is-functor*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $\bigwedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\text{lim-Obj } c(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 shows *the-cf-rKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<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 $\bigwedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\mathfrak{T} \circ_{CF} \mathfrak{K} \text{CF} \sqcap_{O} c >_{CF.colim} \text{lim-Obj } c(\text{UObj}) : \mathfrak{K} \text{CF} \downarrow c \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 shows *the-cf-lKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} : \mathfrak{C} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<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 $\bigwedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\text{lim-Obj } c(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 shows *the-ntcf-rKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} :$
 $\text{the-cf-rKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} \circ_{CF} \mathfrak{K} \mapsto_{CF} \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<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 $\bigwedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\mathfrak{T} \circ_{CF} \mathfrak{K} \text{CF} \sqcap_{O} c >_{CF.colim} \text{lim-Obj } c(\text{UObj}) : \mathfrak{K} \text{CF} \downarrow c \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 shows *the-ntcf-lKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} :$
 $\mathfrak{T} \mapsto_{CF} \text{the-cf-lKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<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 $\bigwedge c. c \in_{\circ} \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\text{lim-Obj } c(\text{UObj}) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 shows *the-ntcf-rKe* $\alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} :$
 $\text{the-cf-rKe } \alpha \mathfrak{T} \mathfrak{K} \text{lim-Obj} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
<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 $\bigwedge c. c \in_o \mathfrak{C}(\text{Obj}) \implies \text{lim-Obj } c(\text{UArr}) :$
 $\mathfrak{T} \circ_{CF} \mathfrak{K} \text{ CF} \prod_O c >_{CF} \text{colim } \text{lim-Obj } c(\text{UObj}) : \mathfrak{K} \text{ CF} \downarrow c \mapsto \mapsto_{C\alpha} \mathfrak{A}$
shows *the-ntcf-lKe* α $\mathfrak{T} \mathfrak{K} \text{ lim-Obj}$:
 $\mathfrak{T} \mapsto_{CF.lKe\alpha} \text{the-cf-lKe } \alpha \mathfrak{T} \mathfrak{K} \text{ lim-Obj} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$
(proof)

14.5 Preservation of Kan extensions

The following definitions are similar to the definitions that can be found in [14] or [8].

locale *is-cat-rKe-preserves* =

is-cat-rKe α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{G} \varepsilon$ + *is-functor* α $\mathfrak{A} \mathfrak{D} \mathfrak{H}$
for α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{D} \mathfrak{K} \mathfrak{T} \mathfrak{G} \mathfrak{H} \varepsilon$ +

assumes *cat-rKe-preserves*:

$\mathfrak{H} \circ_{CF-NTCF} \varepsilon : (\mathfrak{H} \circ_{CF} \mathfrak{G}) \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{H} \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

syntax *-is-cat-rKe-preserves* ::

$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 (
 $\langle (- : / - \circ_{CF} - \mapsto_{CF.rKe\alpha} - : / - \mapsto_C - \mapsto_C - : - \mapsto \mapsto_C -) \rangle$
 $[51, 51, 51, 51, 51, 51, 51, 51, 51, 51]$ 51
)

syntax-consts *-is-cat-rKe-preserves* \equiv *is-cat-rKe-preserves*

translations $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto \mapsto_C \mathfrak{D}) \equiv$

CONST is-cat-rKe-preserves α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{D} \mathfrak{K} \mathfrak{T} \mathfrak{G} \mathfrak{H} \varepsilon$

locale *is-cat-lKe-preserves* =

is-cat-lKe α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} \mathfrak{F} \eta$ + *is-functor* α $\mathfrak{A} \mathfrak{D} \mathfrak{H}$
for α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{D} \mathfrak{K} \mathfrak{T} \mathfrak{F} \mathfrak{H} \eta$ +

assumes *cat-lKe-preserves*:

$\mathfrak{H} \circ_{CF-NTCF} \eta : \mathfrak{H} \circ_{CF} \mathfrak{T} \mapsto_{CF.lKe\alpha} (\mathfrak{H} \circ_{CF} \mathfrak{F}) \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

syntax *-is-cat-lKe-preserves* ::

$V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
 (
 $\langle (- : / - \mapsto_{CF.lKe\alpha} - \circ_{CF} - : / - \mapsto_C - \mapsto_C - : - \mapsto \mapsto_C -) \rangle$
 $[51, 51, 51, 51, 51, 51, 51, 51, 51, 51]$ 51
)

syntax-consts *-is-cat-lKe-preserves* \equiv *is-cat-lKe-preserves*

translations $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H} : \mathfrak{A} \mapsto \mapsto_C \mathfrak{D}) \equiv$

CONST is-cat-lKe-preserves α $\mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{D} \mathfrak{K} \mathfrak{T} \mathfrak{F} \mathfrak{H} \eta$

Rules.

lemma (in *is-cat-rKe-preserves*) *is-cat-rKe-preserves-axioms'*:

assumes $\alpha' = \alpha$

and $\mathfrak{G}' = \mathfrak{G}$

and $\mathfrak{K}' = \mathfrak{K}$

and $\mathfrak{T}' = \mathfrak{T}$

and $\mathfrak{H}' = \mathfrak{H}$

and $\mathfrak{B}' = \mathfrak{B}$

and $\mathfrak{A}' = \mathfrak{A}$

and $\mathfrak{C}' = \mathfrak{C}$

and $\mathfrak{D}' = \mathfrak{D}$

shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha'} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto \mapsto_C \mathfrak{D}')$

$\langle \text{proof} \rangle$

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 $\eta : \mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{F}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$

$\langle \text{proof} \rangle$

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 $\mathfrak{T} \mapsto_{CF.lKe\alpha} \text{op-cf } \mathfrak{G} \circ_{CF} \text{op-cf } \mathfrak{K} :$

op-cat $\mathfrak{B} \mapsto_C \text{op-cat } \mathfrak{C} \mapsto_C (\text{op-cf } \mathfrak{H} : \text{op-cat } \mathfrak{A} \mapsto\mapsto_C \text{op-cat } \mathfrak{D})$

$\langle \text{proof} \rangle$

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

assumes $\mathfrak{T}' = \text{op-cf } \mathfrak{T}$

and $\mathfrak{G}' = \text{op-cf } \mathfrak{G}$

and $\mathfrak{K}' = \text{op-cf } \mathfrak{K}$

and $\mathfrak{B}' = \text{op-cat } \mathfrak{B}$

and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$

and $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$

and $\mathfrak{D}' = \text{op-cat } \mathfrak{D}$

and $\mathfrak{H}' = \text{op-cf } \mathfrak{H}$

shows *op-ntcf* ε :

$\mathfrak{T}' \mapsto_{CF.lKe\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{K}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto\mapsto_C \mathfrak{D}')$

$\langle \text{proof} \rangle$

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 $\mathfrak{F} \circ_{CF} \text{op-cf } \mathfrak{K} \mapsto_{CF.\tauKe\alpha} \text{op-cf } \mathfrak{T} :$

op-cat $\mathfrak{B} \mapsto_C \text{op-cat } \mathfrak{C} \mapsto_C (\text{op-cf } \mathfrak{H} : \text{op-cat } \mathfrak{A} \mapsto\mapsto_C \text{op-cat } \mathfrak{D})$

$\langle \text{proof} \rangle$

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

assumes $\mathfrak{T}' = \text{op-cf } \mathfrak{T}$
and $\mathfrak{F}' = \text{op-cf } \mathfrak{F}$
and $\mathfrak{K}' = \text{op-cf } \mathfrak{K}$
and $\mathfrak{H}' = \text{op-cf } \mathfrak{H}$
and $\mathfrak{B}' = \text{op-cat } \mathfrak{B}$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$
and $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
and $\mathfrak{D}' = \text{op-cat } \mathfrak{D}$
shows $\text{op-ntcf } \eta :$
 $\mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto_C \mathfrak{D}')$
<proof>

14.6 All concepts are Kan extensions

Background information for this subsection is provided in Chapter X-7 in [9] and subsection 6.5 in [14]. It should be noted that only the connections between the Kan extensions, limits and adjunctions are exposed (an alternative proof of the Yoneda lemma using Kan extensions is not provided in the context of this work).

14.6.1 Limits and colimits

lemma *cat-rKe-is-cat-limit:*

— The statement of the theorem is similar to the statement of a part of Theorem 1 in Chapter X-7 in [9] or Proposition 6.5.1 in [14].

assumes $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
and $\mathfrak{T} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
shows $\varepsilon : \mathfrak{G}(\text{ObjMap})(\mathfrak{a}) <_{CF.lim} \mathfrak{T} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
<proof>

lemma *cat-lKe-is-cat-colimit:*

assumes $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
and $\mathfrak{T} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
shows $\eta : \mathfrak{T} >_{CF.colim} \mathfrak{F}(\text{ObjMap})(\mathfrak{a}) : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
<proof>

lemma *cat-limit-is-rKe:*

— The statement of the theorem is similar to the statement of a part of Theorem 1 in Chapter X-7 in [9] or Proposition 6.5.1 in [14].

assumes $\varepsilon : \mathfrak{G}(\text{ObjMap})(\mathfrak{a}) <_{CF.lim} \mathfrak{T} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
and $\mathfrak{G} : \text{cat-1 } \mathfrak{a} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
shows $\varepsilon : \mathfrak{G} \circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{T} : \mathfrak{B} \mapsto_C \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
<proof>

lemma *cat-colimit-is-lKe:*

assumes $\eta : \mathfrak{T} >_{CF.colim} \mathfrak{F}(\text{ObjMap})(\mathfrak{a}) : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto_{\mapsto_{C\alpha}} \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
and $\mathfrak{F} : \text{cat-1 } \mathfrak{a} \mapsto_{\mapsto_{C\alpha}} \mathfrak{A}$
shows $\eta : \mathfrak{T} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathfrak{K} : \mathfrak{B} \mapsto_C \text{cat-1 } \mathfrak{a} \mapsto_C \mathfrak{A}$
<proof>

14.6.2 Adjoints

lemma (in *is-cf-adjunction*) *cf-adjunction-counit-is-rKe:*

— The statement of the theorem is similar to the statement of a part of Theorem 2 in Chapter X-7 in [9] or Proposition 6.5.2 in [14]. The proof follows (approximately) the proof in [14].

shows $\varepsilon_C \Phi : \mathfrak{F} \circ_{CF} \mathfrak{G} \mapsto_{CF.rKe\alpha} \text{cf-id } \mathfrak{D} : \mathfrak{D} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{D}$

<proof>

lemma (in *is-cf-adjunction*) *cf-adjunction-unit-is-lKe*:

shows $\eta_C \Phi : \text{cf-id } \mathcal{C} \mapsto_{CF.lKe\alpha} \mathcal{G} \circ_{CF} \mathfrak{F} : \mathcal{C} \mapsto_C \mathcal{D} \mapsto_C \mathcal{C}$

<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 [9] or Proposition 6.5.2 in [14].

assumes $\eta : \text{cf-id } \mathcal{D} \mapsto_{CF.lKe\alpha} \mathfrak{F} \circ_{CF} \mathcal{G} : \mathcal{D} \mapsto_C \mathcal{C} \mapsto_C (\mathcal{G} : \mathcal{D} \mapsto\mapsto_C \mathcal{C})$

shows *cf-adjunction-of-unit* $\alpha \mathcal{G} \mathfrak{F} \eta : \mathcal{G} \rightleftharpoons_{CF} \mathfrak{F} : \mathcal{D} \rightleftharpoons_{C\alpha} \mathcal{C}$

<proof>

lemma *cf-adjunction-if-rKe-preserves*:

assumes $\varepsilon : \mathfrak{F} \circ_{CF} \mathcal{G} \mapsto_{CF.rKe\alpha} \text{cf-id } \mathcal{D} : \mathcal{D} \mapsto_C \mathcal{C} \mapsto_C (\mathcal{G} : \mathcal{D} \mapsto\mapsto_C \mathcal{C})$

shows *cf-adjunction-of-counit* $\alpha \mathfrak{F} \mathcal{G} \varepsilon : \mathfrak{F} \rightleftharpoons_{CF} \mathcal{G} : \mathcal{C} \rightleftharpoons_{C\alpha} \mathcal{D}$

<proof>

15 Pointwise Kan extensions

15.1 Pointwise Kan extensions

The following subsection is based on elements of the content of section 6.3 in [14] and Chapter X-5 in [9].

locale *is-cat-pw-rKe* = *is-cat-rKe* α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{G} ε
for α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{G} ε +
assumes *cat-pw-rKe-preserved*: $a \in_{\circ} \mathfrak{A}(\text{Obj}) \implies$
 ε :
 $\mathfrak{G} \circ_{CF} \mathfrak{R} \mapsto_{CF.rKe\alpha} \mathfrak{T}$:
 $\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\text{Hom}_{O.C\alpha} \mathfrak{A}(a, -) : \mathfrak{A} \mapsto_C \text{cat-Set } \alpha)$

syntax *-is-cat-pw-rKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(
 $\langle (- : / - \circ_{CF} - \mapsto_{CF.rKe.pw1} - : / - \mapsto_C - \mapsto_C -) \rangle$
[51, 51, 51, 51, 51, 51, 51] 51
))

syntax-consts *-is-cat-pw-rKe* \equiv *is-cat-pw-rKe*
translations ε : $\mathfrak{G} \circ_{CF} \mathfrak{R} \mapsto_{CF.rKe.pw\alpha} \mathfrak{T}$: $\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A} \equiv$
CONST is-cat-pw-rKe α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{G} ε

locale *is-cat-pw-lKe* = *is-cat-lKe* α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{F} η
for α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{F} η +
assumes *cat-pw-lKe-preserved*: $a \in_{\circ} \text{op-cat } \mathfrak{A}(\text{Obj}) \implies$
op-ntcf η :
op-cf $\mathfrak{F} \circ_{CF} \text{op-cf } \mathfrak{R} \mapsto_{CF.rKe\alpha} \text{op-cf } \mathfrak{T}$:
op-cat $\mathfrak{B} \mapsto_C \text{op-cat } \mathfrak{C} \mapsto_C (\text{Hom}_{O.C\alpha} \mathfrak{A}(-, a) : \text{op-cat } \mathfrak{A} \mapsto_C \text{cat-Set } \alpha)$

syntax *-is-cat-pw-lKe* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(
 $\langle (- : / - \mapsto_{CF.lKe.pw1} - \circ_{CF} - : / - \mapsto_C - \mapsto_C -) \rangle$
[51, 51, 51, 51, 51, 51, 51] 51
))

syntax-consts *-is-cat-pw-lKe* \equiv *is-cat-pw-lKe*
translations η : $\mathfrak{T} \mapsto_{CF.lKe.pw\alpha} \mathfrak{F} \circ_{CF} \mathfrak{R}$: $\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A} \equiv$
CONST is-cat-pw-lKe α \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{R} \mathfrak{T} \mathfrak{F} η

lemma (in *is-cat-pw-rKe*) *cat-pw-rKe-preserved'*[*cat-Kan-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{A}(\text{Obj})$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{H}' = \text{Hom}_{O.C\alpha} \mathfrak{A}(a, -)$
and $\mathfrak{C}' = \text{cat-Set } \alpha$
shows ε : $\mathfrak{G} \circ_{CF} \mathfrak{R} \mapsto_{CF.rKe\alpha} \mathfrak{T}$: $\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto_C \mathfrak{C}')$
<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 $a \in_{\circ} \text{op-cat } \mathfrak{A}(\text{Obj})$
and $\mathfrak{F}' = \text{op-cf } \mathfrak{F}$
and $\mathfrak{R}' = \text{op-cf } \mathfrak{R}$
and $\mathfrak{T}' = \text{op-cf } \mathfrak{T}$
and $\mathfrak{B}' = \text{op-cat } \mathfrak{B}$
and $\mathfrak{C}' = \text{op-cat } \mathfrak{C}$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$
and $\mathfrak{H}' = \text{Hom}_{O.C\alpha} \mathfrak{A}(-, a)$
and $\mathfrak{C}' = \text{cat-Set } \alpha$

shows *op-ntcf* η :
 $\mathfrak{F}' \circ_{CF} \mathfrak{R}' \mapsto_{CF.rKe\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C (\mathfrak{H}' : \mathfrak{A}' \mapsto_C \mathfrak{E}')$
 ⟨*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 $\mathfrak{R}' = \mathfrak{R}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\varepsilon : \mathfrak{G}' \circ_{CF} \mathfrak{R}' \mapsto_{CF.rKe.pw\alpha'} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
 ⟨*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 $\mathfrak{R}' = \mathfrak{R}$
and $\mathfrak{T}' = \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{A}' = \mathfrak{A}$
and $\mathfrak{C}' = \mathfrak{C}$
shows $\eta : \mathfrak{T}' \mapsto_{CF.lKe.pw\alpha'} \mathfrak{F}' \circ_{CF} \mathfrak{R}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$
 ⟨*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 :
 $op-cf \mathfrak{T} \mapsto_{CF.lKe.pw\alpha} op-cf \mathfrak{G} \circ_{CF} op-cf \mathfrak{R} :$
 $op-cat \mathfrak{B} \mapsto_C op-cat \mathfrak{C} \mapsto_C op-cat \mathfrak{A}$
 ⟨*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 $\mathfrak{R}' = op-cf \mathfrak{R}$
and $\mathfrak{B}' = op-cat \mathfrak{B}$
and $\mathfrak{A}' = op-cat \mathfrak{A}$
and $\mathfrak{C}' = op-cat \mathfrak{C}$
shows *op-ntcf* $\varepsilon : \mathfrak{T}' \mapsto_{CF.lKe.pw\alpha} \mathfrak{G}' \circ_{CF} \mathfrak{R}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$

$\langle proof \rangle$

lemmas $[cat-op-intros] = is-cat-pw-rKe.is-cat-pw-lKe-op'$

lemma (in $is-cat-pw-lKe$) $is-cat-pw-rKe-op$:

$op-ntcf \eta$:

$op-cf \mathfrak{F} \circ_{CF} op-cf \mathfrak{K} \mapsto_{CF.rKe.pw\alpha} op-cf \mathfrak{T} :$

$op-cat \mathfrak{B} \mapsto_C op-cat \mathfrak{C} \mapsto_C op-cat \mathfrak{A}$

$\langle proof \rangle$

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 $op-ntcf \eta : \mathfrak{F}' \circ_{CF} \mathfrak{K}' \mapsto_{CF.rKe.pw\alpha} \mathfrak{T}' : \mathfrak{B}' \mapsto_C \mathfrak{C}' \mapsto_C \mathfrak{A}'$

$\langle proof \rangle$

lemmas $[cat-op-intros] = is-cat-pw-lKe.is-cat-pw-lKe-op'$

15.2 Lemma X.5: L-10-5-N

This subsection and several further subsections (15.2-15.8) expose definitions that are used in the proof of the technical lemma that was used in the proof of Theorem 3 from Chapter X-5 in [9].

definition $L-10-5-N :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c =$

[
 (
 $\lambda a \in_o \mathfrak{T}(\text{HomCod})(\text{Obj}).$
 $cf-nt \alpha \beta \mathfrak{K}(\text{ObjMap})(cf-map (Hom_{O.C\alpha} \mathfrak{T}(\text{HomCod})(a,-) \circ_{CF} \mathfrak{T}), c).$
),
 (
 $\lambda f \in_o \mathfrak{T}(\text{HomCod})(\text{Arr}).$
 $cf-nt \alpha \beta \mathfrak{K}(\text{ArrMap})($
 $ntcf-arrow (Hom_{A.C\alpha} \mathfrak{T}(\text{HomCod})(f,-) \circ_{NTCF-CF} \mathfrak{T}), \mathfrak{K}(\text{HomCod})(\text{CIId})(c)$
 $).$
),
 $op-cat (\mathfrak{T}(\text{HomCod})),$
 $cat-Set \beta$
] \circ

Components.

lemma $L-10-5-N-components$:

shows $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap}) =$

(
 $\lambda a \in_o \mathfrak{T}(\text{HomCod})(\text{Obj}).$
 $cf-nt \alpha \beta \mathfrak{K}(\text{ObjMap})(cf-map (Hom_{O.C\alpha} \mathfrak{T}(\text{HomCod})(a,-) \circ_{CF} \mathfrak{T}), c).$
)

and $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ArrMap}) =$

(
 $\lambda f \in_o \mathfrak{T}(\text{HomCod})(\text{Arr}).$
 $cf-nt \alpha \beta \mathfrak{K}(\text{ArrMap})($
 $ntcf-arrow (Hom_{A.C\alpha} \mathfrak{T}(\text{HomCod})(f,-) \circ_{NTCF-CF} \mathfrak{T}), \mathfrak{K}(\text{HomCod})(\text{CIId})(c)$
 $).$
)

)
and $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{HomDom}) = \text{op-cat } (\mathfrak{T}(\text{HomCod}))$
and $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{HomCod}) = \text{cat-Set } \beta$
<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 $L-10-5-N\text{-components}' = L-10-5-N\text{-components}$
where $\mathfrak{T}=\mathfrak{T}$ **and** $\mathfrak{K}=\mathfrak{K}$, *unfolded cat-cs-simps*
]

lemmas $[\text{cat-Kan-cs-simps}] = L-10-5-N\text{-components}'(3,4)$

end

15.2.1 Object map

mk-VLambda $L-10-5-N\text{-components}(1)$
 $[\text{vsu } L-10-5-N\text{-ObjMap-vsuv}[\text{cat-Kan-cs-intros}]]$

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} c$
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 $L-10-5-N\text{-components}'(1)[OF \mathfrak{K} \mathfrak{T}]$
 $[\text{vdomain } L-10-5-N\text{-ObjMap-vdomain}[\text{cat-Kan-cs-simps}]]$
 $[\text{app } L-10-5-N\text{-ObjMap-app}[\text{cat-Kan-cs-simps}]]$

end

15.2.2 Arrow map

mk-VLambda $L-10-5-N\text{-components}(2)$
 $[\text{vsu } L-10-5-N\text{-ArrMap-vsuv}[\text{cat-Kan-cs-intros}]]$

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T} c$
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 $L-10-5-N\text{-components}'(2)[OF \mathfrak{K} \mathfrak{T}]$
 $[\text{vdomain } L-10-5-N\text{-ArrMap-vdomain}[\text{cat-Kan-cs-simps}]]$
 $[\text{app } L-10-5-N\text{-ArrMap-app}[\text{cat-Kan-cs-simps}]]$

end

15.2.3 $L-10-5-N$ is a functor

lemma $L-10-5-N\text{-is-functor}$:

assumes $Z \beta$
and $\alpha \in_o \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_o \mathfrak{C}(\text{Obj})$
shows $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c : \text{op-cat } \mathfrak{A} \mapsto \mapsto_{C\beta} \text{cat-Set } \beta$
 ⟨proof⟩

lemma $L-10-5-N\text{-is-functor}'[\text{cat-Kan-cs-intros}]$:

assumes $Z \beta$
and $\alpha \in_o \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_o \mathfrak{C}(\text{Obj})$
and $\mathfrak{A}' = \text{op-cat } \mathfrak{A}$
and $\mathfrak{B}' = \text{cat-Set } \beta$
and $\beta' = \beta$
shows $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c : \mathfrak{A}' \mapsto \mapsto_{C\beta'} \mathfrak{B}'$
 ⟨proof⟩

15.3 Lemma X.5: $L-10-5-v\text{-arrow}$

15.3.1 Definition and elementary properties

definition $L-10-5-v\text{-arrow} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b =$

$[$
 $(\lambda f \in_o \text{Hom } (\mathfrak{K}(\text{HomCod})) c (\mathfrak{K}(\text{ObjMap})(b)). \tau(\text{NTMap})(\emptyset, b, f)\bullet),$
 $\text{Hom } (\mathfrak{K}(\text{HomCod})) c (\mathfrak{K}(\text{ObjMap})(b)),$
 $\text{Hom } (\mathfrak{T}(\text{HomCod})) a (\mathfrak{T}(\text{ObjMap})(b))$
 $]$

Components.

lemma $L-10-5-v\text{-arrow-components}$:

shows $L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b(\text{ArrVal}) =$
 $(\lambda f \in_o \text{Hom } (\mathfrak{K}(\text{HomCod})) c (\mathfrak{K}(\text{ObjMap})(b)). \tau(\text{NTMap})(\emptyset, b, f)\bullet)$
and $L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b(\text{ArrDom}) = \text{Hom } (\mathfrak{K}(\text{HomCod})) c (\mathfrak{K}(\text{ObjMap})(b))$
and $L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b(\text{ArrCod}) = \text{Hom } (\mathfrak{T}(\text{HomCod})) a (\mathfrak{T}(\text{ObjMap})(b))$
 ⟨proof⟩

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T}$
assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

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 $L-10-5-v\text{-arrow-components}' = L-10-5-v\text{-arrow-components}[$

where $\mathfrak{T}=\mathfrak{T}$ **and** $\mathfrak{K}=\mathfrak{K}$, *unfolded cat-cs-simps*

$]$

lemmas $[\text{cat-Kan-cs-simps}] = L-10-5-v\text{-arrow-components}'(2,3)$

end

15.3.2 Arrow value

mk-VLambda *L-10-5-v-arrow-components(1)*
 $[vsv\ L-10-5-v-arrow-ArrVal-vsuv[cat-Kan-cs-intros]]$

context

fixes $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{A}\ \mathfrak{K}\ \mathfrak{T}$
assumes $\mathfrak{K}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

begin

mk-VLambda *L-10-5-v-arrow-components'(1)[OF K T]*
 $[vdomain\ L-10-5-v-arrow-ArrVal-vdomain[cat-Kan-cs-simps]]$
 $[app\ L-10-5-v-arrow-ArrVal-app[unfolded\ in-Hom-iff]]$

end

lemma *L-10-5-v-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 $f: c \mapsto_{\mathfrak{C}} \mathfrak{K}(ObjMap)(b)$
shows $L-10-5-v-arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau\ a\ b(ArrVal)(f) = \tau(NTMap)(\theta, b, f)$.
 $\langle proof \rangle$

15.3.3 *L-10-5-v-arrow* is an arrow

lemma *L-10-5-v-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 $\tau' = ntcf-arrow\ \tau$
and $\tau: a <_{CF.cone} \mathfrak{T} \circ_{CF}\ c\ o\ \sqcap_{CF}\ \mathfrak{K}: c \downarrow_{CF}\ \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $f: c \mapsto_{\mathfrak{C}} \mathfrak{K}(ObjMap)(b)$
and $b \in_{\circ} \mathfrak{B}(Obj)$
shows $L-10-5-v-arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b(ArrVal)(f) : a \mapsto_{\mathfrak{A}} \mathfrak{T}(ObjMap)(b)$
 $\langle proof \rangle$

lemma *L-10-5-v-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 $\tau' = ntcf-arrow\ \tau$
and $a' = a$
and $b' = \mathfrak{T}(ObjMap)(b)$
and $\mathfrak{A}' = \mathfrak{A}$
and $\tau: a <_{CF.cone} \mathfrak{T} \circ_{CF}\ c\ o\ \sqcap_{CF}\ \mathfrak{K}: c \downarrow_{CF}\ \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $f: c \mapsto_{\mathfrak{C}} \mathfrak{K}(ObjMap)(b)$
and $b \in_{\circ} \mathfrak{B}(Obj)$
shows $L-10-5-v-arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b(ArrVal)(f) : a' \mapsto_{\mathfrak{A}'} b'$
 $\langle proof \rangle$

15.3.4 Further properties

lemma *L-10-5-v-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 $c \in_{\circ} \mathfrak{C}(Obj)$
and $\tau' = ntcf-arrow\ \tau$
and $\tau: a <_{CF.cone} \mathfrak{T} \circ_{CF}\ c\ o\ \sqcap_{CF}\ \mathfrak{K}: c \downarrow_{CF}\ \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $b \in_{\circ} \mathfrak{B}(Obj)$
shows $L-10-5-v-arrow\ \mathfrak{T}\ \mathfrak{K}\ c\ \tau'\ a\ b$:

$Hom \mathfrak{C} c (\mathfrak{K}(\mathfrak{ObjMap})(b)) \mapsto_{cat-Set \alpha} Hom \mathfrak{A} a (\mathfrak{T}(\mathfrak{ObjMap})(b))$
 ⟨proof⟩

lemma *L-10-5-v-arrow-is-arr'*[*cat-Kan-cs-intros*]:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_o \mathfrak{C}(\mathfrak{Obj})$
and $\tau' = ntcf\text{-arrow } \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto_{C\alpha} \mathfrak{A}$
and $b \in_o \mathfrak{B}(\mathfrak{Obj})$
and $A = Hom \mathfrak{C} c (\mathfrak{K}(\mathfrak{ObjMap})(b))$
and $B = Hom \mathfrak{A} a (\mathfrak{T}(\mathfrak{ObjMap})(b))$
and $\mathfrak{C}' = cat\text{-Set } \alpha$

shows *L-10-5-v-arrow* $\mathfrak{T} \mathfrak{K} c \tau' a b : A \mapsto_{\mathfrak{C}'} B$

⟨proof⟩

lemma *L-10-5-v-cf-hom*[*cat-Kan-cs-simps*]:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_o \mathfrak{C}(\mathfrak{Obj})$
and $\tau' = ntcf\text{-arrow } \tau$
and $\tau : a <_{CF.cone} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto_{C\alpha} \mathfrak{A}$
and $a \in_o \mathfrak{A}(\mathfrak{Obj})$
and $f : a' \mapsto_{\mathfrak{B}} b'$

shows

L-10-5-v-arrow $\mathfrak{T} \mathfrak{K} c \tau' a b' \circ_{A cat\text{-Set } \alpha}$
cf-hom $\mathfrak{C} [\mathfrak{C}(\mathfrak{CId})(c), \mathfrak{K}(\mathfrak{ArrMap})(f)] \circ =$
cf-hom $\mathfrak{A} [\mathfrak{A}(\mathfrak{CId})(a), \mathfrak{T}(\mathfrak{ArrMap})(f)] \circ_{A cat\text{-Set } \alpha}$
L-10-5-v-arrow $\mathfrak{T} \mathfrak{K} c \tau' a a'$
 (is ?lhs = ?rhs)

⟨proof⟩

15.4 Lemma X.5: *L-10-5-τ*

15.4.1 Definition and elementary properties

definition *L-10-5-τ* where *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a =$

[
 ($\lambda bf \in_o c \downarrow_{CF} \mathfrak{K}(\mathfrak{Obj}). v(\mathfrak{NTMap})(bf(\mathfrak{1}_N))(\mathfrak{ArrVal})(bf(\mathfrak{2}_N))$),
cf-const ($c \downarrow_{CF} \mathfrak{K}$) ($\mathfrak{T}(\mathfrak{HomCod})$) a ,
 $\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K}$,
 $c \downarrow_{CF} \mathfrak{K}$,
 ($\mathfrak{T}(\mathfrak{HomCod})$)
] \circ .

Components.

lemma *L-10-5-τ-components*:

shows *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a(\mathfrak{NTMap}) =$
 ($\lambda bf \in_o c \downarrow_{CF} \mathfrak{K}(\mathfrak{Obj}). v(\mathfrak{NTMap})(bf(\mathfrak{1}_N))(\mathfrak{ArrVal})(bf(\mathfrak{2}_N))$)
and *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a(\mathfrak{NTDom}) = cf\text{-const} (c \downarrow_{CF} \mathfrak{K}) (\mathfrak{T}(\mathfrak{HomCod})) a$
and *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a(\mathfrak{NTCod}) = \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K}$
and *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a(\mathfrak{NTDGDom}) = c \downarrow_{CF} \mathfrak{K}$
and *L-10-5-τ* $\mathfrak{T} \mathfrak{K} c v a(\mathfrak{NTDGCod}) = (\mathfrak{T}(\mathfrak{HomCod}))$

⟨proof⟩

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T}$

assumes $\mathfrak{K} : \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$

and $\mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
begin

interpretation $\mathfrak{R}: \text{is-functor } \alpha \mathfrak{B} \mathfrak{C} \mathfrak{R} \langle \text{proof} \rangle$
interpretation $\mathfrak{T}: \text{is-functor } \alpha \mathfrak{B} \mathfrak{A} \mathfrak{T} \langle \text{proof} \rangle$

lemmas $L-10-5-\tau\text{-components}' = L-10-5-\tau\text{-components}[$
 where $\mathfrak{T}=\mathfrak{T}$ and $\mathfrak{R}=\mathfrak{R}$, *unfolded cat-cs-simps*
 $]$

lemmas $[cat\text{-Kan-cs-simps}] = L-10-5-\tau\text{-components}'(2-5)$

end

15.4.2 Natural transformation map

mk-VLambda $L-10-5-\tau\text{-components}(1)$
 $[vsv\ L-10-5-\tau\text{-NTMap-vs}[cat\text{-Kan-cs-intros}]]$
 $[vdomain\ L-10-5-\tau\text{-NTMap-vdomain}[cat\text{-Kan-cs-simps}]]$

lemma $L-10-5-\tau\text{-NTMap-app}[cat\text{-Kan-cs-simps}]$:
 assumes $bf = [0, b, f]_{\circ}$ and $bf \in_{\circ} c \downarrow_{CF} \mathfrak{R}(\text{Obj})$
 shows $L-10-5-\tau\ \mathfrak{T}\ \mathfrak{R}\ c\ v\ a(\text{NTMap})(\text{Obj}) = v(\text{NTMap})(\text{Obj})(\text{ArrVal})(f)$
 $\langle \text{proof} \rangle$

15.4.3 $L-10-5-\tau$ is a cone

lemma $L-10-5-\tau\text{-is-cat-cone}[cat\text{-cs-intros}]$:
 assumes $\mathfrak{R}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
 and $v'\text{-def}: v' = \text{ntcf-arrow } v$
 and $v: v$:
 $Hom_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{R} \mapsto_{CF} Hom_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$
 and $a: a \in_{\circ} \mathfrak{A}(\text{Obj})$
 shows $L-10-5-\tau\ \mathfrak{T}\ \mathfrak{R}\ c\ v'\ a: a <_{CF.cone} \mathfrak{T} \circ_{CF} c \circ \prod_{CF} \mathfrak{R}: c \downarrow_{CF} \mathfrak{R} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 $\langle \text{proof} \rangle$

lemma $L-10-5-\tau\text{-is-cat-cone}'[cat\text{-cs-intros}]$:
 assumes $\mathfrak{R}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
 and $v' = \text{ntcf-arrow } v$
 and $\mathfrak{F}' = \mathfrak{T} \circ_{CF} c \circ \prod_{CF} \mathfrak{R}$
 and $c\mathfrak{R} = c \downarrow_{CF} \mathfrak{R}$
 and $\mathfrak{A}' = \mathfrak{A}$
 and $\alpha' = \alpha$
 and $v: v$:
 $Hom_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{R} \mapsto_{CF} Hom_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T}: \mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$
 and $a \in_{\circ} \mathfrak{A}(\text{Obj})$
 shows $L-10-5-\tau\ \mathfrak{T}\ \mathfrak{R}\ c\ v'\ a: a <_{CF.cone} \mathfrak{F}': c\mathfrak{R} \mapsto \mapsto_{C\alpha'} \mathfrak{A}'$
 $\langle \text{proof} \rangle$

15.5 Lemma X.5: $L-10-5-v$

15.5.1 Definition and elementary properties

definition $L-10-5-v :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a =$
 $[$
 $(\lambda b \in_o \mathfrak{T}(\mathit{HomDom})(\mathit{Obj}). L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b),$
 $\mathit{Hom}_{O.C\alpha} \mathfrak{K}(\mathit{HomCod})(c, -) \circ_{CF} \mathfrak{K},$
 $\mathit{Hom}_{O.C\alpha} \mathfrak{T}(\mathit{HomCod})(a, -) \circ_{CF} \mathfrak{T},$
 $\mathfrak{T}(\mathit{HomDom}),$
 $\mathit{cat-Set } \alpha$
 $]$

Components.

lemma $L-10-5-v\text{-components}$:

shows $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a(\mathit{NTMap}) =$
 $(\lambda b \in_o \mathfrak{T}(\mathit{HomDom})(\mathit{Obj}). L-10-5-v\text{-arrow } \mathfrak{T} \mathfrak{K} c \tau a b)$
and $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a(\mathit{NTDom}) = \mathit{Hom}_{O.C\alpha} \mathfrak{K}(\mathit{HomCod})(c, -) \circ_{CF} \mathfrak{K}$
and $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a(\mathit{NTCod}) = \mathit{Hom}_{O.C\alpha} \mathfrak{T}(\mathit{HomCod})(a, -) \circ_{CF} \mathfrak{T}$
and $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a(\mathit{NTDGDom}) = \mathfrak{T}(\mathit{HomDom})$
and $L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a(\mathit{NTDGCod}) = \mathit{cat-Set } \alpha$
 $\langle \mathit{proof} \rangle$

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T}$
assumes $\mathfrak{K}: \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation $\mathfrak{K}: \mathit{is-functor } \alpha \mathfrak{B} \mathfrak{C} \mathfrak{K} \langle \mathit{proof} \rangle$

interpretation $\mathfrak{T}: \mathit{is-functor } \alpha \mathfrak{B} \mathfrak{A} \mathfrak{T} \langle \mathit{proof} \rangle$

lemmas $L-10-5-v\text{-components}' = L-10-5-v\text{-components}[$
where $\mathfrak{T}=\mathfrak{T}$ **and** $\mathfrak{K}=\mathfrak{K}$, *unfolded cat-cs-simps*
 $]$

lemmas $[\mathit{cat-Kan-cs-simps}] = L-10-5-v\text{-components}'(2-5)$

end

15.5.2 Natural transformation map

mk-VLambda $L-10-5-v\text{-components}(1)$
 $[\mathit{vsv } L-10-5-v\text{-NTMap}\text{-vsv}[\mathit{cat-Kan-cs-intros}]]$

context

fixes $\alpha \mathfrak{B} \mathfrak{C} \mathfrak{A} \mathfrak{K} \mathfrak{T}$
assumes $\mathfrak{K}: \mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation $\mathfrak{K}: \mathit{is-functor } \alpha \mathfrak{B} \mathfrak{C} \mathfrak{K} \langle \mathit{proof} \rangle$

interpretation $\mathfrak{T}: \mathit{is-functor } \alpha \mathfrak{B} \mathfrak{A} \mathfrak{T} \langle \mathit{proof} \rangle$

mk-VLambda $L-10-5-v\text{-components}'(1)[\mathit{OF } \mathfrak{K} \mathfrak{T}]$
 $[\mathit{vdomain } L-10-5-v\text{-NTMap}\text{-vdomain}[\mathit{cat-Kan-cs-simps}]]$
 $[\mathit{app } L-10-5-v\text{-NTMap}\text{-app}[\mathit{cat-Kan-cs-simps}]]$

end

15.5.3 $L-10-5-v$ is a natural transformation

lemma $L-10-5-v\text{-is-ntcf}$:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
and $\tau' \text{-def} : \tau' = \text{ntcf-arrow } \tau$
and $\tau : \tau : a \langle_{CF} \text{cone } \mathfrak{T} \circ_{CF} c \text{ o} \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $a : a \in_{\circ} \mathfrak{A}(\text{Obj})$
shows $L\text{-}10\text{-}5\text{-}v \alpha \mathfrak{T} \mathfrak{K} c \tau' a :$
 $Hom_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{K} \mapsto_{CF} Hom_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$
 $(\text{is } \langle ?L\text{-}10\text{-}5\text{-}v : ?H\text{-}\mathfrak{C} c \circ_{CF} \mathfrak{K} \mapsto_{CF} ?H\text{-}\mathfrak{A} a \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha \rangle)$
 $\langle \text{proof} \rangle$

lemma $L\text{-}10\text{-}5\text{-}v\text{-is-ntcf}'[\text{cat-Kan-cs-intros}] :$

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
and $\tau' = \text{ntcf-arrow } \tau$
and $\mathfrak{F}' = Hom_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{K}$
and $\mathfrak{G}' = Hom_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T}$
and $\mathfrak{B}' = \mathfrak{B}$
and $\mathfrak{C}' = \text{cat-Set } \alpha$
and $\alpha' = \alpha$
and $\tau : a \langle_{CF} \text{cone } \mathfrak{T} \circ_{CF} c \text{ o} \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $a \in_{\circ} \mathfrak{A}(\text{Obj})$
shows $L\text{-}10\text{-}5\text{-}v \alpha \mathfrak{T} \mathfrak{K} c \tau' a : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B}' \mapsto \mapsto_{C\alpha'} \mathfrak{C}'$
 $\langle \text{proof} \rangle$

15.6 Lemma X.5: $L\text{-}10\text{-}5\text{-}\chi\text{-arrow}$

15.6.1 Definition and elementary properties

definition $L\text{-}10\text{-}5\text{-}\chi\text{-arrow}$

where $L\text{-}10\text{-}5\text{-}\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a =$
 $[$
 $(\lambda v \in_{\circ} L\text{-}10\text{-}5\text{-}N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a). \text{ntcf-arrow } (L\text{-}10\text{-}5\text{-}\tau \mathfrak{T} \mathfrak{K} c v a)),$
 $L\text{-}10\text{-}5\text{-}N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a),$
 $\text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \text{ o} \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(\downarrow a)$
 $].$

Components.

lemma $L\text{-}10\text{-}5\text{-}\chi\text{-arrow-components} :$

shows $L\text{-}10\text{-}5\text{-}\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrVal}) =$
 $(\lambda v \in_{\circ} L\text{-}10\text{-}5\text{-}N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a). \text{ntcf-arrow } (L\text{-}10\text{-}5\text{-}\tau \mathfrak{T} \mathfrak{K} c v a))$
and $L\text{-}10\text{-}5\text{-}\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrDom}) = L\text{-}10\text{-}5\text{-}N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a)$
and $L\text{-}10\text{-}5\text{-}\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrCod}) =$
 $\text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \text{ o} \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(\downarrow a)$
 $\langle \text{proof} \rangle$

lemmas $[\text{cat-Kan-cs-simps}] = L\text{-}10\text{-}5\text{-}\chi\text{-arrow-components}(2,3)$

15.6.2 Arrow value

mk-VLambda $L\text{-}10\text{-}5\text{-}\chi\text{-arrow-components}(1)$

$[\text{vsu } L\text{-}10\text{-}5\text{-}\chi\text{-arrow-vsuv}[\text{cat-Kan-cs-intros}]$
 $[\text{vdomain } L\text{-}10\text{-}5\text{-}\chi\text{-arrow-vdomain}]$
 $[\text{app } L\text{-}10\text{-}5\text{-}\chi\text{-arrow-app}]$

lemma $L\text{-}10\text{-}5\text{-}\chi\text{-arrow-vdomain}'[\text{cat-Kan-cs-simps}] :$

assumes $\mathcal{Z} \beta$
and $\alpha \in_{\circ} \beta$

and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
and $a \in_{\circ} \mathfrak{A}(\text{Obj})$
shows $\mathcal{D}_{\circ} (L-10-5-\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrVal})) = \text{Hom}$
 $(\text{cat-FUNCT } \alpha \mathfrak{B} (\text{cat-Set } \alpha))$
 $(\text{cf-map } (\text{Hom}_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{K}))$
 $(\text{cf-map } (\text{Hom}_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T}))$
 $\langle \text{proof} \rangle$

lemma $L-10-5-\chi\text{-arrow-app}'[\text{cat-Kan-cs-simps}]$:

assumes $Z \beta$
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 $c \in_{\circ} \mathfrak{C}(\text{Obj})$
and $v'\text{-def}: v' = \text{ntcf-arrow } v$
and $v : v :$
 $\text{Hom}_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{K} \mapsto_{CF} \text{Hom}_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$
and $a : a \in_{\circ} \mathfrak{A}(\text{Obj})$
shows
 $L-10-5-\chi\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrVal})(v') =$
 $\text{ntcf-arrow } (L-10-5-\tau \mathfrak{T} \mathfrak{K} c v' a)$
 $\langle \text{proof} \rangle$

lemma $v\tau a\text{-def}$:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\text{Obj})$
and $v\tau a'\text{-def}: v\tau a' = \text{ntcf-arrow } v\tau a$
and $v\tau a : v\tau a :$
 $\text{Hom}_{O.C\alpha} \mathfrak{C}(c, -) \circ_{CF} \mathfrak{K} \mapsto_{CF} \text{Hom}_{O.C\alpha} \mathfrak{A}(a, -) \circ_{CF} \mathfrak{T} :$
 $\mathfrak{B} \mapsto \mapsto_{C\alpha} \text{cat-Set } \alpha$
and $a : a \in_{\circ} \mathfrak{A}(\text{Obj})$
shows $v\tau a = L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c (\text{ntcf-arrow } (L-10-5-\tau \mathfrak{T} \mathfrak{K} c v\tau a' a)) a$
(is $\langle v\tau a = ?L-10-5-v (\text{ntcf-arrow } ?L-10-5-\tau) a \rangle$
 $\langle \text{proof} \rangle$

15.7 Lemma X.5: $L-10-5-\chi'\text{-arrow}$

15.7.1 Definition and elementary properties

definition $L-10-5-\chi'\text{-arrow} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where $L-10-5-\chi'\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a =$

$[$
 $($
 $\lambda \tau \in_{\circ} \text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(|a|).$
 $\text{ntcf-arrow } (L-10-5-v \alpha \mathfrak{T} \mathfrak{K} c \tau a)$
 $),$
 $\text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(|a|),$
 $L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(|a|)$
 $].$

Components.

lemma $L-10-5-\chi'\text{-arrow-components}$:

shows $L-10-5-\chi'\text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a(\text{ArrVal}) =$
 $($
 $\lambda \tau \in_{\circ} \text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(|a|).$

$ntcf\text{-arrow } (L\text{-}10\text{-}5\text{-}v \ \alpha \ \mathfrak{T} \ \mathfrak{K} \ c \ \tau \ a)$
)
and $[cat\text{-}Kan\text{-}cs\text{-}simps]: L\text{-}10\text{-}5\text{-}\chi'\text{-arrow } \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ a \ (ArrDom) =$
 $cf\text{-Cone } \alpha \ \beta \ (\mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K}) \ (ObjMap) \ (a)$
and $[cat\text{-}Kan\text{-}cs\text{-}simps]: L\text{-}10\text{-}5\text{-}\chi'\text{-arrow } \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ a \ (ArrCod) =$
 $L\text{-}10\text{-}5\text{-}N \ \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ (ObjMap) \ (a)$
 $\langle proof \rangle$

15.7.2 Arrow value

mk-VLambda $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-components}(1)$
 $|vsv \ L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-ArrVal-vsuv}[cat\text{-}Kan\text{-}cs\text{-}intros]$
 $|vdomain \ L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-ArrVal-vdomain}$
 $|app \ L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-ArrVal-app}$

lemma $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-ArrVal-vdomain}'[cat\text{-}Kan\text{-}cs\text{-}simps]:$
assumes $Z \ \beta$
and $\alpha \in_o \ \beta$
and $\tau : \tau : a <_{CF.cone} \ \mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a : a \in_o \ \mathfrak{A} \ (Obj)$
shows $\mathcal{D}_o \ (L\text{-}10\text{-}5\text{-}\chi'\text{-arrow } \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ a \ (ArrVal)) = Hom$
 $(cat\text{-}FUNCT \ \alpha \ (c \downarrow_{CF} \mathfrak{K}) \ \mathfrak{A})$
 $(cf\text{-map} \ (cf\text{-const} \ (c \downarrow_{CF} \mathfrak{K}) \ \mathfrak{A} \ a))$
 $(cf\text{-map} \ (\mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K}))$
 $\langle proof \rangle$

lemma $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-ArrVal-app}'[cat\text{-}cs\text{-}simps]:$
assumes $Z \ \beta$
and $\alpha \in_o \ \beta$
and $\tau'\text{-def}: \tau' = ntcf\text{-arrow } \tau$
and $\tau : \tau : a <_{CF.cone} \ \mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $a : a \in_o \ \mathfrak{A} \ (Obj)$
shows $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow } \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ a \ (ArrVal) \ (\tau') =$
 $ntcf\text{-arrow } (L\text{-}10\text{-}5\text{-}v \ \alpha \ \mathfrak{T} \ \mathfrak{K} \ c \ \tau' \ a)$
 $\langle proof \rangle$

15.7.3 $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow}$ is an isomorphism in the category Set

lemma $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow-is-iso-arr}:$
assumes $Z \ \beta$
and $\alpha \in_o \ \beta$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $c \in_o \ \mathfrak{C} \ (Obj)$
and $a \in_o \ \mathfrak{A} \ (Obj)$
shows $L\text{-}10\text{-}5\text{-}\chi'\text{-arrow } \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ a :$
 $cf\text{-Cone } \alpha \ \beta \ (\mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K}) \ (ObjMap) \ (a) \mapsto_{isocat\text{-}Set} \ \beta$
 $L\text{-}10\text{-}5\text{-}N \ \alpha \ \beta \ \mathfrak{T} \ \mathfrak{K} \ c \ (ObjMap) \ (a)$
(
is
 \langle
 $?L\text{-}10\text{-}5\text{-}\chi'\text{-arrow} :$
 $cf\text{-Cone } \alpha \ \beta \ (\mathfrak{T} \circ_{CF} c \ o\sqcap_{CF} \mathfrak{K}) \ (ObjMap) \ (a) \mapsto_{isocat\text{-}Set} \ \beta$
 $?L\text{-}10\text{-}5\text{-}N \ (ObjMap) \ (a)$
 \rangle
)
 $\langle proof \rangle$

lemma *L-10-5- χ' -arrow-is-iso-arr'*[*cat-Kan-cs-intros*]:

assumes $Z \beta$
 and $\alpha \in_o \beta$
 and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $c \in_o \mathfrak{C}(\text{Obj})$
 and $a \in_o \mathfrak{A}(\text{Obj})$
 and $A = \text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(\downarrow a)$
 and $B = \text{L-10-5-N } \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a)$
 and $\mathfrak{C}' = \text{cat-Set } \beta$
 shows *L-10-5- χ' -arrow* $\alpha \beta \mathfrak{T} \mathfrak{K} c a : A \mapsto_{\text{iso}} \mathfrak{C}' B$
 ⟨*proof*⟩

lemma *L-10-5- χ' -arrow-is-arr*:

assumes $Z \beta$
 and $\alpha \in_o \beta$
 and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $c \in_o \mathfrak{C}(\text{Obj})$
 and $a \in_o \mathfrak{A}(\text{Obj})$
 shows *L-10-5- χ' -arrow* $\alpha \beta \mathfrak{T} \mathfrak{K} c a :$
 $\text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(\downarrow a) \mapsto_{\text{cat-Set}} \beta$
 $\text{L-10-5-N } \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a)$
 ⟨*proof*⟩

lemma *L-10-5- χ' -arrow-is-arr'*[*cat-Kan-cs-intros*]:

assumes $Z \beta$
 and $\alpha \in_o \beta$
 and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
 and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
 and $c \in_o \mathfrak{C}(\text{Obj})$
 and $a \in_o \mathfrak{A}(\text{Obj})$
 and $A = \text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})(\text{ObjMap})(\downarrow a)$
 and $B = \text{L-10-5-N } \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{ObjMap})(\downarrow a)$
 and $\mathfrak{C}' = \text{cat-Set } \beta$
 shows *L-10-5- χ' -arrow* $\alpha \beta \mathfrak{T} \mathfrak{K} c a : A \mapsto_{\mathfrak{C}'} B$
 ⟨*proof*⟩

15.8 Lemma X.5: *L-10-5- χ*

15.8.1 Definition and elementary properties

definition *L-10-5- χ* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

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

[
 $(\lambda a \in_o \mathfrak{T}(\text{HomCod})(\downarrow \text{Obj}). \text{L-10-5-}\chi' \text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a),$
 $\text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K}),$
 $\text{L-10-5-N } \alpha \beta \mathfrak{T} \mathfrak{K} c,$
 $\text{op-cat } (\mathfrak{T}(\text{HomCod})),$
 $\text{cat-Set } \beta$
]_o.

Components.

lemma *L-10-5- χ -components*:

shows *L-10-5- χ* $\alpha \beta \mathfrak{T} \mathfrak{K} c(\text{NTMap}) =$
 $(\lambda a \in_o \mathfrak{T}(\text{HomCod})(\downarrow \text{Obj}). \text{L-10-5-}\chi' \text{-arrow } \alpha \beta \mathfrak{T} \mathfrak{K} c a)$
 and [*cat-Kan-cs-simps*]:
 $\text{L-10-5-}\chi \alpha \beta \mathfrak{T} \mathfrak{K} c(\text{NTDom}) = \text{cf-Cone } \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K})$

and [*cat-Kan-cs-simps*]:
 $L-10-5-\chi \alpha \beta \mathfrak{T} \mathfrak{K} c(\mathcal{N}TCod) = L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c$
and $L-10-5-\chi \alpha \beta \mathfrak{T} \mathfrak{K} c(\mathcal{N}TDGDom) = op-cat (\mathfrak{T}(\mathcal{H}omCod))$
and [*cat-Kan-cs-simps*]: $L-10-5-\chi \alpha \beta \mathfrak{T} \mathfrak{K} c(\mathcal{N}TDGCod) = cat-Set \beta$
 ⟨*proof*⟩

context

fixes $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{T}$
assumes $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation *is-functor* $\alpha \mathfrak{B} \mathfrak{A} \mathfrak{T}$ ⟨*proof*⟩

lemmas $L-10-5-\chi-components'$ =
 $L-10-5-\chi-components[\mathbf{where} \ \mathfrak{T}=\mathfrak{T}, \text{ unfolded } cat-cs-simps]$

lemmas [*cat-Kan-cs-simps*] = $L-10-5-\chi-components'(4)$

end

15.8.2 Natural transformation map

mk-VLambda $L-10-5-\chi-components(1)$
 $|vsu \ L-10-5-\chi-NTMap-vsuv[cat-Kan-cs-intros]|$

context

fixes $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{T}$
assumes $\mathfrak{T}: \mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation *is-functor* $\alpha \mathfrak{B} \mathfrak{A} \mathfrak{T}$ ⟨*proof*⟩

mk-VLambda $L-10-5-\chi-components(1)[\mathbf{where} \ \mathfrak{T}=\mathfrak{T}, \text{ unfolded } cat-cs-simps]$
 $|vdomain \ L-10-5-\chi-NTMap-vdomain[cat-Kan-cs-simps]|$
 $|app \ L-10-5-\chi-NTMap-app[cat-Kan-cs-simps]|$

end

15.8.3 $L-10-5-\chi$ is a natural isomorphism

lemma $L-10-5-\chi-is-iso-ntcf$:

— See lemma on page 245 in [9].

assumes $\mathcal{Z} \beta$

and $\alpha \in_o \beta$

and $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$

and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $c \in_o \mathfrak{C}(\mathcal{O}bj)$

shows $L-10-5-\chi \alpha \beta \mathfrak{T} \mathfrak{K} c$:

$cf-Cone \alpha \beta (\mathfrak{T} \circ_{CF} c \circ \square_{CF} \mathfrak{K}) \mapsto_{CF.iso} L-10-5-N \alpha \beta \mathfrak{T} \mathfrak{K} c$:

$op-cat \ \mathfrak{A} \mapsto \mapsto_{C\beta} \ cat-Set \ \beta$

(**is** $\langle ?L-10-5-\chi : ?cf-Cone \mapsto_{CF.iso} ?L-10-5-N : op-cat \ \mathfrak{A} \mapsto \mapsto_{C\beta} \ cat-Set \ \beta \rangle$)

⟨*proof*⟩

15.9 The existence of a canonical limit or a canonical colimit for the pointwise Kan extensions

lemma (in *is-cat-pw-rKe*) *cat-pw-rKe-ex-cat-limit*:

— Based on the elements of Chapter X-5 in [9].

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\mathit{Obj})$
obtains UA
where $UA : \mathfrak{G}(\mathit{ObjMap})(\downarrow c) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<proof>

lemma (in *is-cat-pw-lKe*) *cat-pw-lKe-ex-cat-colimit*:

— Based on the elements of Chapter X-5 in [9].

assumes $\mathfrak{K} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{T} : \mathfrak{B} \mapsto \mapsto_{C\alpha} \mathfrak{A}$
and $c \in_{\circ} \mathfrak{C}(\mathit{Obj})$
obtains UA
where $UA : \mathfrak{T} \circ_{CF} \mathfrak{K} \circ_{CF} \sqcap_{O} c >_{CF.colim} \mathfrak{F}(\mathit{ObjMap})(\downarrow c) : \mathfrak{K} \circ_{CF} \downarrow c \mapsto \mapsto_{C\alpha} \mathfrak{A}$
<proof>

15.10 The limit and the colimit for the pointwise Kan extensions

15.10.1 Definition and elementary properties

See Theorem 3 in Chapter X-5 in [9].

definition *the-pw-cat-rKe-limit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{G} c =$

$[$
 $\mathfrak{G}(\mathit{ObjMap})(\downarrow c),$
 $($
 $SOME UA.$
 $UA : \mathfrak{G}(\mathit{ObjMap})(\downarrow c) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{T}(\mathit{HomCod})$
 $)$
 $]$

definition *the-pw-cat-lKe-colimit* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *the-pw-cat-lKe-colimit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{F} c =$

$[$
 $\mathfrak{F}(\mathit{ObjMap})(\downarrow c),$
 $op-ntcf$
 $($
 $the-pw-cat-rKe-limit \alpha (op-cf \mathfrak{K}) (op-cf \mathfrak{T}) (op-cf \mathfrak{F}) c(\mathit{UArr}) \circ_{NTCF-CF}$
 $op-cf-obj-comma \mathfrak{K} c$
 $)$
 $]$

Components.

lemma *the-pw-cat-rKe-limit-components*:

shows *the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{G} c(\mathit{UObj}) = \mathfrak{G}(\mathit{ObjMap})(\downarrow c)$

and *the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{G} c(\mathit{UArr}) =$

$($
 $SOME UA.$
 $UA : \mathfrak{G}(\mathit{ObjMap})(\downarrow c) <_{CF.lim} \mathfrak{T} \circ_{CF} c \circ \sqcap_{CF} \mathfrak{K} : c \downarrow_{CF} \mathfrak{K} \mapsto \mapsto_{C\alpha} \mathfrak{T}(\mathit{HomCod})$
 $)$

<proof>

lemma *the-pw-cat-lKe-colimit-components*:

shows *the-pw-cat-lKe-colimit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{F} c(\mathit{UObj}) = \mathfrak{F}(\mathit{ObjMap})(\downarrow c)$

and *the-pw-cat-lKe-colimit* $\alpha \mathfrak{K} \mathfrak{T} \mathfrak{F} c(\mathit{UArr}) = op-ntcf$

$($
 $the-pw-cat-rKe-limit \alpha (op-cf \mathfrak{K}) (op-cf \mathfrak{T}) (op-cf \mathfrak{F}) c(\mathit{UArr}) \circ_{NTCF-CF}$
 $op-cf-obj-comma \mathfrak{K} c$
 $)$

)
 ⟨proof⟩

context *is-functor*
 begin

lemmas *the-pw-cat-rKe-limit-components'* =
the-pw-cat-rKe-limit-components[**where** $\mathfrak{I}=\mathfrak{F}$, *unfolded cat-cs-simps*]

end

15.10.2 The limit for the pointwise right Kan extension is a limit, the colimit for the pointwise left Kan extension is a colimit

lemma (in *is-cat-pw-rKe*) *cat-pw-rKe-the-pw-cat-rKe-limit-is-cat-limit*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ and $\mathfrak{I} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$ and $c \in_o \mathfrak{C}(\text{Obj})$

shows *the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{G} c(\text{UArr})$:

the-pw-cat-rKe-limit $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{G} c(\text{UObj}) <_{CF.lim} \mathfrak{I} \circ_{CF} c \circ_{CF} \mathfrak{K}$:

$c \downarrow_{CF} \mathfrak{K} \mapsto_{C\alpha} \mathfrak{A}$

⟨proof⟩

lemma (in *is-cat-pw-lKe*) *cat-pw-lKe-the-pw-cat-lKe-colimit-is-cat-colimit*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ and $\mathfrak{I} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$ and $c \in_o \mathfrak{C}(\text{Obj})$

shows *the-pw-cat-lKe-colimit* $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{F} c(\text{UArr})$:

$\mathfrak{I} \circ_{CF} \mathfrak{K} \circ_{CF} \mathfrak{F} c >_{CF.colim} \mathfrak{I} \circ_{CF} \mathfrak{K} \circ_{CF} \mathfrak{F} c(\text{UObj})$:

$\mathfrak{K} \circ_{CF} \mathfrak{F} c \mapsto_{C\alpha} \mathfrak{A}$

⟨proof⟩

lemma (in *is-cat-pw-rKe*) *cat-pw-rKe-the-ntcf-rKe-is-cat-rKe*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ and $\mathfrak{I} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

shows *the-ntcf-rKe* $\alpha \mathfrak{I} \mathfrak{K}$ (*the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{G}$) :

the-cf-rKe $\alpha \mathfrak{I} \mathfrak{K}$ (*the-pw-cat-rKe-limit* $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{G}$) $\circ_{CF} \mathfrak{K} \mapsto_{CF.rKe\alpha} \mathfrak{I}$:

$\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$

⟨proof⟩

lemma (in *is-cat-pw-lKe*) *cat-pw-lKe-the-ntcf-lKe-is-cat-lKe*:

assumes $\mathfrak{K} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{C}$ and $\mathfrak{I} : \mathfrak{B} \mapsto_{C\alpha} \mathfrak{A}$

shows *the-ntcf-lKe* $\alpha \mathfrak{I} \mathfrak{K}$ (*the-pw-cat-lKe-colimit* $\alpha \mathfrak{K} \mathfrak{I} \mathfrak{F}$) :

$\mathfrak{I} \mapsto_{CF.lKe\alpha} \mathfrak{K} \circ_{CF} \mathfrak{F} c >_{CF.colim} \mathfrak{I} \circ_{CF} \mathfrak{K} \circ_{CF} \mathfrak{F} c(\text{UObj})$ $\circ_{CF} \mathfrak{K}$:

$\mathfrak{B} \mapsto_C \mathfrak{C} \mapsto_C \mathfrak{A}$

⟨proof⟩

16 Pointwise Kan extensions: application example

16.1 Background

The application example presented in this section is based on Exercise 6.1.ii in [14]. The primary purpose of this section is the instantiation of the locales associated with the pointwise Kan extensions.

lemma *cat-ordinal-2-is-arrE*:

assumes $f : a \mapsto_{\text{cat-ordinal } (2_{\mathbf{N}})} b$
obtains $f = [0, 0]_{\circ}$ **and** $a = 0$ **and** $b = 0$
 $| f = [0, 1_{\mathbf{N}}]_{\circ}$ **and** $a = 0$ **and** $b = 1_{\mathbf{N}}$
 $| f = [1_{\mathbf{N}}, 1_{\mathbf{N}}]_{\circ}$ **and** $a = 1_{\mathbf{N}}$ **and** $b = 1_{\mathbf{N}}$
<proof>

lemma *cat-ordinal-3-is-arrE*:

assumes $f : a \mapsto_{\text{cat-ordinal } (3_{\mathbf{N}})} b$
obtains $f = [0, 0]_{\circ}$ **and** $a = 0$ **and** $b = 0$
 $| f = [0, 1_{\mathbf{N}}]_{\circ}$ **and** $a = 0$ **and** $b = 1_{\mathbf{N}}$
 $| f = [0, 2_{\mathbf{N}}]_{\circ}$ **and** $a = 0$ **and** $b = 2_{\mathbf{N}}$
 $| f = [1_{\mathbf{N}}, 1_{\mathbf{N}}]_{\circ}$ **and** $a = 1_{\mathbf{N}}$ **and** $b = 1_{\mathbf{N}}$
 $| f = [1_{\mathbf{N}}, 2_{\mathbf{N}}]_{\circ}$ **and** $a = 1_{\mathbf{N}}$ **and** $b = 2_{\mathbf{N}}$
 $| f = [2_{\mathbf{N}}, 2_{\mathbf{N}}]_{\circ}$ **and** $a = 2_{\mathbf{N}}$ **and** $b = 2_{\mathbf{N}}$
<proof>

lemma *0123*: $0 \in_{\circ} 2_{\mathbf{N}}$ $1_{\mathbf{N}} \in_{\circ} 2_{\mathbf{N}}$ $0 \in_{\circ} 3_{\mathbf{N}}$ $1_{\mathbf{N}} \in_{\circ} 3_{\mathbf{N}}$ $2_{\mathbf{N}} \in_{\circ} 3_{\mathbf{N}}$ *<proof>*

16.2 $\mathfrak{K}23$

16.2.1 Definition and elementary properties

definition $\mathfrak{K}23 :: V$

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

Components.

lemma *$\mathfrak{K}23$ -components*:

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

and $[cat\text{-}Kan\text{-}cs\text{-}simps]: \mathfrak{K}23(\mathit{HomDom}) = cat\text{-}ordinal (2_{\mathbb{N}})$
and $[cat\text{-}Kan\text{-}cs\text{-}simps]: \mathfrak{K}23(\mathit{HomCod}) = cat\text{-}ordinal (3_{\mathbb{N}})$
 $\langle proof \rangle$

16.2.2 Object map

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

lemma $\mathfrak{K}23\text{-}ObjMap\text{-}app\text{-}0[cat\text{-}Kan\text{-}cs\text{-}simps]:$
assumes $x = 0$
shows $\mathfrak{K}23(\mathit{ObjMap})(x) = 0$
 $\langle proof \rangle$

lemma $\mathfrak{K}23\text{-}ObjMap\text{-}app\text{-}1[cat\text{-}Kan\text{-}cs\text{-}simps]:$
assumes $x = 1_{\mathbb{N}}$
shows $\mathfrak{K}23(\mathit{ObjMap})(x) = 2_{\mathbb{N}}$
 $\langle proof \rangle$

16.2.3 Arrow map

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

lemma $\mathfrak{K}23\text{-}ArrMap\text{-}app\text{-}00[cat\text{-}Kan\text{-}cs\text{-}simps]:$
assumes $f = [0, 0]_{\circ}$
shows $\mathfrak{K}23(\mathit{ArrMap})(f) = [0, 0]_{\circ}$
 $\langle proof \rangle$

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

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

16.2.4 $\mathfrak{K}23$ is a tiny functor

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

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

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

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

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

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

16.3 $LK23$: the functor associated with the left Kan extension along $\mathcal{K}23$

16.3.1 Definition and elementary properties

definition $LK23 :: V \Rightarrow V$

where $LK23 \mathfrak{F} =$

[
 (
 $\lambda a \in_o \text{cat-ordinal } (3_{\mathbb{N}}) (\text{Obj})$.
 $\text{if } a = 0 \Rightarrow \mathfrak{F}(\text{ObjMap})(\text{Obj})$
 $\quad | a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(\text{Obj})$
 $\quad | a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbb{N}})$
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Obj})$
)
 (
 $\lambda f \in_o \text{cat-ordinal } (3_{\mathbb{N}}) (\text{Arr})$.
 $\text{if } f = [0, 0]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [0, 1_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [0, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, 1_{\mathbb{N}})$.
 $\quad | f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, 1_{\mathbb{N}})$.
 $\quad | f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Arr})$
)
 $\text{cat-ordinal } (3_{\mathbb{N}})$,
 $\mathfrak{F}(\text{HomCod})$
]_o

Components.

lemma $LK23$ -components:

shows $LK23 \mathfrak{F}(\text{ObjMap}) =$

(
 $\lambda a \in_o \text{cat-ordinal } (3_{\mathbb{N}}) (\text{Obj})$.
 $\text{if } a = 0 \Rightarrow \mathfrak{F}(\text{ObjMap})(\text{Obj})$
 $\quad | a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(\text{Obj})$
 $\quad | a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbb{N}})$
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Obj})$
)

and $LK23 \mathfrak{F}(\text{ArrMap}) =$

(
 $\lambda f \in_o \text{cat-ordinal } (3_{\mathbb{N}}) (\text{Arr})$.
 $\text{if } f = [0, 0]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [0, 1_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [0, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, 1_{\mathbb{N}})$.
 $\quad | f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, \text{Obj})$.
 $\quad | f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(\text{Obj}, 1_{\mathbb{N}})$.
 $\quad | f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_o \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Arr})$
)

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

context *is-functor*
begin

lemmas $LK23\text{-components}' = LK23\text{-components}$ [where $\mathfrak{F}=\mathfrak{F}$, *unfolded cat-cs-simps*]

lemmas $[cat\text{-Kan-cs-simps}] = LK23\text{-components}'(3,4)$

end

lemmas $[cat\text{-Kan-cs-simps}] = \text{is-functor.LK23-components}'(3,4)$

16.3.2 Object map

mk-VLambda $LK23\text{-components}(1)$
 |*vsv* $LK23\text{-ObjMap-vsv}[cat\text{-Kan-cs-intros}]$
 |*vdomain* $LK23\text{-ObjMap-vdomain}[cat\text{-Kan-cs-simps}]$
 |*app* $LK23\text{-ObjMap-app}$

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

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

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

16.3.3 Arrow map

mk-VLambda $LK23\text{-components}(2)$
 |*vsv* $LK23\text{-ArrMap-vsv}[cat\text{-Kan-cs-intros}]$
 |*vdomain* $LK23\text{-ArrMap-vdomain}[cat\text{-Kan-cs-simps}]$
 |*app* $LK23\text{-ArrMap-app}$

lemma $LK23\text{-ArrMap-app-00}[cat\text{-Kan-cs-simps}]$:
assumes $f = [0, 0]_{\circ}$
shows $LK23 \mathfrak{F}(\text{ArrMap})(|f) = \mathfrak{F}(\text{ArrMap})(|0, 0)_{\bullet}$
 ⟨proof⟩

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

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

lemma *LK23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ}$
shows $LK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(0, 0)$.
<proof>

lemma *LK23-ArrMap-app-12*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows $LK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbb{N}})$.
<proof>

lemma *LK23-ArrMap-app-22*[*cat-Kan-cs-simps*]:
assumes $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows $LK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
<proof>

16.3.4 *LK23* is a functor

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

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

16.3.5 The fundamental property of *LK23*

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

16.4 *RK23*: the functor associated with the right Kan extension along *℔23*

16.4.1 Definition and elementary properties

definition *RK23* :: $V \Rightarrow V$
where *RK23* $\mathfrak{F} =$

[
 (
 $\lambda a \in_{\circ} \text{cat-ordinal } (3_{\mathbb{N}})(\text{Obj})$.
 if $a = 0 \Rightarrow \mathfrak{F}(\text{ObjMap})(0)$
 | $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbb{N}})$
 | $a = 2_{\mathbb{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbb{N}})$
 | otherwise $\Rightarrow \mathfrak{F}(\text{HomCod})(\text{Obj})$
),
 (
 $\lambda f \in_{\circ} \text{cat-ordinal } (3_{\mathbb{N}})(\text{Arr})$.
 if $f = [0, 0]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 0)$.
 | $f = [0, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbb{N}})$.
 | $f = [0, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbb{N}})$.
 | $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
 | $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
 | $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
 | otherwise $\Rightarrow \mathfrak{F}(\text{HomCod})(\text{Arr})$
)
]

),
cat-ordinal ($3_{\mathbf{N}}$),
 $\mathfrak{F}(\text{HomCod})$
 $]_{\circ}$

Components.

lemma *RK23-components*:

shows *RK23* $\mathfrak{F}(\text{ObjMap}) =$
 (
 $\lambda a \in_{\circ} \text{cat-ordinal } (3_{\mathbf{N}})(\text{Obj}).$
 $\text{if } a = 0 \Rightarrow \mathfrak{F}(\text{ObjMap})(\text{Obj})$
 $\quad | a = 1_{\mathbf{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbf{N}})$
 $\quad | a = 2_{\mathbf{N}} \Rightarrow \mathfrak{F}(\text{ObjMap})(1_{\mathbf{N}})$
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Obj})$
)
and *RK23* $\mathfrak{F}(\text{ArrMap}) =$
 (
 $\lambda f \in_{\circ} \text{cat-ordinal } (3_{\mathbf{N}})(\text{Arr}).$
 $\text{if } f = [0, 0]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 0)$
 $\quad | f = [0, 1_{\mathbf{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbf{N}})$
 $\quad | f = [0, 2_{\mathbf{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbf{N}})$
 $\quad | f = [1_{\mathbf{N}}, 1_{\mathbf{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbf{N}}, 1_{\mathbf{N}})$
 $\quad | f = [1_{\mathbf{N}}, 2_{\mathbf{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbf{N}}, 1_{\mathbf{N}})$
 $\quad | f = [2_{\mathbf{N}}, 2_{\mathbf{N}}]_{\circ} \Rightarrow \mathfrak{F}(\text{ArrMap})(1_{\mathbf{N}}, 1_{\mathbf{N}})$
 $\quad | \text{otherwise} \Rightarrow \mathfrak{F}(\text{HomCod})(\text{Arr})$
)
and *RK23* $\mathfrak{F}(\text{HomDom}) = \text{cat-ordinal } (3_{\mathbf{N}})$
and *RK23* $\mathfrak{F}(\text{HomCod}) = \mathfrak{F}(\text{HomCod})$
<proof>

context *is-functor*

begin

lemmas *RK23-components'* = *RK23-components*[**where** $\mathfrak{F}=\mathfrak{F}$, *unfolded cat-cs-simps*]

lemmas [*cat-Kan-cs-simps*] = *RK23-components'*(3,4)

end

lemmas [*cat-Kan-cs-simps*] = *is-functor.RK23-components'*(3,4)

16.4.2 Object map

mk-VLambda *RK23-components*(1)

$| \text{vsv } \text{RK23-ObjMap-vsv}[\text{cat-Kan-cs-intros}]$
 $| \text{vdomain } \text{RK23-ObjMap-vdomain}[\text{cat-Kan-cs-simps}]$
 $| \text{app } \text{RK23-ObjMap-app}$

lemma *RK23-ObjMap-app-0*[*cat-Kan-cs-simps*]:

assumes $a = 0$
shows *RK23* $\mathfrak{F}(\text{ObjMap})(a) = \mathfrak{F}(\text{ObjMap})(\text{Obj})$
<proof>

lemma *RK23-ObjMap-app-1*[*cat-Kan-cs-simps*]:

assumes $a = 1_{\mathbf{N}}$
shows *RK23* $\mathfrak{F}(\text{ObjMap})(a) = \mathfrak{F}(\text{ObjMap})(1_{\mathbf{N}})$
<proof>

lemma *RK23-ObjMap-app-2*[*cat-Kan-cs-simps*]:
assumes $a = 2_{\mathbb{N}}$
shows $RK23 \mathfrak{F}(\text{ObjMap})(a) = \mathfrak{F}(\text{ObjMap})(1_{\mathbb{N}})$
<proof>

16.4.3 Arrow map

mk-VLambda *RK23-components*(2)
|*vsu RK23-ArrMap-vsuv*[*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 $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(0, 0)$.
<proof>

lemma *RK23-ArrMap-app-01*[*cat-Kan-cs-simps*]:
assumes $f = [0, 1_{\mathbb{N}}]_{\circ}$
shows $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbb{N}})$.
<proof>

lemma *RK23-ArrMap-app-02*[*cat-Kan-cs-simps*]:
assumes $f = [0, 2_{\mathbb{N}}]_{\circ}$
shows $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(0, 1_{\mathbb{N}})$.
<proof>

lemma *RK23-ArrMap-app-11*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 1_{\mathbb{N}}]_{\circ}$
shows $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
<proof>

lemma *RK23-ArrMap-app-12*[*cat-Kan-cs-simps*]:
assumes $f = [1_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
<proof>

lemma *RK23-ArrMap-app-22*[*cat-Kan-cs-simps*]:
assumes $f = [2_{\mathbb{N}}, 2_{\mathbb{N}}]_{\circ}$
shows $RK23 \mathfrak{F}(\text{ArrMap})(f) = \mathfrak{F}(\text{ArrMap})(1_{\mathbb{N}}, 1_{\mathbb{N}})$.
<proof>

16.4.4 RK23 is a functor

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

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

16.4.5 The fundamental property of RK23

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

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

16.5 $RK\text{-}\sigma 23$: towards the universal property of the right Kan extension along $\mathfrak{R}23$

16.5.1 Definition and elementary properties

definition $RK\text{-}\sigma 23 :: V \Rightarrow V \Rightarrow V \Rightarrow V$

where $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' =$

[
 (
 $\lambda a \in_o \text{cat-ordinal } (\mathfrak{3}_N) (\text{Obj})$.
 $\text{if } a = 0 \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj})$
 $\mid a = 1_N \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj}) \circ_{A\mathfrak{T}} (\text{HomCod}) \mathfrak{F}' (\text{ArrMap}) (\text{Obj}, 2_N)$.
 $\mid a = 2_N \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj})$
 $\mid \text{otherwise} \Rightarrow \mathfrak{T} (\text{HomCod}) (\text{Arr})$
),
 \mathfrak{F}' ,
 $RK23 \mathfrak{T}$,
 $\text{cat-ordinal } (\mathfrak{3}_N)$,
 $\mathfrak{F}' (\text{HomCod})$
]_o.

Components.

lemma $RK\text{-}\sigma 23\text{-components}$:

shows $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' (\text{NTMap}) =$

(
 $\lambda a \in_o \text{cat-ordinal } (\mathfrak{3}_N) (\text{Obj})$.
 $\text{if } a = 0 \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj})$
 $\mid a = 1_N \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj}) \circ_{A\mathfrak{T}} (\text{HomCod}) \mathfrak{F}' (\text{ArrMap}) (\text{Obj}, 2_N)$.
 $\mid a = 2_N \Rightarrow \varepsilon' (\text{NTMap}) (\text{Obj})$
 $\mid \text{otherwise} \Rightarrow \mathfrak{T} (\text{HomCod}) (\text{Arr})$
)

and $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' (\text{NTDom}) = \mathfrak{F}'$

and $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' (\text{NTCod}) = RK23 \mathfrak{T}$

and $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' (\text{NTDGDom}) = \text{cat-ordinal } (\mathfrak{3}_N)$

and $RK\text{-}\sigma 23 \mathfrak{T} \varepsilon' \mathfrak{F}' (\text{NTDGCod}) = \mathfrak{F}' (\text{HomCod})$

⟨proof⟩

context

fixes $\alpha \mathfrak{A} \mathfrak{F}' \mathfrak{T}$

assumes $\mathfrak{F}' : \text{cat-ordinal } (\mathfrak{3}_N) \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $\mathfrak{T} : \text{cat-ordinal } (2_N) \mapsto \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation \mathfrak{F}' : is-functor $\alpha \langle \text{cat-ordinal } (\mathfrak{3}_N) \rangle \mathfrak{A} \mathfrak{F}'$ ⟨proof⟩

interpretation \mathfrak{T} : is-functor $\alpha \langle \text{cat-ordinal } (2_N) \rangle \mathfrak{A} \mathfrak{T}$ ⟨proof⟩

lemmas $RK\text{-}\sigma 23\text{-components}' =$

$RK\text{-}\sigma 23\text{-components}$ [where $\mathfrak{F}' = \mathfrak{F}'$ and $\mathfrak{T} = \mathfrak{T}$, unfolded cat-cs-simps]

lemmas [cat-Kan-cs-simps] = $RK\text{-}\sigma 23\text{-components}' (2-5)$

end

16.5.2 Natural transformation map

mk-VLambda $RK\text{-}\sigma 23\text{-components}(1)$
 $|vsv\ RK\text{-}\sigma 23\text{-NTMap}\text{-vsv}[cat\text{-Kan}\text{-cs}\text{-intros}]|$
 $|vdomain\ RK\text{-}\sigma 23\text{-NTMap}\text{-vdomain}[cat\text{-Kan}\text{-cs}\text{-simps}]|$
 $|app\ RK\text{-}\sigma 23\text{-NTMap}\text{-app}|$

lemma $RK\text{-}\sigma 23\text{-NTMap}\text{-app}\text{-}0[cat\text{-Kan}\text{-cs}\text{-simps}]$:
assumes $a = 0$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon' \mathfrak{F}'(NTMap)(a) = \varepsilon'(NTMap)(0)$
 $\langle proof \rangle$

lemma (**in** $is\text{-functor}$) $RK\text{-}\sigma 23\text{-NTMap}\text{-app}\text{-}1[cat\text{-Kan}\text{-cs}\text{-simps}]$:
assumes $a = 1_{\mathbb{N}}$
shows $RK\text{-}\sigma 23\ \mathfrak{F}\ \varepsilon' \mathfrak{F}'(NTMap)(a) = \varepsilon'(NTMap)(1_{\mathbb{N}}) \circ_{A\mathfrak{B}} \mathfrak{F}'(ArrMap)(1_{\mathbb{N}}, 2_{\mathbb{N}})$.
 $\langle proof \rangle$

lemmas $[cat\text{-Kan}\text{-cs}\text{-simps}] = is\text{-functor}.RK\text{-}\sigma 23\text{-NTMap}\text{-app}\text{-}1$

lemma $RK\text{-}\sigma 23\text{-NTMap}\text{-app}\text{-}2[cat\text{-Kan}\text{-cs}\text{-simps}]$:
assumes $a = 2_{\mathbb{N}}$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon' \mathfrak{F}'(NTMap)(a) = \varepsilon'(NTMap)(1_{\mathbb{N}})$
 $\langle proof \rangle$

16.5.3 $RK\text{-}\sigma 23$ is a natural transformation

lemma $RK\text{-}\sigma 23\text{-is}\text{-ntcf}$:
assumes $\mathfrak{F}' : cat\text{-ordinal}\ (3_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\varepsilon' : \mathfrak{F}' \circ_{CF} \mathfrak{K}23 \mapsto_{CF} \mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon' \mathfrak{F}' : \mathfrak{F}' \mapsto_{CF} RK23\ \mathfrak{T} : cat\text{-ordinal}\ (3_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
 $\langle proof \rangle$

lemma $RK\text{-}\sigma 23\text{-is}\text{-ntcf}'[cat\text{-Kan}\text{-cs}\text{-intros}]$:
assumes $\mathfrak{F}' : cat\text{-ordinal}\ (3_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\varepsilon' : \mathfrak{F}' \circ_{CF} \mathfrak{K}23 \mapsto_{CF} \mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G}' = \mathfrak{F}'$
and $\mathfrak{H}' = RK23\ \mathfrak{T}$
and $\mathfrak{C}' = cat\text{-ordinal}\ (3_{\mathbb{N}})$
shows $RK\text{-}\sigma 23\ \mathfrak{T}\ \varepsilon' \mathfrak{F}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{C}' \mapsto_{C\alpha} \mathfrak{A}$
 $\langle proof \rangle$

16.6 The right Kan extension along $\mathfrak{K}23$

lemma $\varepsilon 23\text{-is}\text{-cat}\text{-rKe}$:
assumes $\mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
shows $ntcf\text{-id}\ \mathfrak{T}$:
 $RK23\ \mathfrak{T} \circ_{CF} \mathfrak{K}23 \mapsto_{CF.\tau K\varepsilon\alpha} \mathfrak{T} : cat\text{-ordinal}\ (2_{\mathbb{N}}) \mapsto_C cat\text{-ordinal}\ (3_{\mathbb{N}}) \mapsto_C \mathfrak{A}$
 $\langle proof \rangle$

16.7 $LK\text{-}\sigma 23$: towards the universal property of the left Kan extension along $\mathfrak{K}23$

16.7.1 Definition and elementary properties

definition $LK\text{-}\sigma 23 :: V \Rightarrow V \Rightarrow V \Rightarrow V$
where $LK\text{-}\sigma 23\ \mathfrak{T}\ \eta' \mathfrak{F}' =$
 $[$

(
 $\lambda a \in_{\circ} \text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}}) (\text{Obj})$.
 if $a = 0 \Rightarrow \eta'(\text{NTMap})(\emptyset)$
 | $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}'(\text{ArrMap})(\emptyset, 1_{\mathbb{N}}) \bullet \circ_{A\mathfrak{T}}(\text{HomCod}) \eta'(\text{NTMap})(\emptyset)$
 | $a = 2_{\mathbb{N}} \Rightarrow \eta'(\text{NTMap})(1_{\mathbb{N}})$
 | otherwise $\Rightarrow \mathfrak{T}(\text{HomCod})(\text{Arr})$
),
 LK23 \mathfrak{T} ,
 \mathfrak{F}' ,
 $\text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}})$,
 $\mathfrak{F}'(\text{HomCod})$
]_o.

Components.

lemma *LK- σ 23-components*:

shows $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTMap}) =$

(
 $\lambda a \in_{\circ} \text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}}) (\text{Obj})$.
 if $a = 0 \Rightarrow \eta'(\text{NTMap})(\emptyset)$
 | $a = 1_{\mathbb{N}} \Rightarrow \mathfrak{F}'(\text{ArrMap})(\emptyset, 1_{\mathbb{N}}) \bullet \circ_{A\mathfrak{T}}(\text{HomCod}) \eta'(\text{NTMap})(\emptyset)$
 | $a = 2_{\mathbb{N}} \Rightarrow \eta'(\text{NTMap})(1_{\mathbb{N}})$
 | otherwise $\Rightarrow \mathfrak{T}(\text{HomCod})(\text{Arr})$
)

and $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTDom}) = \text{LK23 } \mathfrak{T}$

and $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTCod}) = \mathfrak{F}'$

and $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTDGDom}) = \text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}})$

and $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTDGCod}) = \mathfrak{F}'(\text{HomCod})$

<proof>

context

fixes $\alpha \mathfrak{A} \mathfrak{F}' \mathfrak{T}$

assumes $\mathfrak{F}': \mathfrak{F}' : \text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}}) \mapsto \mapsto_{C\alpha} \mathfrak{A}$

and $\mathfrak{T}: \mathfrak{T} : \text{cat-ordinal } (\mathfrak{2}_{\mathbb{N}}) \mapsto \mapsto_{C\alpha} \mathfrak{A}$

begin

interpretation $\mathfrak{F}': \text{is-functor } \alpha \langle \text{cat-ordinal } (\mathfrak{3}_{\mathbb{N}}) \rangle \mathfrak{A} \mathfrak{F}'$ *<proof>*

interpretation $\mathfrak{T}: \text{is-functor } \alpha \langle \text{cat-ordinal } (\mathfrak{2}_{\mathbb{N}}) \rangle \mathfrak{A} \mathfrak{T}$ *<proof>*

lemmas *LK- σ 23-components'* =

$\text{LK-}\sigma\text{23-components}[\text{where } \mathfrak{F}' = \mathfrak{F}' \text{ and } \mathfrak{T} = \mathfrak{T}, \text{ unfolded cat-cs-simps}]$

lemmas $[\text{cat-Kan-cs-simps}] = \text{LK-}\sigma\text{23-components}'(2-5)$

end

16.7.2 Natural transformation map

mk-VLambda *LK- σ 23-components(1)*

$[\text{vsu } \text{LK-}\sigma\text{23-NTMap-vsuv}[\text{cat-Kan-cs-intros}]]$

$[\text{vdomain } \text{LK-}\sigma\text{23-NTMap-vdomain}[\text{cat-Kan-cs-simps}]]$

$[\text{app } \text{LK-}\sigma\text{23-NTMap-app}]$

lemma *LK- σ 23-NTMap-app-0* $[\text{cat-Kan-cs-simps}]$:

assumes $a = 0$

shows $\text{LK-}\sigma\text{23 } \mathfrak{T} \eta' \mathfrak{F}'(\text{NTMap})(a) = \eta'(\text{NTMap})(\emptyset)$

<proof>

lemma (in *is-functor*) *LK- σ 23-NTMap-app-1* $[\text{cat-Kan-cs-simps}]$:

assumes $a = 1_{\mathbb{N}}$
shows $LK\text{-}\sigma 23 \mathfrak{F} \eta' \mathfrak{F}'(NTMap)(a) = \mathfrak{F}'(ArrMap)(0, 1_{\mathbb{N}}) \bullet \circ_{A\mathfrak{B}} \eta'(NTMap)(0)$
 ⟨proof⟩

lemmas [cat-Kan-cs-simps] = is-functor.LK- $\sigma 23$ -NTMap-app-1

lemma LK- $\sigma 23$ -NTMap-app-2[cat-Kan-cs-simps]:

assumes $a = 2_{\mathbb{N}}$
shows $LK\text{-}\sigma 23 \mathfrak{T} \eta' \mathfrak{F}'(NTMap)(a) = \eta'(NTMap)(1_{\mathbb{N}})$
 ⟨proof⟩

16.7.3 LK- $\sigma 23$ is a natural transformation

lemma LK- $\sigma 23$ -is-ntcf:

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

lemma LK- $\sigma 23$ -is-ntcf'[cat-Kan-cs-intros]:

assumes $\mathfrak{F}' : \text{cat-ordinal } (3_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{T} : \text{cat-ordinal } (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\eta' : \mathfrak{T} \mapsto_{CF} \mathfrak{F}' \circ_{CF} \mathfrak{K}23 : \text{cat-ordinal } (2_{\mathbb{N}}) \mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G}' = LK23 \mathfrak{T}$
and $\mathfrak{H}' = \mathfrak{F}'$
and $\mathfrak{C}' = \text{cat-ordinal } (3_{\mathbb{N}})$
shows $LK\text{-}\sigma 23 \mathfrak{T} \eta' \mathfrak{F}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{C}' \mapsto_{C\alpha} \mathfrak{A}$
 ⟨proof⟩

16.8 The left Kan extension along $\mathfrak{K}23$

lemma $\eta 23$ -is-cat-rKe:

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

16.9 Pointwise Kan extensions along $\mathfrak{K}23$

lemma $\varepsilon 23$ -is-cat-pw-rKe:

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

lemma $\eta 23$ -is-cat-pw-lKe:

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

References

- [1] nLab. URL <https://ncatlab.org/nlab/show/HomePage>.
- [2] Wikipedia, 2001. URL <https://www.wikipedia.org/>.
- [3] S. Awodey. *Category Theory*. Oxford University Press, Oxford, UK, 2010. ISBN 978-0-19-958736-0.
- [4] P. Bodo. *Categories and Functors*. Academic Press, New York, 1970.
- [5] F. Haftmann. Sketch-and-Explore, 2021. URL https://isabelle.in.tum.de/library/HOL/HOL-ex/Sketch_and_Explore.html.
- [6] T. W. Hungerford. *Algebra*. Springer, New York, 2003. ISBN 978-0-387-90518-1.
- [7] D. M. Kan. Adjoint Functors. *Transactions of the American Mathematical Society*, 87(2): 294–329, 1958.
- [8] M. C. Lehner. “All Concepts are Kan Extensions”: Kan Extensions as the Most Universal of the Universal Constructions. Bachelor of Arts Thesis, Harvard College, Cambridge, MA, 2014.
- [9] S. Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer, New York, 2 edition, 2010. ISBN 978-1-4419-3123-8.
- [10] M. Milehins. Category Theory for ZFC in HOL II: Elementary Theory of 1-Categories. *Archive of Formal Proofs*, 2021.
- [11] S. Obua. Partizan Games in Isabelle/HOLZF. In K. Barkaoui, A. Cavalcanti, and A. Cerone, editors, *ICTAC 2006*, volume 4281, pages 272–286. Springer, Berlin, 2006. ISBN 978-3-540-48815-6.
- [12] L. C. Paulson. Natural Deduction as Higher-Order Resolution. *The Journal of Logic Programming*, 3(3):237–258, 1986. doi: 10.1016/0743-1066(86)90015-4.
- [13] L. C. Paulson. Zermelo Fraenkel Set Theory in Higher-Order Logic. *Archive of Formal Proofs*, 2019.
- [14] E. Riehl. *Category Theory in Context*. Emily Riehl, 2016.