Category Theory for ZFC in HOL II
Elementary Theory of 1-Categories

Mihails Milehins

March 19, 2025

Abstract

This article provides a formalization of the foundations of the theory of 1-categories in the object logic *ZFC in HOL* ([11], also see [9]) of the formal proof assistant *Isabelle* [10]. The methodology chosen for the formalization rests on the ideas that were originally expressed in [5]. Thus, in the context of this work, each category is represented as a term of the type V embedded into a stage of the von Neumann hierarchy [14].

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" [6] 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

1.1 Background

This article provides a formalization of the elementary theory of 1-categories without an additional structure. For further information see chapter Introduction in [8].

1.2 Preliminaries

named-theorems *cat-op-simps*
named-theorems *cat-op-intros*

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

named-theorems *cat-arrow-cs-intros*

1.3 CS setup for foundations

lemmas (in $\mathcal{Z}$**)** [*cat-cs-intros*] = $\mathcal{Z}$-β

2 Category

2.1 Background

lemmas [*cat-cs-simps*] = *dg-shared-cs-simps*
lemmas [*cat-cs-intros*] = *dg-shared-cs-intros*

definition *CId* :: *V*
  where [*dg-field-simps*]: *CId* = $5_{\mathbb{N}}$

2.1.1 Slicing

definition *cat-smc* :: *V* ⇒ *V*
  where *cat-smc* $\mathfrak{C}$ = [$\mathfrak{C}(\!|Obj|\!)$, $\mathfrak{C}(\!|Arr|\!)$, $\mathfrak{C}(\!|Dom|\!)$, $\mathfrak{C}(\!|Cod|\!)$, $\mathfrak{C}(\!|Comp|\!)$]$_\circ$

Components.

lemma *cat-smc-components*[*slicing-simps*]:
  shows *cat-smc* $\mathfrak{C}(\!|Obj|\!)$ = $\mathfrak{C}(\!|Obj|\!)$
    and *cat-smc* $\mathfrak{C}(\!|Arr|\!)$ = $\mathfrak{C}(\!|Arr|\!)$
    and *cat-smc* $\mathfrak{C}(\!|Dom|\!)$ = $\mathfrak{C}(\!|Dom|\!)$
    and *cat-smc* $\mathfrak{C}(\!|Cod|\!)$ = $\mathfrak{C}(\!|Cod|\!)$
    and *cat-smc* $\mathfrak{C}(\!|Comp|\!)$ = $\mathfrak{C}(\!|Comp|\!)$
  ⟨*proof*⟩

Regular definitions.

lemma *cat-smc-is-arr*[*slicing-simps*]:
  $f : a \mapsto_{\textit{cat-smc}\ \mathfrak{C}}\ b \longleftrightarrow f : a \mapsto_{\mathfrak{C}}\ b$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-is-arr*[*THEN iffD2*]

lemma *cat-smc-composable-arrs*[*slicing-simps*]:
  composable-arrs (*cat-smc* $\mathfrak{C}$) = *composable-arrs* $\mathfrak{C}$
  ⟨*proof*⟩

lemma *cat-smc-is-monic-arr*[*slicing-simps*]:
  $f : a \mapsto_{mon\,\textit{cat-smc}\ \mathfrak{C}}\ b \longleftrightarrow f : a \mapsto_{mon\,\mathfrak{C}}\ b$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-is-monic-arr*[*THEN iffD2*]

lemma *cat-smc-is-epic-arr*[*slicing-simps*]:
  $f : a \mapsto_{epi\,\textit{cat-smc}\ \mathfrak{C}}\ b \longleftrightarrow f : a \mapsto_{epi\,\mathfrak{C}}\ b$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-is-epic-arr*[*THEN iffD2*]

lemma *cat-smc-is-idem-arr*[*slicing-simps*]:
  $f : \mapsto_{ide\,\textit{cat-smc}\ \mathfrak{C}}\ b \longleftrightarrow f : \mapsto_{ide\,\mathfrak{C}}\ b$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-is-idem-arr*[*THEN iffD2*]

lemma *cat-smc-obj-terminal*[*slicing-simps*]:
  obj-terminal (*cat-smc* $\mathfrak{C}$) *a* ⟷ *obj-terminal* $\mathfrak{C}$ *a*
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-obj-terminal*[*THEN iffD2*]

lemma *cat-smc-obj-intial*[*slicing-simps*]:
obj-initial (*cat-smc* $\mathfrak{C}$) $a \longleftrightarrow$ *obj-initial* $\mathfrak{C}$ a
⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-obj-intial*[*THEN iffD2*]

lemma *cat-smc-obj-null*[*slicing-simps*]:
obj-null (*cat-smc* $\mathfrak{C}$) $a \longleftrightarrow$ *obj-null* $\mathfrak{C}$ a
⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-obj-null*[*THEN iffD2*]

lemma *cat-smc-is-zero-arr*[*slicing-simps*]:
$f : a \mapsto_{0 \, cat\text{-}smc \, \mathfrak{C}} b \longleftrightarrow f : a \mapsto_{0 \, \mathfrak{C}} b$
⟨*proof*⟩

lemmas [*slicing-intros*] = *cat-smc-is-zero-arr*[*THEN iffD2*]

2.2 Definition and elementary properties

The definition of a category that is used in this work is is similar to the definition that can be found in Chapter I-2 in [7]. The amendments to the definitions that are associated with size have already been explained in [8].

locale *category* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *CId*: *vsv* ‹$\mathfrak{C}(\!|CId|\!)$› **for** α $\mathfrak{C}$ +
assumes *cat-length*[*cat-cs-simps*]: *vcard* $\mathfrak{C} = 6_{\mathbb{N}}$
and *cat-semicategory*[*slicing-intros*]: *semicategory* α (*cat-smc* $\mathfrak{C}$)
and *cat-CId-vdomain*[*cat-cs-simps*]: $\mathcal{D}_{\circ}(\mathfrak{C}(\!|CId|\!)) = \mathfrak{C}(\!|Obj|\!)$
and *cat-CId-is-arr*[*cat-cs-intros*]: $a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
and *cat-CId-left-left*[*cat-cs-simps*]:
$f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A \mathfrak{C}} f = f$
and *cat-CId-right-left*[*cat-cs-simps*]:
$f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A \mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$

lemmas [*cat-cs-simps*] =
category.cat-length
category.cat-CId-vdomain
category.cat-CId-left-left
category.cat-CId-right-left

lemma (**in** *category*) *cat-CId-is-arr'*[*cat-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b = a$ **and** $c = a$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) : b \mapsto_{\mathfrak{C}'} c$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-CId-is-arr'*

lemma (**in** *category*) *cat-CId-is-arr''*[*cat-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
shows $f : a \mapsto_{\mathfrak{C}} a$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-CId-is-arr''*

lemmas [*slicing-intros*] = *category.cat-semicategory*

lemma (**in** *category*) *cat-CId-vrange*: $\mathcal{R}_{\circ}(\mathfrak{C}(\!|CId|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$

⟨*proof*⟩

Rules.

lemma (in *category*) *category-axioms'*[*cat-cs-intros*]:
assumes $\alpha' = \alpha$
shows *category* α' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *category-def*[*unfolded category-axioms-def*]
|*intro categoryI*|
|*dest categoryD*[*dest*]|
|*elim categoryE*[*elim*]|

lemma *categoryI'*:
assumes $\mathcal{Z}$ α
and *vfsequence* $\mathfrak{C}$
and *vcard* $\mathfrak{C} = 6_{\mathbb{N}}$
and *vsv* ($\mathfrak{C}(\!|Dom|\!)$)
and *vsv* ($\mathfrak{C}(\!|Cod|\!)$)
and *vsv* ($\mathfrak{C}(\!|Comp|\!)$)
and *vsv* ($\mathfrak{C}(\!|CId|\!)$)
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and $\bigwedge gf.\ gf \in_\circ \mathcal{D}_\circ\ (\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
$(\exists g\ f\ b\ c\ a.\ gf = [g, f]_\circ \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|CId|\!)$) = $\mathfrak{C}(\!|Obj|\!)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
and $\bigwedge c\ d\ h\ b\ g\ a\ f.\ [\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$
and $\bigwedge a.\ a \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
and $\bigwedge b\ c\ f.\ f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$
and $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ$ *Vset* α
and $\bigwedge A\ B.\ [\![\ A \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ B \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ A \in_\circ Vset\ \alpha;\ B \in_\circ Vset\ \alpha\ ]\!] \Longrightarrow$
$(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ \mathfrak{C}\ a\ b) \in_\circ Vset\ \alpha$
shows *category* α $\mathfrak{C}$
⟨*proof*⟩

lemma *categoryD'*:
assumes *category* α $\mathfrak{C}$
shows $\mathcal{Z}$ α
and *vfsequence* $\mathfrak{C}$
and *vcard* $\mathfrak{C} = 6_{\mathbb{N}}$
and *vsv* ($\mathfrak{C}(\!|Dom|\!)$)
and *vsv* ($\mathfrak{C}(\!|Cod|\!)$)
and *vsv* ($\mathfrak{C}(\!|Comp|\!)$)
and *vsv* ($\mathfrak{C}(\!|CId|\!)$)
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and $\bigwedge gf.\ gf \in_\circ \mathcal{D}_\circ\ (\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
$(\exists g\ f\ b\ c\ a.\ gf = [g, f]_\circ \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
and $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|CId|\!)$) = $\mathfrak{C}(\!|Obj|\!)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
and $\bigwedge c\ d\ h\ b\ g\ a\ f.\ [\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$

$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$
and $\bigwedge a.\ a \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
and $\bigwedge b\ c\ f.\ f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$
and $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and $\bigwedge A\ B.\ [\![\ A \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ B \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ A \in_\circ Vset\ \alpha;\ B \in_\circ Vset\ \alpha\ ]\!] \Longrightarrow$
$(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ \mathfrak{C}\ a\ b) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *categoryE′*:
assumes *category* $\alpha\ \mathfrak{C}$
obtains $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{C}$
and *vcard* $\mathfrak{C} = \mathfrak{6}_{\mathbb{N}}$
and *vsv* $(\mathfrak{C}(\!|Dom|\!))$
and *vsv* $(\mathfrak{C}(\!|Cod|\!))$
and *vsv* $(\mathfrak{C}(\!|Comp|\!))$
and *vsv* $(\mathfrak{C}(\!|CId|\!))$
and $\mathcal{D}_\circ\ (\mathfrak{C}(\!|Dom|\!)) = \mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ\ (\mathfrak{C}(\!|Dom|\!)) \subseteq_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathcal{D}_\circ\ (\mathfrak{C}(\!|Cod|\!)) = \mathfrak{C}(\!|Arr|\!)$
and $\mathcal{R}_\circ\ (\mathfrak{C}(\!|Cod|\!)) \subseteq_\circ \mathfrak{C}(\!|Obj|\!)$
and $\bigwedge gf.\ gf \in_\circ \mathcal{D}_\circ\ (\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
$(\exists\, g\ f\ b\ c\ a.\ gf = [g, f]_\circ \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
and $\mathcal{D}_\circ\ (\mathfrak{C}(\!|CId|\!)) = \mathfrak{C}(\!|Obj|\!)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
and $\bigwedge c\ d\ h\ b\ g\ a\ f.\ [\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$
and $\bigwedge a.\ a \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
and $\bigwedge b\ c\ f.\ f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$
and $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and $\bigwedge A\ B.\ [\![\ A \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ B \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ A \in_\circ Vset\ \alpha;\ B \in_\circ Vset\ \alpha\ ]\!] \Longrightarrow$
$(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ \mathfrak{C}\ a\ b) \in_\circ Vset\ \alpha$
⟨*proof*⟩

Slicing.

context *category*
begin

interpretation *smc*: *semicategory* α ‹*cat-smc* $\mathfrak{C}$› ⟨*proof*⟩

sublocale *Dom*: *vsv* ‹$\mathfrak{C}(\!|Dom|\!)$›
⟨*proof*⟩
sublocale *Cod*: *vsv* ‹$\mathfrak{C}(\!|Cod|\!)$›
⟨*proof*⟩
sublocale *Comp*: *pbinop* ‹$\mathfrak{C}(\!|Arr|\!)$› ‹$\mathfrak{C}(\!|Comp|\!)$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
cat-Dom-vdomain[*cat-cs-simps*] = *smc.smc-Dom-vdomain*
and *cat-Dom-vrange* = *smc.smc-Dom-vrange*
and *cat-Cod-vdomain*[*cat-cs-simps*] = *smc.smc-Cod-vdomain*
and *cat-Cod-vrange* = *smc.smc-Cod-vrange*
and *cat-Obj-vsubset-Vset* = *smc.smc-Obj-vsubset-Vset*
and *cat-Hom-vifunion-in-Vset*[*cat-cs-intros*] = *smc.smc-Hom-vifunion-in-Vset*
and *cat-Obj-if-Dom-vrange* = *smc.smc-Obj-if-Dom-vrange*
and *cat-Obj-if-Cod-vrange* = *smc.smc-Obj-if-Cod-vrange*

and *cat-is-arrD* = *smc.smc-is-arrD*
and *cat-is-arrE*[*elim*] = *smc.smc-is-arrE*
and *cat-in-ArrE*[*elim*] = *smc.smc-in-ArrE*
and *cat-Hom-in-Vset*[*cat-cs-intros*] = *smc.smc-Hom-in-Vset*
and *cat-Arr-vsubset-Vset* = *smc.smc-Arr-vsubset-Vset*
and *cat-Dom-vsubset-Vset* = *smc.smc-Dom-vsubset-Vset*
and *cat-Cod-vsubset-Vset* = *smc.smc-Cod-vsubset-Vset*
and *cat-Obj-in-Vset* = *smc.smc-Obj-in-Vset*
and *cat-in-Obj-in-Vset*[*cat-cs-intros*] = *smc.smc-in-Obj-in-Vset*
and *cat-Arr-in-Vset* = *smc.smc-Arr-in-Vset*
and *cat-in-Arr-in-Vset*[*cat-cs-intros*] = *smc.smc-in-Arr-in-Vset*
and *cat-Dom-in-Vset* = *smc.smc-Dom-in-Vset*
and *cat-Cod-in-Vset* = *smc.smc-Cod-in-Vset*
and *cat-semicategory-if-ge-Limit* = *smc.smc-semicategory-if-ge-Limit*
and *cat-Dom-app-in-Obj* = *smc.smc-Dom-app-in-Obj*
and *cat-Cod-app-in-Obj* = *smc.smc-Cod-app-in-Obj*
and *cat-Arr-vempty-if-Obj-vempty* = *smc.smc-Arr-vempty-if-Obj-vempty*
and *cat-Dom-vempty-if-Arr-vempty* = *smc.smc-Dom-vempty-if-Arr-vempty*
and *cat-Cod-vempty-if-Arr-vempty* = *smc.smc-Cod-vempty-if-Arr-vempty*

lemmas [*cat-cs-intros*] = *cat-is-arrD*(*2*,*3*)

lemmas-with [*unfolded slicing-simps slicing-commute*]:
cat-Comp-vdomain = *smc.smc-Comp-vdomain*
and *cat-Comp-is-arr*[*cat-cs-intros*] = *smc.smc-Comp-is-arr*
and *cat-Comp-assoc*[*cat-cs-intros*] = *smc.smc-Comp-assoc*
and *cat-Comp-vdomainI*[*cat-cs-intros*] = *smc.smc-Comp-vdomainI*
and *cat-Comp-vdomainE*[*elim!*] = *smc.smc-Comp-vdomainE*
and *cat-Comp-vdomain-is-composable-arrs* =
smc.smc-Comp-vdomain-is-composable-arrs
and *cat-Comp-vrange* = *smc.smc-Comp-vrange*
and *cat-Comp-vsubset-Vset* = *smc.smc-Comp-vsubset-Vset*
and *cat-Comp-in-Vset* = *smc.smc-Comp-in-Vset*
and *cat-Comp-vempty-if-Arr-vempty* = *smc.smc-Comp-vempty-if-Arr-vempty*
and *cat-assoc-helper* = *smc.smc-assoc-helper*
and *cat-pattern-rectangle-right* = *smc.smc-pattern-rectangle-right*
and *cat-pattern-rectangle-left* = *smc.smc-pattern-rectangle-left*
and *is-epic-arrI* = *smc.is-epic-arrI*
and *is-epic-arrD*[*dest*] = *smc.is-epic-arrD*
and *is-epic-arrE*[*elim!*] = *smc.is-epic-arrE*
and *cat-comp-is-monic-arr*[*cat-arrow-cs-intros*] = *smc.smc-Comp-is-monic-arr*
and *cat-comp-is-epic-arr*[*cat-arrow-cs-intros*] = *smc.smc-Comp-is-epic-arr*
and *cat-comp-is-monic-arr-is-monic-arr* =
smc.smc-Comp-is-monic-arr-is-monic-arr
and *cat-is-zero-arr-comp-right*[*cat-arrow-cs-intros*] =
smc.smc-is-zero-arr-Comp-right
and *cat-is-zero-arr-comp-left*[*cat-arrow-cs-intros*] =
smc.smc-is-zero-arr-Comp-left

lemma *cat-Comp-is-arr'*[*cat-cs-intros*]:
assumes $g : b \mapsto_{\mathfrak{C}} c$
and $f : a \mapsto_{\mathfrak{C}} b$
and $\mathfrak{C}' = \mathfrak{C}$
shows $g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}'} c$
⟨*proof*⟩

end

lemmas [*cat-cs-simps*] = *is-idem-arrD*(*2*)

lemmas [*cat-cs-simps*] = *category.cat-Comp-assoc*

lemmas [*cat-cs-intros*] =
category.cat-Comp-vdomainI
category.cat-Hom-in-Vset
category.cat-is-arrD(*1–3*)
category.cat-Comp-is-arr′
category.cat-Comp-is-arr

lemmas [*cat-arrow-cs-intros*] =
is-monic-arrD(*1*)
is-epic-arr-is-arr
category.cat-comp-is-monic-arr
category.cat-comp-is-epic-arr
category.cat-is-zero-arr-comp-right
category.cat-is-zero-arr-comp-left

lemmas [*cat-cs-intros*] = *HomI*
lemmas [*cat-cs-simps*] = *in-Hom-iff*

Elementary properties.

lemma *cat-eqI*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{A}(\!|Obj|\!) = \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}(\!|Arr|\!) = \mathfrak{B}(\!|Arr|\!)$
and $\mathfrak{A}(\!|Dom|\!) = \mathfrak{B}(\!|Dom|\!)$
and $\mathfrak{A}(\!|Cod|\!) = \mathfrak{B}(\!|Cod|\!)$
and $\mathfrak{A}(\!|Comp|\!) = \mathfrak{B}(\!|Comp|\!)$
and $\mathfrak{A}(\!|CId|\!) = \mathfrak{B}(\!|CId|\!)$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

lemma *cat-smc-eqI*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{A}(\!|CId|\!) = \mathfrak{B}(\!|CId|\!)$
and *cat-smc* $\mathfrak{A}$ = *cat-smc* $\mathfrak{B}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *category*) *cat-def*:
$\mathfrak{C} = [\mathfrak{C}(\!|Obj|\!), \mathfrak{C}(\!|Arr|\!), \mathfrak{C}(\!|Dom|\!), \mathfrak{C}(\!|Cod|\!), \mathfrak{C}(\!|Comp|\!), \mathfrak{C}(\!|CId|\!)]_{\circ}$
⟨*proof*⟩

Size.

lemma (**in** *category*) *cat-CId-vsubset-Vset*: $\mathfrak{C}(\!|CId|\!) \subseteq_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-category-in-Vset-4*: $\mathfrak{C} \in_{\circ} Vset\ (\alpha + 4_{\mathbb{N}})$
⟨*proof*⟩

lemma (**in** *category*) *cat-CId-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows $\mathfrak{C}(\!|CId|\!) \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (in *category*) *cat-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $\mathfrak{C} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (in *category*) *cat-category-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *category* $\beta\ \mathfrak{C}$
⟨*proof*⟩

lemma *tiny-category*[*simp*]: *small* $\{\mathfrak{C}.\ category\ \alpha\ \mathfrak{C}\}$
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *categories-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *set* $\{\mathfrak{C}.\ category\ \alpha\ \mathfrak{C}\} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma *category-if-category*:
assumes *category* $\beta\ \mathfrak{C}$
and $\mathcal{Z}\ \alpha$
and $\mathfrak{C}\lfloor Obj \rfloor \subseteq_\circ Vset\ \alpha$
and $\bigwedge A\ B.\ [\![\ A \subseteq_\circ \mathfrak{C}\lfloor Obj \rfloor;\ B \subseteq_\circ \mathfrak{C}\lfloor Obj \rfloor;\ A \in_\circ Vset\ \alpha;\ B \in_\circ Vset\ \alpha\ ]\!] \Longrightarrow$
$(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ \mathfrak{C}\ a\ b) \in_\circ Vset\ \alpha$
shows *category* $\alpha\ \mathfrak{C}$
⟨*proof*⟩

Further elementary properties.

sublocale *category* $\subseteq$ *CId*: *v11* ‹$\mathfrak{C}\lfloor CId \rfloor$›
⟨*proof*⟩

lemma (in *category*) *cat-CId-vempty-if-Arr-vempty*:
assumes $\mathfrak{C}\lfloor Arr \rfloor = 0$
shows $\mathfrak{C}\lfloor CId \rfloor = 0$
⟨*proof*⟩

2.3 Opposite category

2.3.1 Definition and elementary properties

See Chapter II-2 in [7].

definition *op-cat* :: $V \Rightarrow V$
where *op-cat* $\mathfrak{C} = [\mathfrak{C}\lfloor Obj \rfloor, \mathfrak{C}\lfloor Arr \rfloor, \mathfrak{C}\lfloor Cod \rfloor, \mathfrak{C}\lfloor Dom \rfloor, fflip\ (\mathfrak{C}\lfloor Comp \rfloor), \mathfrak{C}\lfloor CId \rfloor]_\circ$

Components.

lemma *op-cat-components*:
shows [*cat-op-simps*]: *op-cat* $\mathfrak{C}\lfloor Obj \rfloor = \mathfrak{C}\lfloor Obj \rfloor$
and [*cat-op-simps*]: *op-cat* $\mathfrak{C}\lfloor Arr \rfloor = \mathfrak{C}\lfloor Arr \rfloor$
and [*cat-op-simps*]: *op-cat* $\mathfrak{C}\lfloor Dom \rfloor = \mathfrak{C}\lfloor Cod \rfloor$
and [*cat-op-simps*]: *op-cat* $\mathfrak{C}\lfloor Cod \rfloor = \mathfrak{C}\lfloor Dom \rfloor$
and *op-cat* $\mathfrak{C}\lfloor Comp \rfloor = fflip\ (\mathfrak{C}\lfloor Comp \rfloor)$
and [*cat-op-simps*]: *op-cat* $\mathfrak{C}\lfloor CId \rfloor = \mathfrak{C}\lfloor CId \rfloor$
⟨*proof*⟩

lemma *op-cat-component-intros*[*cat-op-intros*]:
shows $a \in_\circ \mathfrak{C}\lfloor Obj \rfloor \Longrightarrow a \in_\circ op\text{-}cat\ \mathfrak{C}\lfloor Obj \rfloor$

and $f \in_\circ \mathfrak{C}(\!|Arr|\!) \Longrightarrow f \in_\circ op\text{-}cat\ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

Slicing.

lemma *cat-smc-op-cat*[*slicing-commute*]: $op\text{-}smc\ (cat\text{-}smc\ \mathfrak{C}) = cat\text{-}smc\ (op\text{-}cat\ \mathfrak{C})$
⟨*proof*⟩

lemma (**in** *category*) *op-smc-op-cat*[*cat-op-simps*]: $op\text{-}smc\ (op\text{-}cat\ \mathfrak{C}) = cat\text{-}smc\ \mathfrak{C}$
⟨*proof*⟩

lemma *op-cat-is-arr*[*cat-op-simps*]: $f : b \mapsto_{op\text{-}cat\ \mathfrak{C}} a \longleftrightarrow f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *op-cat-is-arr*[*THEN iffD2*]

lemma *op-cat-Hom*[*cat-op-simps*]: $Hom\ (op\text{-}cat\ \mathfrak{C})\ a\ b = Hom\ \mathfrak{C}\ b\ a$
⟨*proof*⟩

lemma *op-cat-obj-initial*[*cat-op-simps*]:
$obj\text{-}initial\ (op\text{-}cat\ \mathfrak{C})\ a \longleftrightarrow obj\text{-}terminal\ \mathfrak{C}\ a$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *op-cat-obj-initial*[*THEN iffD2*]

lemma *op-cat-obj-terminal*[*cat-op-simps*]:
$obj\text{-}terminal\ (op\text{-}cat\ \mathfrak{C})\ a \longleftrightarrow obj\text{-}initial\ \mathfrak{C}\ a$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *op-cat-obj-terminal*[*THEN iffD2*]

lemma *op-cat-obj-null*[*cat-op-simps*]: $obj\text{-}null\ (op\text{-}cat\ \mathfrak{C})\ a \longleftrightarrow obj\text{-}null\ \mathfrak{C}\ a$
⟨*proof*⟩

lemmas [*cat-op-intros*] = *op-cat-obj-null*[*THEN iffD2*]

context *category*
begin

interpretation *smc*: *semicategory* α ‹*cat-smc* $\mathfrak{C}$› ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
op-cat-Comp-vrange[*cat-op-simps*] = *smc.op-smc-Comp-vrange*
and *op-cat-Comp*[*cat-op-simps*] = *smc.op-smc-Comp*
and *op-cat-is-epic-arr*[*cat-op-simps*] = *smc.op-smc-is-epic-arr*
and *op-cat-is-monic-arr*[*cat-op-simps*] = *smc.op-smc-is-monic-arr*
and *op-cat-is-zero-arr*[*cat-op-simps*] = *smc.op-smc-is-zero-arr*

end

lemmas [*cat-op-simps*] =
category.op-cat-Comp-vrange
category.op-cat-Comp
category.op-cat-is-epic-arr
category.op-cat-is-monic-arr
category.op-cat-is-zero-arr

context
fixes $\mathfrak{C} :: V$

begin

lemmas-with [
  where $\mathfrak{C}$=‹*cat-smc* $\mathfrak{C}$›, *unfolded slicing-simps slicing-commute*[*symmetric*]
  ]:
  op-cat-Comp-vdomain[*cat-op-simps*] = *op-smc-Comp-vdomain*

end

Elementary properties.

lemma *op-cat-vsv*[*cat-op-intros*]: *vsv* (*op-cat* $\mathfrak{C}$) ⟨*proof*⟩

2.3.2 Further properties

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

lemmas *category-op*[*cat-op-intros*] = *category.category-op*

lemma (**in** *category*) *cat-op-cat-op-cat*[*cat-op-simps*]: *op-cat* (*op-cat* $\mathfrak{C}$) = $\mathfrak{C}$
⟨*proof*⟩

lemmas *cat-op-cat-op-cat*[*cat-op-simps*] = *category.cat-op-cat-op-cat*

lemma *eq-op-cat-iff*[*cat-op-simps*]:
  assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$
  shows *op-cat* $\mathfrak{A}$ = *op-cat* $\mathfrak{B}$ ⟷ $\mathfrak{A}$ = $\mathfrak{B}$
⟨*proof*⟩

2.4 Monic arrow and epic arrow

lemma (**in** *category*) *cat-CId-is-monic-arr*[*cat-arrow-cs-intros*]:
  assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
  shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{mon\mathfrak{C}} a$
  ⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-CId-is-monic-arr*

lemma (**in** *category*) *cat-CId-is-epic-arr*[*cat-arrow-cs-intros*]:
  assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
  shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{epi\mathfrak{C}} a$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-CId-is-epic-arr*

2.5 Right inverse and left inverse of an arrow

See Chapter I-5 in [7].

definition *is-right-inverse* :: $V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *is-right-inverse* $\mathfrak{C}$ g f =
    ($\exists a\ b.\ g : b \mapsto_{\mathfrak{C}} a \wedge f : a \mapsto_{\mathfrak{C}} b \wedge f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$)

definition *is-left-inverse* :: $V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *is-left-inverse* $\mathfrak{C} \equiv$ *is-right-inverse* (*op-cat* $\mathfrak{C}$)

Rules.

lemma *is-right-inverseI*:

assumes $g : b \mapsto_{\mathfrak{C}} a$ **and** $f : a \mapsto_{\mathfrak{C}} b$ **and** $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
shows *is-right-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma *is-right-inverseD*[*dest*]:
assumes *is-right-inverse* $\mathfrak{C}$ g f
shows $\exists\, a\ b.\ g : b \mapsto_{\mathfrak{C}} a \wedge f : a \mapsto_{\mathfrak{C}} b \wedge f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
⟨*proof*⟩

lemma *is-right-inverseE*[*elim*]:
assumes *is-right-inverse* $\mathfrak{C}$ g f
obtains a b **where** $g : b \mapsto_{\mathfrak{C}} a$
and $f : a \mapsto_{\mathfrak{C}} b$
and $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
⟨*proof*⟩

lemma (**in** *category*) *is-left-inverseI*:
assumes $g : b \mapsto_{\mathfrak{C}} a$ **and** $f : a \mapsto_{\mathfrak{C}} b$ **and** $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
shows *is-left-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma (**in** *category*) *is-left-inverseD*[*dest*]:
assumes *is-left-inverse* $\mathfrak{C}$ g f
shows $\exists\, a\ b.\ g : b \mapsto_{\mathfrak{C}} a \wedge f : a \mapsto_{\mathfrak{C}} b \wedge g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
⟨*proof*⟩

lemma (**in** *category*) *is-left-inverseE*[*elim*]:
assumes *is-left-inverse* $\mathfrak{C}$ g f
obtains a b **where** $g : b \mapsto_{\mathfrak{C}} a$
and $f : a \mapsto_{\mathfrak{C}} b$
and $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
⟨*proof*⟩

Elementary properties.

lemma (**in** *category*) *op-cat-is-left-inverse*[*cat-op-simps*]:
is-left-inverse (*op-cat* $\mathfrak{C}$) g $f \longleftrightarrow$ *is-right-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-is-left-inverse*

lemmas [*cat-op-intros*] = *category.op-cat-is-left-inverse*[*THEN iffD2*]

lemma (**in** *category*) *op-cat-is-right-inverse*[*cat-op-simps*]:
is-right-inverse (*op-cat* $\mathfrak{C}$) g $f \longleftrightarrow$ *is-left-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-is-right-inverse*

lemmas [*cat-op-intros*] = *category.op-cat-is-right-inverse*[*THEN iffD2*]

2.6 Inverse of an arrow

See Chapter I-5 in [7].

definition *is-inverse* :: $V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-inverse* $\mathfrak{C}$ g f =
(
$\exists\, a\ b.$

$g : b \mapsto_{\mathfrak{C}} a \wedge$
$f : a \mapsto_{\mathfrak{C}} b \wedge$
$g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor) \wedge$
$f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor b \rfloor)$
)

Rules.

lemma *is-inverseI*:
assumes $g : b \mapsto_{\mathfrak{C}} a$
and $f : a \mapsto_{\mathfrak{C}} b$
and $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor)$
and $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor b \rfloor)$
shows *is-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma *is-inverseD*[*dest*]:
assumes *is-inverse* $\mathfrak{C}$ g f
shows
(
$\exists a\ b.$
$g : b \mapsto_{\mathfrak{C}} a \wedge$
$f : a \mapsto_{\mathfrak{C}} b \wedge$
$g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor) \wedge$
$f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor b \rfloor)$
)
⟨*proof*⟩

lemma *is-inverseE*[*elim*]:
assumes *is-inverse* $\mathfrak{C}$ g f
obtains a b **where** $g : b \mapsto_{\mathfrak{C}} a$
and $f : a \mapsto_{\mathfrak{C}} b$
and $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor)$
and $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor b \rfloor)$
⟨*proof*⟩

Elementary properties.

lemma (in *category*) *op-cat-is-inverse*[*cat-op-simps*]:
is-inverse (*op-cat* $\mathfrak{C}$) g f ⟷ *is-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-is-inverse*

lemmas [*cat-op-intros*] = *category.op-cat-is-inverse*[*THEN iffD2*]

lemma *is-inverse-sym*: *is-inverse* $\mathfrak{C}$ g f ⟷ *is-inverse* $\mathfrak{C}$ f g
⟨*proof*⟩

lemma (in *category*) *cat-is-inverse-eq*:
— See Chapter I-5 in [7].
assumes *is-inverse* $\mathfrak{C}$ h f **and** *is-inverse* $\mathfrak{C}$ g f
shows $h = g$
⟨*proof*⟩

lemma *is-inverse-Comp-CId-left*:
— See Chapter I-5 in [7].
assumes *is-inverse* $\mathfrak{C}$ g' g **and** $g : a \mapsto_{\mathfrak{C}} b$
shows $g' \circ_{A\mathfrak{C}} g = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor)$
⟨*proof*⟩

lemma *is-inverse-Comp-CId-right*:
assumes *is-inverse* $\mathfrak{C}$ g' g **and** $g : a \mapsto_{\mathfrak{C}} b$
shows $g \circ_{A\mathfrak{C}} g' = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor b \rfloor)$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-inverse-Comp*:
— See Chapter I-5 in [7].
assumes *gbc*[*intro*]: $g : b \mapsto_{\mathfrak{C}} c$
and *fab*[*intro*]: $f : a \mapsto_{\mathfrak{C}} b$
and *g'g*[*intro*]: *is-inverse* $\mathfrak{C}$ g' g
and *f'f*[*intro*]: *is-inverse* $\mathfrak{C}$ f' f
shows *is-inverse* $\mathfrak{C}$ $(f' \circ_{A\mathfrak{C}} g')$ $(g \circ_{A\mathfrak{C}} f)$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-inverse-Comp'*:
assumes $g : b \mapsto_{\mathfrak{C}} c$
and $f : a \mapsto_{\mathfrak{C}} b$
and *is-inverse* $\mathfrak{C}$ g' g
and *is-inverse* $\mathfrak{C}$ f' f
and $f'g' = f' \circ_{A\mathfrak{C}} g'$
and $gf = g \circ_{A\mathfrak{C}} f$
shows *is-inverse* $\mathfrak{C}$ $f'g'$ gf
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-is-inverse-Comp'*

lemma *is-inverse-is-right-inverse*[*dest*]:
assumes *is-inverse* $\mathfrak{C}$ g f
shows *is-right-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma (**in** *category*) *cat-is-inverse-is-left-inverse*[*dest*]:
assumes *is-inverse* $\mathfrak{C}$ g f
shows *is-left-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma (**in** *category*) *cat-is-right-left-inverse-is-inverse*:
assumes *is-right-inverse* $\mathfrak{C}$ g f *is-left-inverse* $\mathfrak{C}$ g f
shows *is-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

2.7 Isomorphism

See Chapter I-5 in [7].

definition *is-iso-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-iso-arr* $\mathfrak{C}$ a b $f \longleftrightarrow$
$(f : a \mapsto_{\mathfrak{C}} b \wedge (\exists g.\ \textit{is-inverse}\ \mathfrak{C}\ g\ f))$

syntax *-is-iso-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
$(\langle \text{-} : \text{-} \mapsto_{iso^{\mathfrak{1}}} \text{-} \rangle\ [51, 51, 51]\ 51)$
syntax-consts *-is-iso-arr* ⇌ *is-iso-arr*
translations $f : a \mapsto_{iso\mathfrak{C}} b \rightleftharpoons$ *CONST is-iso-arr* $\mathfrak{C}$ a b f

Rules.

lemma *is-iso-arrI*:
assumes $f : a \mapsto_{\mathfrak{C}} b$ **and** *is-inverse* $\mathfrak{C}$ g f

shows $f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

lemma *is-iso-arrD*[*dest*]:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows $f : a \mapsto_{\mathfrak{C}} b$ **and** $\exists g.$ *is-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma *is-iso-arrE*[*elim*]:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
obtains g **where** $f : a \mapsto_{\mathfrak{C}} b$ **and** *is-inverse* $\mathfrak{C}$ g f
⟨*proof*⟩

lemma *is-iso-arrE'*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
obtains g **where** $g : b \mapsto_{iso\mathfrak{C}} a$
and $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
and $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
⟨*proof*⟩

Elementary properties.

lemma (**in** *category*) *op-cat-is-iso-arr*[*cat-op-simps*]:
$f : b \mapsto_{iso\,op\text{-}cat\,\mathfrak{C}} a \longleftrightarrow f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-is-iso-arr*

lemmas [*cat-op-intros*] = *category.op-cat-is-iso-arr*[*THEN iffD2*]

lemma (**in** *category*) *is-iso-arrI'*:
assumes $f : a \mapsto_{\mathfrak{C}} b$
and $g : b \mapsto_{\mathfrak{C}} a$
and $g \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
and $f \circ_{A\mathfrak{C}} g = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
shows $f : a \mapsto_{iso\mathfrak{C}} b$ **and** $g : b \mapsto_{iso\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-inverse-is-iso-arr*:
assumes $f : a \mapsto_{\mathfrak{C}} b$ **and** *is-inverse* $\mathfrak{C}$ g f
shows $g : b \mapsto_{iso\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *category*) *cat-Comp-is-iso-arr*[*cat-arrow-cs-intros*]:
assumes $g : b \mapsto_{iso\mathfrak{C}} c$ **and** $f : a \mapsto_{iso\mathfrak{C}} b$
shows $g \circ_{A\mathfrak{C}} f : a \mapsto_{iso\mathfrak{C}} c$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-Comp-is-iso-arr*

lemma (**in** *category*) *cat-CId-is-iso-arr*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{iso\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *category*) *cat-CId-is-iso-arr'*[*cat-arrow-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{C}' = \mathfrak{C}$
and $b = a$

and $c = a$
shows $\mathfrak{C}(\lfloor CId \rfloor)(\lfloor a \rfloor) : b \mapsto_{iso\mathfrak{C}'} c$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-CId-is-iso-arr'*

lemma (**in** *category*) *cat-is-iso-arr-is-monic-arr*[*cat-arrow-cs-intros*]:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows $f : a \mapsto_{mon\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-is-iso-arr-is-monic-arr*

lemma (**in** *category*) *cat-is-iso-arr-is-epic-arr*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows $f : a \mapsto_{epi\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-is-iso-arr-is-epic-arr*

lemma (**in** *category*) *cat-is-iso-arr-if-is-monic-arr-is-right-inverse*:
assumes $f : a \mapsto_{mon\mathfrak{C}} b$ **and** *is-right-inverse* $\mathfrak{C}$ g f
shows $f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-iso-arr-if-is-epic-arr-is-left-inverse*:
assumes $f : a \mapsto_{epi\mathfrak{C}} b$ **and** *is-left-inverse* $\mathfrak{C}$ g f
shows $f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

2.8 The inverse arrow

See Chapter I-5 in [7].

definition *the-inverse* :: $V \Rightarrow V \Rightarrow V$ (‹(-$^{-1}{}_{C1}$)› [*1000*] *999*)
where $f^{-1}{}_{C\mathfrak{C}} = (THE\ g.\ is\text{-}inverse\ \mathfrak{C}\ g\ f)$

Elementary properties.

lemma (**in** *category*) *cat-is-inverse-is-inverse-the-inverse*:
assumes *is-inverse* $\mathfrak{C}$ g f
shows *is-inverse* $\mathfrak{C}$ $(f^{-1}{}_{C\mathfrak{C}})$ f
⟨*proof*⟩

lemma (**in** *category*) *cat-is-inverse-eq-the-inverse*:
assumes *is-inverse* $\mathfrak{C}$ g f
shows $g = f^{-1}{}_{C\mathfrak{C}}$
⟨*proof*⟩

The inverse arrow is an inverse of an isomorphism.

lemma (**in** *category*) *cat-the-inverse-is-inverse*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows *is-inverse* $\mathfrak{C}$ $(f^{-1}{}_{C\mathfrak{C}})$ f
⟨*proof*⟩

lemma (**in** *category*) *cat-the-inverse-is-iso-arr*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows $f^{-1}{}_{C\mathfrak{C}} : b \mapsto_{iso\mathfrak{C}} a$
⟨*proof*⟩

lemma (in *category*) *cat-the-inverse-is-iso-arr'*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows ${f^{-1}}_{C\mathfrak{C}} : b \mapsto_{iso\mathfrak{C}'} a$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-the-inverse-is-iso-arr'*

lemma (in *category*) *op-cat-the-inverse*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows ${f^{-1}}_{C\,op\text{-}cat\ \mathfrak{C}} = {f^{-1}}_{C\mathfrak{C}}$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-the-inverse*

lemma (in *category*) *cat-Comp-the-inverse*:
assumes $g : b \mapsto_{iso\mathfrak{C}} c$ **and** $f : a \mapsto_{iso\mathfrak{C}} b$
shows ${(g \circ_{A\mathfrak{C}} f)^{-1}}_{C\mathfrak{C}} = {f^{-1}}_{C\mathfrak{C}} \circ_{A\mathfrak{C}} {g^{-1}}_{C\mathfrak{C}}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-Comp-the-inverse*

lemma (in *category*) *cat-the-inverse-Comp-CId*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows *cat-the-inverse-Comp-CId-left*: ${f^{-1}}_{C\mathfrak{C}} \circ_{A\mathfrak{C}} f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
and *cat-the-inverse-Comp-CId-right*: $f \circ_{A\mathfrak{C}} {f^{-1}}_{C\mathfrak{C}} = \mathfrak{C}(\!|CId|\!)(\!|b|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-the-inverse-Comp-CId*

lemma (in *category*) *cat-the-inverse-the-inverse*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows ${({f^{-1}}_{C\mathfrak{C}})^{-1}}_{C\mathfrak{C}} = f$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-the-inverse-the-inverse*

2.9 Isomorphic objects

See Chapter I-5 in [7].

definition *obj-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *obj-iso* $\mathfrak{C}\ a\ b \longleftrightarrow (\exists f.\ f : a \mapsto_{iso\mathfrak{C}} b)$

syntax *-obj-iso* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$ $(\langle(\text{-}/ \approx_{obj^1} \text{-})\rangle\ [55, 56]\ 55)$
syntax-consts *-obj-iso* ⇌ *obj-iso*
translations $a \approx_{obj\mathfrak{C}} b \rightleftharpoons CONST\ obj\text{-}iso\ \mathfrak{C}\ a\ b$

Rules.

lemma *obj-isoI*:
assumes $f : a \mapsto_{iso\mathfrak{C}} b$
shows $a \approx_{obj\mathfrak{C}} b$
⟨*proof*⟩

lemma *obj-isoD*[*dest*]:
assumes $a \approx_{obj\mathfrak{C}} b$
shows $\exists f.\ f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

lemma *obj-isoE*[*elim*!]:
assumes $a \approx_{obj\mathfrak{C}} b$
obtains f **where** $f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

Elementary properties.

lemma (**in** *category*) *op-cat-obj-iso*[*cat-op-simps*]:
$a \approx_{obj op\text{-}cat\ \mathfrak{C}} b = b \approx_{obj\mathfrak{C}} a$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.op-cat-obj-iso*

lemmas [*cat-op-intros*] = *category.op-cat-obj-iso*[*THEN iffD2*]

Equivalence relation.

lemma (**in** *category*) *cat-obj-iso-refl*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $a \approx_{obj\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-iso-sym*[*sym*]:
assumes $a \approx_{obj\mathfrak{C}} b$
shows $b \approx_{obj\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-iso-trans*[*trans*]:
assumes $a \approx_{obj\mathfrak{C}} b$ **and** $b \approx_{obj\mathfrak{C}} c$
shows $a \approx_{obj\mathfrak{C}} c$
⟨*proof*⟩

2.10 Terminal object and initial object

lemma (**in** *category*) *cat-obj-terminal-CId*:
— See Chapter I-5 in [7].
assumes *obj-terminal* $\mathfrak{C}$ a **and** $f : a \mapsto_{\mathfrak{C}} a$
shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) = f$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-initial-CId*:
— See Chapter I-5 in [7].
assumes *obj-initial* $\mathfrak{C}$ a **and** $f : a \mapsto_{\mathfrak{C}} a$
shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) = f$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-terminal-obj-iso*:
— See Chapter I-5 in [7].
assumes *obj-terminal* $\mathfrak{C}$ a **and** *obj-terminal* $\mathfrak{C}$ a'
shows $a \approx_{obj\mathfrak{C}} a'$
⟨*proof*⟩

lemma (**in** *category*) *cat-obj-initial-obj-iso*:
— See Chapter I-5 in [7].
assumes *obj-initial* $\mathfrak{C}$ a **and** *obj-initial* $\mathfrak{C}$ a'
shows $a' \approx_{obj\mathfrak{C}} a$
⟨*proof*⟩

2.11 Null object

lemma (in *category*) *cat-obj-null-obj-iso*:
— See Chapter I-5 in [7].
assumes *obj-null* $\mathfrak{C}$ z **and** *obj-null* $\mathfrak{C}$ z'
shows $z \approx_{obj\mathfrak{C}} z'$
⟨*proof*⟩

2.12 Groupoid

See Chapter I-5 in [7].

locale *groupoid* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
assumes *grpd-is-iso-arr*: $f : a \mapsto_{\mathfrak{C}} b \Longrightarrow f : a \mapsto_{iso\mathfrak{C}} b$

Rules.

mk-ide rf *groupoid-def*[*unfolded groupoid-axioms-def*]
|*intro groupoidI*|
|*dest groupoidD*[*dest*]|
|*elim groupoidE*[*elim*]|

3 Smallness for categories

3.1 Background

An explanation of the methodology chosen for the exposition of all matters related to the size of the categories and associated entities is given in [8].

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

3.2 Tiny category

3.2.1 Definition and elementary properties

locale *tiny-category* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *CId*: *vsv* ‹$\mathfrak{C}(\!|CId|\!)$› **for** α $\mathfrak{C}$ +
  assumes *tiny-cat-length*[*cat-cs-simps*]: *vcard* $\mathfrak{C}$ = $6_{\mathbb{N}}$
    and *tiny-cat-tiny-semicategory*[*slicing-intros*]:
      tiny-semicategory α (*cat-smc* $\mathfrak{C}$)
    and *tiny-cat-CId-vdomain*[*cat-cs-simps*]: $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|CId|\!)$) = $\mathfrak{C}(\!|Obj|\!)$
    and *tiny-cat-CId-is-arr*[*cat-cs-intros*]:
      $a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
    and *tiny-cat-CId-left-left*[*cat-cs-simps*]:
      $f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
    and *tiny-cat-CId-right-left*[*cat-cs-simps*]:
      $f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$

lemmas [*slicing-intros*] = *tiny-category.tiny-cat-tiny-semicategory*

Rules.

lemma (**in** *tiny-category*) *tiny-category-axioms'*[*cat-small-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *tiny-category* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *tiny-category-def*[*unfolded tiny-category-axioms-def*]
  |*intro tiny-categoryI*|
  |*dest tiny-categoryD*[*dest*]|
  |*elim tiny-categoryE*[*elim*]|

lemma *tiny-categoryI'*:
  assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{C}(\!|Obj|\!) \in_{\circ}$ *Vset* α **and** $\mathfrak{C}(\!|Arr|\!) \in_{\circ}$ *Vset* α
  shows *tiny-category* α $\mathfrak{C}$
⟨*proof*⟩

lemma *tiny-categoryI''*:
  assumes $\mathcal{Z}$ α
    and *vfsequence* $\mathfrak{C}$
    and *vcard* $\mathfrak{C}$ = $6_{\mathbb{N}}$
    and *vsv* ($\mathfrak{C}(\!|Dom|\!)$)
    and *vsv* ($\mathfrak{C}(\!|Cod|\!)$)
    and *vsv* ($\mathfrak{C}(\!|Comp|\!)$)
    and *vsv* ($\mathfrak{C}(\!|CId|\!)$)
    and $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|Dom|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
    and $\mathcal{R}_{\circ}$ ($\mathfrak{C}(\!|Dom|\!)$) $\subseteq_{\circ}$ $\mathfrak{C}(\!|Obj|\!)$
    and $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|Cod|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
    and $\mathcal{R}_{\circ}$ ($\mathfrak{C}(\!|Cod|\!)$) $\subseteq_{\circ}$ $\mathfrak{C}(\!|Obj|\!)$
    and $\bigwedge gf.\ gf \in_{\circ} \mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|Comp|\!)$) $\longleftrightarrow$
      ($\exists g\ f\ b\ c\ a.\ gf = [g, f]_{\circ} \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b$)
    and $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|CId|\!)$) = $\mathfrak{C}(\!|Obj|\!)$

and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
and $\bigwedge c\ d\ h\ b\ g\ a\ f.\ [\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$
and $\bigwedge a.\ a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
and $\bigwedge b\ c\ f.\ f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$
and $\mathfrak{C}(\!|Obj|\!) \in_{\circ} Vset\ \alpha$
and $\mathfrak{C}(\!|Arr|\!) \in_{\circ} Vset\ \alpha$
shows *tiny-category* $\alpha\ \mathfrak{C}$
⟨*proof*⟩

Slicing.

context *tiny-category*
begin

interpretation *smc*: *tiny-semicategory* α ‹*cat-smc* $\mathfrak{C}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
tiny-cat-semicategory = *smc.semicategory-axioms*
and *tiny-cat-Obj-in-Vset*[*cat-small-cs-intros*] = *smc.tiny-smc-Obj-in-Vset*
and *tiny-cat-Arr-in-Vset*[*cat-small-cs-intros*] = *smc.tiny-smc-Arr-in-Vset*
and *tiny-cat-Dom-in-Vset*[*cat-small-cs-intros*] = *smc.tiny-smc-Dom-in-Vset*
and *tiny-cat-Cod-in-Vset*[*cat-small-cs-intros*] = *smc.tiny-smc-Cod-in-Vset*
and *tiny-cat-Comp-in-Vset*[*cat-small-cs-intros*] = *smc.tiny-smc-Comp-in-Vset*

end

Elementary properties.

sublocale *tiny-category* ⊆ *category*
⟨*proof*⟩

lemmas (**in** *tiny-category*) *tiny-cat-category* = *category-axioms*

lemmas [*cat-small-cs-intros*] = *tiny-category.tiny-cat-category*

Size.

lemma (**in** *tiny-category*) *tiny-cat-CId-in-Vset*: $\mathfrak{C}(\!|CId|\!) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *tiny-category*) *tiny-cat-in-Vset*: $\mathfrak{C} \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma *tiny-category*[*simp*]: *small* $\{\mathfrak{C}.\ tiny\text{-}category\ \alpha\ \mathfrak{C}\}$
⟨*proof*⟩

lemma *small-categories-vsubset-Vset*: *set* $\{\mathfrak{C}.\ tiny\text{-}category\ \alpha\ \mathfrak{C}\} \subseteq_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-tiny-category-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows *tiny-category* $\beta\ \mathfrak{C}$
⟨*proof*⟩

3.2.2 Opposite tiny category

lemma (**in** *tiny-category*) *tiny-category-op*: *tiny-category* α (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas *tiny-category-op*[*cat-op-intros*] = *tiny-category.tiny-category-op*

3.3 Finite category

3.3.1 Definition and elementary properties

A definition of a finite category can be found in nLab [1][1].

locale *finite-category* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *CId*: *vsv* ‹$\mathfrak{C}(\!|CId|\!)$› **for** α $\mathfrak{C}$ +
  assumes *fin-cat-length*[*cat-cs-simps*]: *vcard* $\mathfrak{C}$ = $6_{\mathbb{N}}$
    and *fin-cat-finite-semicategory*[*slicing-intros*]:
      finite-semicategory α (*cat-smc* $\mathfrak{C}$)
    and *fin-cat-CId-vdomain*[*cat-cs-simps*]: $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|CId|\!)$) = $\mathfrak{C}(\!|Obj|\!)$
    and *fin-cat-CId-is-arr*[*cat-cs-intros*]:
      $a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{C}} a$
    and *fin-cat-CId-left-left*[*cat-cs-simps*]:
      $f : a \mapsto_{\mathfrak{C}} b \Longrightarrow \mathfrak{C}(\!|CId|\!)(\!|b|\!) \circ_{A\mathfrak{C}} f = f$
    and *fin-cat-CId-right-left*[*cat-cs-simps*]:
      $f : b \mapsto_{\mathfrak{C}} c \Longrightarrow f \circ_{A\mathfrak{C}} \mathfrak{C}(\!|CId|\!)(\!|b|\!) = f$

lemmas [*slicing-intros*] = *finite-category.fin-cat-finite-semicategory*

Rules.

lemma (**in** *finite-category*) *fin-category-axioms'*[*cat-small-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *finite-category* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *finite-category-def*[*unfolded finite-category-axioms-def*]
  |*intro finite-categoryI*|
  |*dest finite-categoryD*[*dest*]|
  |*elim finite-categoryE*[*elim*]|

lemma *finite-categoryI'*:
  assumes *category* α $\mathfrak{C}$ **and** *vfinite* ($\mathfrak{C}(\!|Obj|\!)$) **and** *vfinite* ($\mathfrak{C}(\!|Arr|\!)$)
  shows *finite-category* α $\mathfrak{C}$
⟨*proof*⟩

lemma *finite-categoryI''*:
  assumes *tiny-category* α $\mathfrak{C}$ **and** *vfinite* ($\mathfrak{C}(\!|Obj|\!)$) **and** *vfinite* ($\mathfrak{C}(\!|Arr|\!)$)
  shows *finite-category* α $\mathfrak{C}$
  ⟨*proof*⟩

Slicing.

context *finite-category*
begin

interpretation *smc*: *finite-semicategory* α ‹*cat-smc* $\mathfrak{C}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
  fin-cat-tiny-semicategory = *smc.tiny-semicategory-axioms*
  and *fin-smc-Obj-vfinite*[*cat-small-cs-intros*] = *smc.fin-smc-Obj-vfinite*
  and *fin-smc-Arr-vfinite*[*cat-small-cs-intros*] = *smc.fin-smc-Arr-vfinite*

end

[1] https://ncatlab.org/nlab/show/finite+category

Elementary properties.

sublocale *finite-category* ⊆ *tiny-category*
⟨*proof*⟩

lemmas (**in** *finite-category*) *fin-cat-tiny-category* = *tiny-category-axioms*

lemmas [*cat-small-cs-intros*] = *finite-category.fin-cat-tiny-category*

lemma (**in** *finite-category*) *fin-cat-in-Vset*: $\mathfrak{C}$ ∈$_\circ$ *Vset* α
⟨*proof*⟩

Size.

lemma *small-finite-categories*[*simp*]: *small* {$\mathfrak{C}$. *finite-category* α $\mathfrak{C}$}
⟨*proof*⟩

lemma *small-finite-categories-vsubset-Vset*:
set {$\mathfrak{C}$. *finite-category* α $\mathfrak{C}$} ⊆$_\circ$ *Vset* α
⟨*proof*⟩

3.3.2 Opposite finite category

lemma (**in** *finite-category*) *finite-category-op*: *finite-category* α (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas *finite-category-op*[*cat-op-intros*] = *finite-category.finite-category-op*

4 Functor

4.1 Background

named-theorems *cf-cs-simps*
named-theorems *cf-cs-intros*

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

lemmas [*cat-cs-simps*] = *dg-shared-cs-simps*
lemmas [*cat-cs-intros*] = *dg-shared-cs-intros*

4.1.1 Slicing

definition *cf-smcf* :: $V \Rightarrow V$
  where *cf-smcf* $\mathfrak{C}$ =
    $[\mathfrak{C}(\!|ObjMap|\!),\ \mathfrak{C}(\!|ArrMap|\!),\ cat\text{-}smc\ (\mathfrak{C}(\!|HomDom|\!)),\ cat\text{-}smc\ (\mathfrak{C}(\!|HomCod|\!))]_\circ$

Components.

lemma *cf-smcf-components*:
  shows [*slicing-simps*]: *cf-smcf* $\mathfrak{F}(\!|ObjMap|\!)$ = $\mathfrak{F}(\!|ObjMap|\!)$
    and [*slicing-simps*]: *cf-smcf* $\mathfrak{F}(\!|ArrMap|\!)$ = $\mathfrak{F}(\!|ArrMap|\!)$
    and [*slicing-commute*]: *cf-smcf* $\mathfrak{F}(\!|HomDom|\!)$ = *cat-smc* $(\mathfrak{F}(\!|HomDom|\!))$
    and [*slicing-commute*]: *cf-smcf* $\mathfrak{F}(\!|HomCod|\!)$ = *cat-smc* $(\mathfrak{F}(\!|HomCod|\!))$
  ⟨*proof*⟩

4.2 Definition and elementary properties

See Chapter I-3 in [7].

locale *is-functor* =
  $\mathcal{Z}$ α + *vfsequence* $\mathfrak{F}$ + *HomDom*: *category* α $\mathfrak{A}$ + *HomCod*: *category* α $\mathfrak{B}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *cf-length*[*cat-cs-simps*]: *vcard* $\mathfrak{F}$ = $\mathfrak{4}_{\mathbb{N}}$
    and *cf-is-semifunctor*[*slicing-intros*]:
      cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ *cat-smc* $\mathfrak{B}$
    and *cf-HomDom*[*cat-cs-simps*]: $\mathfrak{F}(\!|HomDom|\!)$ = $\mathfrak{A}$
    and *cf-HomCod*[*cat-cs-simps*]: $\mathfrak{F}(\!|HomCod|\!)$ = $\mathfrak{B}$
    and *cf-ObjMap-CId*[*cat-cs-intros*]:
      $c \in_\circ \mathfrak{A}(\!|Obj|\!) \Longrightarrow \mathfrak{F}(\!|ArrMap|\!)(\!|\mathfrak{A}(\!|CId|\!)(\!|c|\!)|\!) = \mathfrak{B}(\!|CId|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|c|\!)|\!)$

syntax *-is-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-} \mapsto\!\!\mapsto_{C1} \text{-})\rangle\ [51,\ 51,\ 51]\ 51)$
syntax-consts *-is-functor* ⇌ *is-functor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$ ⇌ *CONST is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation (*input*) *is-cn-cf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  where *is-cn-cf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ ≡ $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$

syntax *-is-cn-cf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-} \ {}_{C}\!\mapsto\!\!\mapsto_{1} \text{-})\rangle\ [51,\ 51,\ 51]\ 51)$
syntax-consts *-is-cn-cf* ⇌ *is-cn-cf*
translations $\mathfrak{F} : \mathfrak{A}\ {}_{C}\!\mapsto\!\!\mapsto_{\alpha} \mathfrak{B}$ ↽ *CONST is-cn-cf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation *all-cfs* :: $V \Rightarrow V$
  where *all-cfs* α ≡ *set* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}\}$

abbreviation *cfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cfs* α $\mathfrak{A}$ $\mathfrak{B}$ ≡ *set* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$

lemmas [*cat-cs-simps*] =
is-functor.cf-length
is-functor.cf-HomDom
is-functor.cf-HomCod
is-functor.cf-ObjMap-CId

lemma *cn-cf-ObjMap-CId*[*cat-cn-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$ **and** $c \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|\mathfrak{A}(\!|CId|\!)(\!|c|\!)|\!) = \mathfrak{B}(\!|CId|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|c|\!)|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-is-semifunctor'*:
assumes $\mathfrak{A}'$ = *cat-smc* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-functor.cf-is-semifunctor'*

lemma *cn-smcf-comp-is-semifunctor*:
assumes $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows *cf-smcf* $\mathfrak{F}$: *cat-smc* $\mathfrak{A}$ ${}_{SMC}\mapsto\mapsto_{\alpha}$*cat-smc* $\mathfrak{B}$
⟨*proof*⟩

lemma *cn-smcf-comp-is-semifunctor'*[*slicing-intros*]:
assumes $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
and $\mathfrak{A}'$ = *op-smc* (*cat-smc* $\mathfrak{A}$)
and $\mathfrak{B}'$ = *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
⟨*proof*⟩

Rules.

lemma (**in** *is-functor*) *is-functor-axioms'*[*cat-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-functor-def*[*unfolded is-functor-axioms-def*]
|*intro is-functorI*|
|*dest is-functorD*[*dest*]|
|*elim is-functorE*[*elim*]|

lemmas [*cat-cs-intros*] = *is-functorD*(*3*,*4*)

lemma *is-functorI'*:
assumes $\mathcal{Z}$ α
and *vfsequence* $\mathfrak{F}$
and *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and *vcard* $\mathfrak{F} = 4_{\mathbb{N}}$
and $\mathfrak{F}(\!|HomDom|\!) = \mathfrak{A}$
and $\mathfrak{F}(\!|HomCod|\!) = \mathfrak{B}$
and *vsv* ($\mathfrak{F}(\!|ObjMap|\!)$)
and *vsv* ($\mathfrak{F}(\!|ArrMap|\!)$)
and $\mathcal{D}_{\circ}$ ($\mathfrak{F}(\!|ObjMap|\!)$) = $\mathfrak{A}(\!|Obj|\!)$
and $\mathcal{R}_{\circ}$ ($\mathfrak{F}(\!|ObjMap|\!)$) $\subseteq_{\circ}$ $\mathfrak{B}(\!|Obj|\!)$
and $\mathcal{D}_{\circ}$ ($\mathfrak{F}(\!|ArrMap|\!)$) = $\mathfrak{A}(\!|Arr|\!)$

and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
and $(\bigwedge c.\ c \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{A}(\lfloor CId \rfloor)(\lfloor c \rfloor) \rfloor) = \mathfrak{B}(\lfloor CId \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor) \rfloor))$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-functorD′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{F}$
and *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and *vcard* $\mathfrak{F} = \mathfrak{4}_{\mathbb{N}}$
and $\mathfrak{F}(\lfloor HomDom \rfloor) = \mathfrak{A}$
and $\mathfrak{F}(\lfloor HomCod \rfloor) = \mathfrak{B}$
and *vsv* $(\mathfrak{F}(\lfloor ObjMap \rfloor))$
and *vsv* $(\mathfrak{F}(\lfloor ArrMap \rfloor))$
and $\mathcal{D}_{\circ}(\mathfrak{F}(\lfloor ObjMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
and $\mathcal{R}_{\circ}(\mathfrak{F}(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{B}(\lfloor Obj \rfloor)$
and $\mathcal{D}_{\circ}(\mathfrak{F}(\lfloor ArrMap \rfloor)) = \mathfrak{A}(\lfloor Arr \rfloor)$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
and $(\bigwedge c.\ c \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{A}(\lfloor CId \rfloor)(\lfloor c \rfloor) \rfloor) = \mathfrak{B}(\lfloor CId \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor) \rfloor))$
⟨*proof*⟩

lemma *is-functorE′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
obtains $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{F}$
and *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and *vcard* $\mathfrak{F} = \mathfrak{4}_{\mathbb{N}}$
and $\mathfrak{F}(\lfloor HomDom \rfloor) = \mathfrak{A}$
and $\mathfrak{F}(\lfloor HomCod \rfloor) = \mathfrak{B}$
and *vsv* $(\mathfrak{F}(\lfloor ObjMap \rfloor))$
and *vsv* $(\mathfrak{F}(\lfloor ArrMap \rfloor))$
and $\mathcal{D}_{\circ}(\mathfrak{F}(\lfloor ObjMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
and $\mathcal{R}_{\circ}(\mathfrak{F}(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{B}(\lfloor Obj \rfloor)$
and $\mathcal{D}_{\circ}(\mathfrak{F}(\lfloor ArrMap \rfloor)) = \mathfrak{A}(\lfloor Arr \rfloor)$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
and $(\bigwedge c.\ c \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{A}(\lfloor CId \rfloor)(\lfloor c \rfloor) \rfloor) = \mathfrak{B}(\lfloor CId \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor) \rfloor))$
⟨*proof*⟩

A functor is a semifunctor.

context *is-functor*
begin

interpretation *smcf*: *is-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
⟨*proof*⟩

sublocale *ObjMap*: *vsv* ‹$\mathfrak{F}(\lfloor ObjMap \rfloor)$›

⟨*proof*⟩
sublocale *ArrMap*: *vsv* ‹$\mathfrak{F}(\!|ArrMap|\!)$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
cf-ObjMap-vsv = *smcf.smcf-ObjMap-vsv*
and *cf-ArrMap-vsv* = *smcf.smcf-ArrMap-vsv*
and *cf-ObjMap-vdomain*[*cat-cs-simps*] = *smcf.smcf-ObjMap-vdomain*
and *cf-ObjMap-vrange* = *smcf.smcf-ObjMap-vrange*
and *cf-ArrMap-vdomain*[*cat-cs-simps*] = *smcf.smcf-ArrMap-vdomain*
and *cf-ArrMap-is-arr* = *smcf.smcf-ArrMap-is-arr*
and *cf-ArrMap-is-arr''*[*cat-cs-intros*] = *smcf.smcf-ArrMap-is-arr''*
and *cf-ArrMap-is-arr'*[*cat-cs-intros*] = *smcf.smcf-ArrMap-is-arr'*
and *cf-ObjMap-app-in-HomCod-Obj*[*cat-cs-intros*] =
smcf.smcf-ObjMap-app-in-HomCod-Obj
and *cf-ArrMap-vrange* = *smcf.smcf-ArrMap-vrange*
and *cf-ArrMap-app-in-HomCod-Arr*[*cat-cs-intros*] =
smcf.smcf-ArrMap-app-in-HomCod-Arr
and *cf-ObjMap-vsubset-Vset* = *smcf.smcf-ObjMap-vsubset-Vset*
and *cf-ArrMap-vsubset-Vset* = *smcf.smcf-ArrMap-vsubset-Vset*
and *cf-ObjMap-in-Vset* = *smcf.smcf-ObjMap-in-Vset*
and *cf-ArrMap-in-Vset* = *smcf.smcf-ArrMap-in-Vset*
and *cf-is-semifunctor-if-ge-Limit* = *smcf.smcf-is-semifunctor-if-ge-Limit*
and *cf-is-arr-HomCod* = *smcf.smcf-is-arr-HomCod*
and *cf-vimage-dghm-ArrMap-vsubset-Hom* =
smcf.smcf-vimage-dghm-ArrMap-vsubset-Hom

lemmas-with [*unfolded slicing-simps*]:
cf-ArrMap-Comp = *smcf.smcf-ArrMap-Comp*

end

lemmas [*cat-cs-simps*] =
is-functor.cf-ObjMap-vdomain
is-functor.cf-ArrMap-vdomain
is-functor.cf-ArrMap-Comp

lemmas [*cat-cs-intros*] =
is-functor.cf-ObjMap-app-in-HomCod-Obj
is-functor.cf-ArrMap-app-in-HomCod-Arr
is-functor.cf-ArrMap-is-arr'

Elementary properties.

lemma *cn-cf-ArrMap-Comp*[*cat-cn-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
and $g : c \mapsto_{\mathfrak{A}} b$
and $f : b \mapsto_{\mathfrak{A}} a$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|f \circ_{A\mathfrak{A}} g|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)$
⟨*proof*⟩

lemma *cf-eqI*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G}(\!|ObjMap|\!) = \mathfrak{F}(\!|ObjMap|\!)$
and $\mathfrak{G}(\!|ArrMap|\!) = \mathfrak{F}(\!|ArrMap|\!)$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$

shows $\mathfrak{G} = \mathfrak{F}$
⟨*proof*⟩

lemma *cf-smcf-eqI*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$
and *cf-smcf* $\mathfrak{G}$ = *cf-smcf* $\mathfrak{F}$
shows $\mathfrak{G} = \mathfrak{F}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-def*: $\mathfrak{F} = [\mathfrak{F}(\!|ObjMap|\!), \mathfrak{F}(\!|ArrMap|\!), \mathfrak{F}(\!|HomDom|\!), \mathfrak{F}(\!|HomCod|\!)]_\circ$
⟨*proof*⟩

Size.

lemma (**in** *is-functor*) *cf-in-Vset*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows $\mathfrak{F} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-is-functor-if-ge-Limit*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\beta} \mathfrak{B}$
⟨*proof*⟩

lemma *small-all-cfs*[*simp*]: *small* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-in-Vset-*γ: $\mathfrak{F} \in_\circ Vset\ (\alpha + \gamma_{\mathbf{N}})$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *all-cfs-in-Vset*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *all-cfs* $\alpha \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma *small-cfs*[*simp*]: *small* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

4.2.1 Further properties

lemma (**in** *is-functor*) *cf-ArrMap-is-iso-arr*:
assumes $f : a \mapsto_{iso\mathfrak{A}} b$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{iso\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ArrMap-is-iso-arr'*[*cat-arrow-cs-intros*]:
assumes $f : a \mapsto_{iso\mathfrak{A}} b$ **and** $\mathfrak{F}a = \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$ **and** $\mathfrak{F}b = \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) : \mathfrak{F}a \mapsto_{iso\mathfrak{B}} \mathfrak{F}b$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *is-functor.cf-ArrMap-is-iso-arr'*

4.3 Opposite functor

4.3.1 Definition and elementary properties

See Chapter II-2 in [7].

definition *op-cf* :: $V \Rightarrow V$
  where *op-cf* $\mathfrak{F}$ =
    $[\mathfrak{F}(\!|ObjMap|\!), \mathfrak{F}(\!|ArrMap|\!), op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!)), op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))]_\circ$

Components.

lemma *op-cf-components*[*cat-op-simps*]:
  shows *op-cf* $\mathfrak{F}(\!|ObjMap|\!) = \mathfrak{F}(\!|ObjMap|\!)$
    and *op-cf* $\mathfrak{F}(\!|ArrMap|\!) = \mathfrak{F}(\!|ArrMap|\!)$
    and *op-cf* $\mathfrak{F}(\!|HomDom|\!) = op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))$
    and *op-cf* $\mathfrak{F}(\!|HomCod|\!) = op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))$
  ⟨*proof*⟩

Slicing.

lemma *cf-smcf-op-cf*[*slicing-commute*]: *op-smcf* (*cf-smcf* $\mathfrak{F}$) = *cf-smcf* (*op-cf* $\mathfrak{F}$)
⟨*proof*⟩

Elementary properties.

lemma *op-cf-vsv*[*cat-op-intros*]: *vsv* (*op-cf* $\mathfrak{F}$) ⟨*proof*⟩

4.3.2 Further properties

lemma (**in** *is-functor*) *is-functor-op*: *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\!\!\mapsto_{C\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-functor*) *is-functor-op'*[*cat-op-intros*]:
  assumes $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}' =$ *op-cat* $\mathfrak{B}$
  shows *op-cf* $\mathfrak{F} : \mathfrak{A}' \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}'$
  ⟨*proof*⟩

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

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

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

lemma *eq-op-cf-iff*[*cat-op-simps*]:
  assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{F} : \mathfrak{C} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{D}$
  shows *op-cf* $\mathfrak{G}$ = *op-cf* $\mathfrak{F} \longleftrightarrow \mathfrak{G} = \mathfrak{F}$
⟨*proof*⟩

4.4 Composition of covariant functors

4.4.1 Definition and elementary properties

abbreviation (*input*) *cf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\circ_{CF}\rangle$ 55)
  where *cf-comp* ≡ *dghm-comp*

Slicing.

lemma *cf-smcf-smcf-comp*[*slicing-commute*]:
  cf-smcf $\mathfrak{G} \circ_{SMCF}$ *cf-smcf* $\mathfrak{F}$ = *cf-smcf* ($\mathfrak{G} \circ_{CF} \mathfrak{F}$)
  ⟨*proof*⟩

4.4.2 Object map

lemma *cf-comp-ObjMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *vsv* $((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-comp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-comp-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ObjMap|\!)) \subseteq_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-comp-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** [*simp*]: $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $(\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

4.4.3 Arrow map

lemma *cf-comp-ArrMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *vsv* $((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *cf-comp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ((\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ArrMap|\!)) \subseteq_\circ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-comp-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** [*simp*]: $f \in_\circ \mathfrak{A}(\!|Arr|\!)$
shows $(\mathfrak{G} \circ_{CF} \mathfrak{F})(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{G}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)|\!)$
⟨*proof*⟩

4.4.4 Further properties

lemma *cf-comp-is-functorI*[*cat-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-comp-assoc*[*cat-cs-simps*]:
assumes $\mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $(\mathfrak{H} \circ_{CF} \mathfrak{G}) \circ_{CF} \mathfrak{F} = \mathfrak{H} \circ_{CF} (\mathfrak{G} \circ_{CF} \mathfrak{F})$
⟨*proof*⟩

The opposite of the covariant composition of functors.

lemma *op-cf-cf-comp*[*cat-op-simps*]: *op-cf* $(\mathfrak{G} \circ_{CF} \mathfrak{F})$ = *op-cf* $\mathfrak{G} \circ_{CF}$ *op-cf* $\mathfrak{F}$
⟨*proof*⟩

Composition helper.

lemma *cf-comp-assoc-helper*:
  assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $\mathfrak{H} \circ_{CF} \mathfrak{G} = \mathcal{Q}$
  shows $\mathfrak{H} \circ_{CF} (\mathfrak{G} \circ_{CF} \mathfrak{F}) = \mathcal{Q} \circ_{CF} \mathfrak{F}$
⟨*proof*⟩

4.5 Composition of contravariant functors

4.5.1 Definition and elementary properties

See section 1.2 in [3].

definition *cf-cn-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$_{CF}\circ$› *55*)
  where $\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F} =$
    [
      $\mathfrak{G}(\!|ObjMap|\!) \circ_{\circ} \mathfrak{F}(\!|ObjMap|\!)$,
      $\mathfrak{G}(\!|ArrMap|\!) \circ_{\circ} \mathfrak{F}(\!|ArrMap|\!)$,
      op-cat $(\mathfrak{F}(\!|HomDom|\!))$,
      $\mathfrak{G}(\!|HomCod|\!)$
    ]$_{\circ}$

Components.

lemma *cf-cn-comp-components*:
  shows $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!) = \mathfrak{G}(\!|ObjMap|\!) \circ_{\circ} \mathfrak{F}(\!|ObjMap|\!)$
    and $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!) = \mathfrak{G}(\!|ArrMap|\!) \circ_{\circ} \mathfrak{F}(\!|ArrMap|\!)$
    and [*cat-cn-cs-simps*]: $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|HomDom|\!) = op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))$
    and [*cat-cn-cs-simps*]: $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|HomCod|\!) = \mathfrak{G}(\!|HomCod|\!)$
  ⟨*proof*⟩

Slicing.

lemma *cf-smcf-cf-cn-comp*[*slicing-commute*]:
  cf-smcf $\mathfrak{G}\ {}_{SMCF}\circ$ *cf-smcf* $\mathfrak{F}$ = *cf-smcf* $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})$
  ⟨*proof*⟩

4.5.2 Object map: two contravariant functors

lemma *cf-cn-comp-ObjMap-vsv*[*cat-cn-cs-intros*]:
  assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
  shows *vsv* $((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-cn-comp-ObjMap-vdomain*[*cat-cn-cs-simps*]:
  assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
  shows $\mathcal{D}_{\circ} ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-comp-ObjMap-vrange*:
  assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
  shows $\mathcal{R}_{\circ} ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-comp-ObjMap-app*[*cat-cn-cs-simps*]:
  assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
  shows $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

4.5.3 Arrow map: two contravariant functors

lemma *cf-cn-comp-ArrMap-vsv*[*cat-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *cf-cn-comp-ArrMap-vdomain*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-comp-ArrMap-app*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\!|Arr|\!)$
shows $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)(\!|a|\!) = \mathfrak{G}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

4.5.4 Object map: contravariant and covariant functor

lemma *cf-cn-cov-comp-ObjMap-vsv*[*cat-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-cn-cov-comp-ObjMap-vdomain*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-comp-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-comp-ObjMap-app*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
shows $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{G}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

4.5.5 Arrow map: contravariant and covariant functors

lemma *cf-cn-cov-comp-ArrMap-vsv*[*cat-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *cf-cn-cov-comp-ArrMap-vdomain*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$

⟨*proof*⟩

lemma *cf-cn-cov-comp-ArrMap-app*[*cat-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\!|Arr|\!)$
shows $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})(\!|ArrMap|\!)(\!|a|\!) = \mathfrak{G}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

4.5.6 Further properties

lemma *cf-cn-comp-is-functorI*[*cat-cn-cs-intros*]:
assumes *category* α $\mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

See section 1.2 in [3]).

lemma *cf-cn-cov-comp-is-functor*[*cat-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$
⟨*proof*⟩

See section 1.2 in [3].

lemma *cf-cov-cn-comp-is-functor*[*cat-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A}\ {}_{C}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$
⟨*proof*⟩

The opposite of the contravariant composition of functors.

lemma *op-cf-cf-cn-comp*[*cat-op-simps*]: *op-cf* $(\mathfrak{G}\ {}_{CF}\circ\ \mathfrak{F})$ = *op-cf* $\mathfrak{G}\ {}_{CF}\circ$ *op-cf* $\mathfrak{F}$
⟨*proof*⟩

4.6 Identity functor

4.6.1 Definition and elementary properties

See Chapter I-3 in [7].

abbreviation (*input*) *cf-id* :: $V \Rightarrow V$ **where** *cf-id* ≡ *dghm-id*

Slicing.

lemma *cf-smcf-cf-id*[*slicing-commute*]: *smcf-id* (*cat-smc* $\mathfrak{C}$) = *cf-smcf* (*cf-id* $\mathfrak{C}$)
⟨*proof*⟩

context *category*
begin

interpretation *smc*: *semicategory* α ‹*cat-smc* $\mathfrak{C}$› ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
cat-smcf-id-is-semifunctor = *smc.smc-smcf-id-is-semifunctor*

end

4.6.2 Object map

lemmas [*cat-cs-simps*] = *dghm-id-ObjMap-app*

4.6.3 Arrow map

lemmas [*cat-cs-simps*] = *dghm-id-ArrMap-app*

4.6.4 Opposite of an identity functor.

lemma *op-cf-cf-id*[*cat-op-simps*]: *op-cf* (*cf-id* $\mathfrak{C}$) = *cf-id* (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

4.6.5 An identity functor is a functor

lemma (**in** *category*) *cat-cf-id-is-functor*: *cf-id* $\mathfrak{C} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-id-is-functor'*:
assumes $\mathfrak{A} = \mathfrak{C}$ **and** $\mathfrak{B} = \mathfrak{C}$
shows *cf-id* $\mathfrak{C} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-id-is-functor'*

4.6.6 Further properties

lemma (**in** *is-functor*) *cf-cf-comp-cf-id-left*[*cat-cs-simps*]: *cf-id* $\mathfrak{B}$ $\circ_{CF}$ $\mathfrak{F}$ = $\mathfrak{F}$
— See Chapter I-3 in [7]).
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-cf-comp-cf-id-left*

lemma (**in** *is-functor*) *cf-cf-comp-cf-id-right*[*cat-cs-simps*]: $\mathfrak{F}$ $\circ_{CF}$ *cf-id* $\mathfrak{A}$ = $\mathfrak{F}$
— See Chapter I-3 in [7]).
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-cf-comp-cf-id-right*

4.7 Constant functor

4.7.1 Definition and elementary properties

See Chapter III-3 in [7].

abbreviation *cf-const* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-const* $\mathfrak{C}$ $\mathfrak{D}$ $a \equiv$ *smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ a ($\mathfrak{D}$(|*CId*|)(|a|))

Slicing.

lemma *cf-smcf-cf-const*[*slicing-commute*]:
smcf-const (*cat-smc* $\mathfrak{C}$) (*cat-smc* $\mathfrak{D}$) a ($\mathfrak{D}$(|*CId*|)(|a|)) = *cf-smcf* (*cf-const* $\mathfrak{C}$ $\mathfrak{D}$ a)
⟨*proof*⟩

4.7.2 Object map and arrow map

context
fixes $\mathfrak{D}$ a :: V
begin

lemmas-with [**where** $\mathfrak{D}$=$\mathfrak{D}$ **and** a=a **and** f=‹$\mathfrak{D}$(|*CId*|)(|a|)›, *cat-cs-simps*]:
dghm-const-ObjMap-app
dghm-const-ArrMap-app

end

4.7.3 Opposite constant functor

lemma *op-cf-cf-const*[*cat-op-simps*]:

op-cf (*cf-const* $\mathfrak{C}$ $\mathfrak{D}$ *a*) = *cf-const* (*op-cat* $\mathfrak{C}$) (*op-cat* $\mathfrak{D}$) *a*
⟨*proof*⟩

4.7.4 A constant functor is a functor

lemma *cf-const-is-functor*:
assumes *category* α $\mathfrak{C}$ **and** *category* α $\mathfrak{D}$ **and** $a \in_\circ \mathfrak{D}(\!|Obj|\!)$
shows *cf-const* $\mathfrak{C}$ $\mathfrak{D}$ *a* : $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-const-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $a \in_\circ \mathfrak{D}(\!|Obj|\!)$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$
and $f = (\mathfrak{D}(\!|CId|\!)(\!|a|\!))$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f* : $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

4.7.5 Further properties

lemma *cf-comp-cf-const-right*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{G} \circ_{CF}$ *cf-const* $\mathfrak{A}$ $\mathfrak{B}$ *b* = *cf-const* $\mathfrak{A}$ $\mathfrak{C}$ ($\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

lemma *cf-comp-cf-const-right'*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $f = \mathfrak{B}(\!|CId|\!)(\!|b|\!)$
shows $\mathfrak{G} \circ_{CF}$ *dghm-const* $\mathfrak{A}$ $\mathfrak{B}$ *b f* = *cf-const* $\mathfrak{A}$ $\mathfrak{C}$ ($\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-comp-cf-const-left*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *cf-const* $\mathfrak{B}$ $\mathfrak{C}$ *a* $\circ_{CF}$ $\mathfrak{F}$ = *cf-const* $\mathfrak{A}$ $\mathfrak{C}$ *a*
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-comp-cf-const-left'*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
shows *dghm-const* $\mathfrak{B}$ $\mathfrak{C}$ *a f* $\circ_{CF}$ $\mathfrak{F}$ = *cf-const* $\mathfrak{A}$ $\mathfrak{C}$ *a*
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-comp-cf-const-left'*

4.8 Faithful functor

4.8.1 Definition and elementary properties

See Chapter I-3 in [7]).

locale *is-ft-functor* = *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *ft-cf-is-ft-semifunctor*[*slicing-intros*]:

cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC.faithful\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-ft-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
($\langle$(- :/ - $\mapsto\mapsto_{C.faithful1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-ft-functor* $\rightleftharpoons$ *is-ft-functor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.faithful\alpha} \mathfrak{B} \rightleftharpoons$ *CONST is-ft-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

lemma (in *is-ft-functor*) *ft-cf-is-ft-functor'*:
assumes $\mathfrak{A}' =$ *cat-smc* $\mathfrak{A}$ **and** $\mathfrak{B}' =$ *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-ft-functor.ft-cf-is-ft-functor'*

Rules.

lemma (in *is-ft-functor*) *is-ft-functor-axioms'*[*cf-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.faithful_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-ft-functor-def*[*unfolded is-ft-functor-axioms-def*]
|*intro is-ft-functorI*|
|*dest is-ft-functorD*[*dest*]|
|*elim is-ft-functorE*[*elim*]|

lemmas [*cf-cs-intros*] = *is-ft-functorD*(*1*)

lemma *is-ft-functorI'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow v11\ (\mathfrak{F}(\!|ArrMap|\!) \restriction^l_\circ Hom\ \mathfrak{A}\ a\ b)$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.faithful\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-ft-functorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.faithful\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow v11\ (\mathfrak{F}(\!|ArrMap|\!) \restriction^l_\circ Hom\ \mathfrak{A}\ a\ b)$
⟨*proof*⟩

lemma *is-ft-functorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.faithful\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow v11\ (\mathfrak{F}(\!|ArrMap|\!) \restriction^l_\circ Hom\ \mathfrak{A}\ a\ b)$
⟨*proof*⟩

lemma *is-ft-functorI''*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\bigwedge a\ b\ g\ f.$
$[\![\ g : a \mapsto_{\mathfrak{A}} b;\ f : a \mapsto_{\mathfrak{A}} b;\ \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)\ ]\!] \Longrightarrow g = f$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.faithful\alpha} \mathfrak{B}$
⟨*proof*⟩

Elementary properties.

context *is-ft-functor*
begin

interpretation *smcf*: *is-ft-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
ft-cf-v11-on-Hom = *smcf.ft-smcf-v11-on-Hom*
and *ft-cf-ArrMap-eqD* = *smcf.ft-smcf-ArrMap-eqD*

end

4.8.2 Opposite faithful functor.

lemma (**in** *is-ft-functor*) *is-ft-functor-op'*:
op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\mapsto\mapsto_{C.faithful\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-ft-functor*) *is-ft-functor-op*:
assumes $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
shows *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\mapsto\mapsto_{C.faithful\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

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

4.8.3 The composition of faithful functors is a faithful functor

lemma *cf-comp-is-ft-functor*[*cf-cs-intros*]:
assumes $\mathfrak{G}$: $\mathfrak{B}$ $\mapsto\mapsto_{C.faithful\alpha}$ $\mathfrak{C}$ **and** $\mathfrak{F}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.faithful\alpha}$ $\mathfrak{B}$
shows $\mathfrak{G}$ $\circ_{CF}$ $\mathfrak{F}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.faithful\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

4.9 Full functor

4.9.1 Definition and elementary properties

See Chapter I-3 in [7]).

locale *is-fl-functor* = *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *fl-cf-is-fl-semifunctor*:
cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC.full\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-fl-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $\mapsto\mapsto_{C.full}$1 -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-fl-functor* ⇌ *is-fl-functor*
translations $\mathfrak{F}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.full\alpha}$ $\mathfrak{B}$ ⇌ *CONST is-fl-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

lemma (**in** *is-fl-functor*) *fl-cf-is-fl-functor'*[*slicing-intros*]:
assumes $\mathfrak{A}'$ = *cat-smc* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F}$: $\mathfrak{A}'$ $\mapsto\mapsto_{SMC.full\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-fl-functor.fl-cf-is-fl-semifunctor*

Rules.

lemma (**in** *is-fl-functor*) *is-fl-functor-axioms'*[*cf-cs-intros*]:
assumes α' = α **and** $\mathfrak{A}'$ = $\mathfrak{A}$ **and** $\mathfrak{B}'$ = $\mathfrak{B}$
shows $\mathfrak{F}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C.full_{\alpha'}}$ $\mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-fl-functor-def*[*unfolded is-fl-functor-axioms-def*]
|*intro is-fl-functorI*|
|*dest is-fl-functorD*[*dest*]|
|*elim is-fl-functorE*[*elim*]|

lemmas [*cf-cs-intros*] = *is-fl-functorD*(*1*)

lemma *is-fl-functorI'*:
  assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow$
    $\mathfrak{F}(\!|ArrMap|\!)\ `_\circ\ (Hom\ \mathfrak{A}\ a\ b) = Hom\ \mathfrak{B}\ (\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))\ (\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
  shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.full\alpha} \mathfrak{B}$
  ⟨*proof*⟩

lemma *is-fl-functorD'*:
  assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.full\alpha} \mathfrak{B}$
  shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow$
    $\mathfrak{F}(\!|ArrMap|\!)\ `_\circ\ (Hom\ \mathfrak{A}\ a\ b) = Hom\ \mathfrak{B}\ (\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))\ (\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
  ⟨*proof*⟩

lemma *is-fl-functorE'*:
  assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.full\alpha} \mathfrak{B}$
  obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow$
    $\mathfrak{F}(\!|ArrMap|\!)\ `_\circ\ (Hom\ \mathfrak{A}\ a\ b) = Hom\ \mathfrak{B}\ (\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))\ (\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
  ⟨*proof*⟩

Elementary properties.

context *is-fl-functor*
begin

interpretation *smcf*: *is-fl-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
  fl-cf-surj-on-Hom = *smcf.fl-smcf-surj-on-Hom*

end

4.9.2 Opposite full functor

lemma (**in** *is-fl-functor*) *is-fl-functor-op*[*cat-op-intros*]:
  op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.full\alpha}$ *op-cat* $\mathfrak{B}$
  ⟨*proof*⟩

lemmas *is-fl-functor-op*[*cat-op-intros*] = *is-fl-functor.is-fl-functor-op*

4.9.3 The composition of full functor is a full functor

lemma *cf-comp-is-fl-functor*[*cf-cs-intros*]:
  assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.full\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.full\alpha} \mathfrak{B}$
  shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.full\alpha} \mathfrak{C}$
⟨*proof*⟩

4.10 Fully faithful functor

4.10.1 Definition and elementary properties

See Chapter I-3 in [7]).

locale *is-ff-functor* = *is-ft-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ + *is-fl-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

syntax *-is-ff-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
($\langle$(- :/ - $\mapsto\mapsto_{C.ff1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-ff-functor* $\rightleftharpoons$ *is-ff-functor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.ff\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}ff\text{-}functor\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

Rules.

mk-ide rf *is-ff-functor-def*
|*intro is-ff-functorI*|
|*dest is-ff-functorD*[*dest*]|
|*elim is-ff-functorE*[*elim*]|

lemmas [*cf-cs-intros*] = *is-ff-functorD*

Elementary properties.

lemma (**in** *is-ff-functor*) *ff-cf-is-ff-semifunctor*:
cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.ff\alpha}$ *cat-smc* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-ff-functor*) *ff-cf-is-ff-semifunctor'*[*slicing-intros*]:
assumes $\mathfrak{A}'$ = *cat-smc* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.ff\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-ff-functor.ff-cf-is-ff-semifunctor'*

4.10.2 Opposite fully faithful functor

lemma (**in** *is-ff-functor*) *is-ff-functor-op*:
op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.ff\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-ff-functor*) *is-ff-functor-op'*[*cat-op-intros*]:
assumes $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
shows *op-cf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.ff\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-ff-functor-op*[*cat-op-intros*] = *is-ff-functor.is-ff-functor-op*

4.10.3 The composition of fully faithful functors is a fully faithful functor

lemma *cf-comp-is-ff-functor*[*cf-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.ff\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.ff\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.ff\alpha} \mathfrak{C}$
⟨*proof*⟩

4.11 Isomorphism of categories

4.11.1 Definition and elementary properties

See Chapter I-3 in [7]).

locale *is-iso-functor* = *is-functor* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ **for** $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ +
assumes *iso-cf-is-iso-semifunctor*:
cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-iso-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
($\langle$(- :/ - $\mapsto\mapsto_{C.iso1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-iso-functor* $\rightleftharpoons$ *is-iso-functor*

translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B} \rightleftharpoons$ *CONST is-iso-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

lemma (in *is-iso-functor*) *iso-cf-is-iso-semifunctor'*[*slicing-intros*]:
assumes $\mathfrak{A}' =$ *cat-smc* $\mathfrak{A}$ $\mathfrak{B}' =$ *cat-smc* $\mathfrak{B}$
shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-iso-semifunctor.iso-smcf-is-iso-dghm'*

Rules.

lemma (in *is-iso-functor*) *is-iso-functor-axioms'*[*cf-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.iso_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-iso-functor-def*[*unfolded is-iso-functor-axioms-def*]
|*intro is-iso-functorI*|
|*dest is-iso-functorD*[*dest*]|
|*elim is-iso-functorE*[*elim*]|

lemma *is-iso-functorI'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *v11* ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$)
and *v11* ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$)
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Obj*$\rfloor)$
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Arr*$\rfloor)$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-iso-functorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *v11* ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$)
and *v11* ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$)
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Obj*$\rfloor)$
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Arr*$\rfloor)$
⟨*proof*⟩

lemma *is-iso-functorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *v11* ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$)
and *v11* ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$)
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ObjMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Obj*$\rfloor)$
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor$*ArrMap*$\rfloor)$) = $\mathfrak{B}(\lfloor$*Arr*$\rfloor)$
⟨*proof*⟩

Elementary properties.

context *is-iso-functor*
begin

interpretation *smcf*: *is-iso-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
iso-cf-ObjMap-vrange[*simp*] = *smcf.iso-smcf-ObjMap-vrange*
and *iso-cf-ArrMap-vrange*[*simp*] = *smcf.iso-smcf-ArrMap-vrange*

sublocale *ObjMap*: *v11* ‹$\mathfrak{F}(\!|ObjMap|\!)$›
  rewrites $\mathcal{D}_\circ\ (\mathfrak{F}(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$ **and** $\mathcal{R}_\circ\ (\mathfrak{F}(\!|ObjMap|\!)) = \mathfrak{B}(\!|Obj|\!)$
  ⟨*proof*⟩

sublocale *ArrMap*: *v11* ‹$\mathfrak{F}(\!|ArrMap|\!)$›
  rewrites $\mathcal{D}_\circ\ (\mathfrak{F}(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$ **and** $\mathcal{R}_\circ\ (\mathfrak{F}(\!|ArrMap|\!)) = \mathfrak{B}(\!|Arr|\!)$
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
  iso-cf-Obj-HomDom-if-Obj-HomCod[*elim*] =
    smcf.iso-smcf-Obj-HomDom-if-Obj-HomCod
  and *iso-cf-Arr-HomDom-if-Arr-HomCod*[*elim*] =
    smcf.iso-smcf-Arr-HomDom-if-Arr-HomCod
  and *iso-cf-ObjMap-eqE*[*elim*] = *smcf.iso-smcf-ObjMap-eqE*
  and *iso-cf-ArrMap-eqE*[*elim*] = *smcf.iso-smcf-ArrMap-eqE*

end

sublocale *is-iso-functor* ⊆ *is-ff-functor*
⟨*proof*⟩

lemmas (**in** *is-iso-functor*) *iso-cf-is-ff-functor* = *is-ff-functor-axioms*
lemmas [*cf-cs-intros*] = *is-iso-functor.iso-cf-is-ff-functor*

4.11.2 Opposite isomorphism of categories

lemma (**in** *is-iso-functor*) *is-iso-functor-op*:
  op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.iso\alpha}$ *op-cat* $\mathfrak{B}$
  ⟨*proof*⟩

lemma (**in** *is-iso-functor*) *is-iso-functor-op'*:
  assumes $\mathfrak{A}'$ = *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-cat* $\mathfrak{B}$
  shows *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.iso\alpha}$ *op-cat* $\mathfrak{B}$
  ⟨*proof*⟩

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

4.11.3 The composition of isomorphisms of categories is an isomorphism of categories

lemma *cf-comp-is-iso-functor*[*cf-cs-intros*]:
  assumes $\mathfrak{G}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C.iso\alpha}$ $\mathfrak{C}$ **and** $\mathfrak{F}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.iso\alpha}$ $\mathfrak{B}$
  shows $\mathfrak{G} \circ_{CF} \mathfrak{F}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.iso\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

4.12 Inverse functor

abbreviation (*input*) *inv-cf* :: $V \Rightarrow V$
  where *inv-cf* ≡ *inv-dghm*

Slicing.

lemma *dghm-inv-semifunctor*[*slicing-commute*]:
  inv-smcf (*cf-smcf* $\mathfrak{F}$) = *cf-smcf* (*inv-cf* $\mathfrak{F}$)
  ⟨*proof*⟩

context *is-iso-functor*
begin

interpretation *smcf*: *is-iso-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
inv-cf-ObjMap-v11 = *smcf.inv-smcf-ObjMap-v11*
and *inv-cf-ObjMap-vdomain* = *smcf.inv-smcf-ObjMap-vdomain*
and *inv-cf-ObjMap-app* = *smcf.inv-smcf-ObjMap-app*
and *inv-cf-ObjMap-vrange* = *smcf.inv-smcf-ObjMap-vrange*
and *inv-cf-ArrMap-v11* = *smcf.inv-smcf-ArrMap-v11*
and *inv-cf-ArrMap-vdomain* = *smcf.inv-smcf-ArrMap-vdomain*
and *inv-cf-ArrMap-app* = *smcf.inv-smcf-ArrMap-app*
and *inv-cf-ArrMap-vrange* = *smcf.inv-smcf-ArrMap-vrange*
and *iso-cf-ObjMap-inv-cf-ObjMap-app*[*cf-cs-simps*] =
smcf.iso-smcf-ObjMap-inv-smcf-ObjMap-app
and *iso-cf-ArrMap-inv-cf-ArrMap-app*[*cf-cs-simps*] =
smcf.iso-smcf-ArrMap-inv-smcf-ArrMap-app
and *iso-cf-HomDom-is-arr-conv* = *smcf.iso-smcf-HomDom-is-arr-conv*
and *iso-cf-HomCod-is-arr-conv* = *smcf.iso-smcf-HomCod-is-arr-conv*
and *iso-inv-cf-ObjMap-cf-ObjMap-app*[*cf-cs-simps*] =
smcf.iso-inv-smcf-ObjMap-smcf-ObjMap-app
and *iso-inv-cf-ArrMap-cf-ArrMap-app*[*cf-cs-simps*] =
smcf.iso-inv-smcf-ArrMap-smcf-ArrMap-app

end

lemmas [*cf-cs-intros*] =
is-iso-functor.inv-cf-ObjMap-v11
is-iso-functor.inv-cf-ArrMap-v11

lemmas [*cf-cs-simps*] =
is-iso-functor.inv-cf-ObjMap-vdomain
is-iso-functor.inv-cf-ObjMap-app
is-iso-functor.inv-cf-ObjMap-vrange
is-iso-functor.inv-cf-ArrMap-vdomain
is-iso-functor.inv-cf-ArrMap-app
is-iso-functor.inv-cf-ArrMap-vrange
is-iso-functor.iso-cf-ObjMap-inv-cf-ObjMap-app
is-iso-functor.iso-cf-ArrMap-inv-cf-ArrMap-app
is-iso-functor.iso-inv-cf-ObjMap-cf-ObjMap-app
is-iso-functor.iso-inv-cf-ArrMap-cf-ArrMap-app

4.13 An isomorphism of categories is an isomorphism in the category CAT

lemma *is-iso-arr-is-iso-functor*:
— See Chapter I-3 in [7].
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
and $\mathfrak{G} \circ_{CF} \mathfrak{F}$ = *cf-id* $\mathfrak{A}$
and $\mathfrak{F} \circ_{CF} \mathfrak{G}$ = *cf-id* $\mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-iso-functor-is-iso-arr*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
shows [*cf-cs-intros*]: *inv-cf* $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C.iso\alpha} \mathfrak{A}$
and [*cf-cs-simps*]: *inv-cf* $\mathfrak{F} \circ_{CF} \mathfrak{F}$ = *cf-id* $\mathfrak{A}$
and [*cf-cs-simps*]: $\mathfrak{F} \circ_{CF}$ *inv-cf* $\mathfrak{F}$ = *cf-id* $\mathfrak{B}$

⟨*proof*⟩

4.13.1 An identity functor is an isomorphism of categories

lemma (**in** *category*) *cat-cf-id-is-iso-functor*: *cf-id* $\mathfrak{C} : \mathfrak{C} \mapsto\mapsto_{C.iso\alpha} \mathfrak{C}$
⟨*proof*⟩

4.14 Isomorphic categories

4.14.1 Definition and elementary properties

See Chapter I-3 in [7]).

locale *iso-category* = *L*: *category* α $\mathfrak{A}$ + *R*: *category* α $\mathfrak{B}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ +
assumes *iso-cat-is-iso-functor*: $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$

notation *iso-category* (**infixl** $\langle\approx_{C1}\rangle$ *50*)

Rules.

lemma *iso-categoryI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *iso-categoryD*[*dest*]:
assumes $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
shows $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *iso-categoryE*[*elim*]:
assumes $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
obtains $\mathfrak{F}$ **where** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

Isomorphic categories are isomorphic semicategories.

lemma (**in** *iso-category*) *iso-cat-iso-semicategory*:
cat-smc $\mathfrak{A} \approx_{SMC\alpha}$ *cat-smc* $\mathfrak{B}$
⟨*proof*⟩

4.14.2 A category isomorphism is an equivalence relation

lemma *iso-category-refl*:
assumes *category* α $\mathfrak{A}$
shows $\mathfrak{A} \approx_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *iso-category-sym*[*sym*]:
assumes $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
shows $\mathfrak{B} \approx_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *iso-category-trans*[*trans*]:
assumes $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \approx_{C\alpha} \mathfrak{C}$
shows $\mathfrak{A} \approx_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

5 Smallness for functors

5.1 Functor with tiny maps

5.1.1 Definition and elementary properties

locale *is-tm-functor* = *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *tm-cf-is-semifunctor*[*slicing-intros*]:
    cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC.tm\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-tm-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  ($\langle$(- :/ - $\mapsto\mapsto_{C.tm1}$ -)$\rangle$ [*51*, *51*, *51*] *51*)
syntax-consts *-is-tm-functor* $\rightleftharpoons$ *is-tm-functor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B} \rightleftharpoons$ *CONST is-tm-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation (*input*) *is-cn-tm-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *is-cn-tm-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} \equiv \mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$

syntax *-is-cn-tm-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  ($\langle$(- :/ - $_{C.tm}\mapsto\mapsto_1$ -)$\rangle$ [*51*, *51*, *51*] *51*)
syntax-consts *-is-cn-tm-functor* $\rightleftharpoons$ *is-cn-tm-functor*
translations $\mathfrak{F} : \mathfrak{A}\ {}_{C.tm}\mapsto\mapsto_{\alpha}\ \mathfrak{B} \rightharpoonup$ *CONST is-cn-tm-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation *all-tm-cfs* :: $V \Rightarrow V$
  where *all-tm-cfs* $\alpha \equiv$ *set* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

abbreviation *small-tm-cfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *small-tm-cfs* α $\mathfrak{A}$ $\mathfrak{B} \equiv$ *set* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

lemma (**in** *is-tm-functor*) *tm-cf-is-semifunctor'*:
  assumes $\alpha' = \alpha$
    and $\mathfrak{A}' =$ *cat-smc* $\mathfrak{A}$
    and $\mathfrak{B}' =$ *cat-smc* $\mathfrak{B}$
  shows *cf-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.tm_{\alpha'}} \mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *is-tm-functor.tm-cf-is-semifunctor'*

Rules.

lemma (**in** *is-tm-functor*) *is-tm-functor-axioms'*[*cat-small-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
  shows $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{B}'$
  ⟨*proof*⟩

mk-ide rf *is-tm-functor-def*[*unfolded is-tm-functor-axioms-def*]
  |*intro is-tm-functorI*|
  |*dest is-tm-functorD*[*dest*]|
  |*elim is-tm-functorE*[*elim*]|

lemmas [*cat-small-cs-intros*] = *is-tm-functorD*(*1*)

Slicing.

context *is-tm-functor*
begin

interpretation *smcf*: *is-tm-semifunctor* α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:

tm-cf-ObjMap-in-Vset[*cat-cs-intros*] = *smcf.tm-smcf-ObjMap-in-Vset*
and *tm-cf-ArrMap-in-Vset*[*cat-cs-intros*] = *smcf.tm-smcf-ArrMap-in-Vset*

end

sublocale *is-tm-functor* ⊆ *HomDom*: *tiny-category* α $\mathfrak{A}$
⟨*proof*⟩

Further rules.

lemma *is-tm-functorI'*:
assumes [*simp*]: $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and [*simp*]: $\mathfrak{F}(\lfloor ObjMap \rfloor) \in_\circ Vset\ \alpha$
and [*simp*]: $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_\circ Vset\ \alpha$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tm-functorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F}(\lfloor ObjMap \rfloor) \in_\circ Vset\ \alpha$
and $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functorD'*(*1*)

lemma *is-tm-functorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F}(\lfloor ObjMap \rfloor) \in_\circ Vset\ \alpha$
and $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_\circ Vset\ \alpha$
⟨*proof*⟩

Size.

lemma *small-all-tm-cfs*[*simp*]: *small* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

5.1.2 Opposite functor with tiny maps

lemma (**in** *is-tm-functor*) *is-tm-functor-op*:
op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *is-tm-functor-op'*[*cat-op-intros*]:
assumes $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}' =$ *op-cat* $\mathfrak{B}$ **and** $\alpha' = \alpha$
shows *op-cf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

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

5.1.3 Composition of functors with tiny maps

lemma *cf-comp-is-tm-functor*[*cat-small-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

5.1.4 Finite categories and functors with tiny maps

lemma (**in** *is-functor*) *cf-is-tm-functor-if-HomDom-finite-category*:

assumes *finite-category* α $\mathfrak{A}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

5.1.5 Constant functor with tiny maps

lemma *cf-const-is-tm-functor*:
assumes *tiny-category* α $\mathfrak{C}$ **and** *category* α $\mathfrak{D}$ **and** $a \in_\circ \mathfrak{D}(\lfloor Obj \rfloor)$
shows *cf-const* $\mathfrak{C}$ $\mathfrak{D}$ $a : \mathfrak{C} \mapsto\mapsto_{C.tm\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-const-is-tm-functor'*[*cat-small-cs-intros*]:
assumes *tiny-category* α $\mathfrak{C}$
and *category* α $\mathfrak{D}$
and $a \in_\circ \mathfrak{D}(\lfloor Obj \rfloor)$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows *cf-const* $\mathfrak{C}$ $\mathfrak{D}$ $a : \mathfrak{C}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{D}'$
⟨*proof*⟩

5.2 Tiny functor

5.2.1 Definition and elementary properties

locale *is-tiny-functor* = *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *tiny-cf-is-tiny-semifunctor*[*slicing-intros*]:
cf-smcf $\mathfrak{F}$: *cat-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-tiny-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - $\mapsto\mapsto_{C.tiny}$1 -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-tiny-functor* ⇌ *is-tiny-functor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$ ⇌ *CONST is-tiny-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation (*input*) *is-cn-tiny-cf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
where *is-cn-tiny-cf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} \equiv \mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$

syntax *-is-cn-tiny-cf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - $_{C.tiny}\mapsto\mapsto_1$ -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-cn-tiny-cf* ⇌ *is-cn-cf*
translations $\mathfrak{F} : \mathfrak{A}$ $_{C.tiny}\mapsto\mapsto_\alpha$ $\mathfrak{B}$ ⇀ *CONST is-cn-cf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation *all-tiny-cfs* :: $V \Rightarrow V$
where *all-tiny-cfs* $\alpha \equiv$ *set* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$

abbreviation *tiny-cfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *tiny-cfs* α $\mathfrak{A}$ $\mathfrak{B} \equiv$ *set* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$

lemmas [*slicing-intros*] = *is-tiny-functor.tiny-cf-is-tiny-semifunctor*

Rules.

lemma (**in** *is-tiny-functor*) *is-tiny-functor-axioms'*[*cat-small-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.tiny\alpha'} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-tiny-functor-def*[*unfolded is-tiny-functor-axioms-def*]
|*intro is-tiny-functorI*|
|*dest is-tiny-functorD*[*dest*]|
|*elim is-tiny-functorE*[*elim*]|

lemmas [*cat-small-cs-intros*] = *is-tiny-functorD*(*1*)

Elementary properties.

sublocale *is-tiny-functor* ⊆ *HomDom*: *tiny-category* α $\mathfrak{A}$
⟨*proof*⟩

sublocale *is-tiny-functor* ⊆ *HomCod*: *tiny-category* α $\mathfrak{B}$
⟨*proof*⟩

sublocale *is-tiny-functor* ⊆ *is-tm-functor*
⟨*proof*⟩

Further rules.

lemma *is-tiny-functorI′*:
assumes [*simp*]: $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *tiny-category* α $\mathfrak{A}$
and *tiny-category* α $\mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-functorD′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *tiny-category* α $\mathfrak{A}$
and *tiny-category* α $\mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tiny-functorD′*(*2*,*3*)

lemma *is-tiny-functorE′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *tiny-category* α $\mathfrak{A}$
and *tiny-category* α $\mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-functor-iff*:
$\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \longleftrightarrow$
$(\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B} \wedge$ *tiny-category* α $\mathfrak{A} \wedge$ *tiny-category* α $\mathfrak{B})$
⟨*proof*⟩

Size.

lemma (**in** *is-tiny-functor*) *tiny-cf-in-Vset*: $\mathfrak{F} \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *small-all-tiny-cfs*[*simp*]: *small* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-tiny-cfs*[*simp*]: *small* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *all-tiny-cfs-vsubset-Vset*[*simp*]:
set $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\} \subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-is-tiny-functor-if-ge-Limit*:

assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} \mathfrak{B}$
⟨*proof*⟩

5.2.2 Opposite tiny semifunctor

lemma (**in** *is-tiny-functor*) *is-tiny-functor-op*:
op-cf $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.tiny\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tiny-functor*) *is-tiny-functor-op'*[*cat-op-intros*]:
assumes $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$ **and** $\mathfrak{B}' =$ *op-cat* $\mathfrak{B}$ **and** $\alpha' = \alpha$
shows *op-cf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.tiny_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

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

5.2.3 Composition of tiny functors

lemma *cf-comp-is-tiny-functor*[*cat-small-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{C}$
⟨*proof*⟩

5.2.4 Tiny constant functor

lemma *cf-const-is-tiny-functor*:
assumes *tiny-category* $\alpha\ \mathfrak{C}$ **and** *tiny-category* $\alpha\ \mathfrak{D}$ **and** $a \in_{\circ} \mathfrak{D}(\!|Obj|\!)$
shows *cf-const* $\mathfrak{C}\ \mathfrak{D}\ a : \mathfrak{C} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-const-is-tiny-functor'*:
assumes *tiny-category* $\alpha\ \mathfrak{C}$
and *tiny-category* $\alpha\ \mathfrak{D}$
and $a \in_{\circ} \mathfrak{D}(\!|Obj|\!)$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows *cf-const* $\mathfrak{C}\ \mathfrak{D}\ a : \mathfrak{C}' \mapsto\mapsto_{C.tiny\alpha} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *cf-const-is-tiny-functor'*

6 Natural transformation

6.1 Background

named-theorems *ntcf-cs-simps*
named-theorems *ntcf-cs-intros*

lemmas [*cat-cs-simps*] = *dg-shared-cs-simps*
lemmas [*cat-cs-intros*] = *dg-shared-cs-intros*

6.1.1 Slicing

definition *ntcf-ntsmcf* :: $V \Rightarrow V$
  where *ntcf-ntsmcf* $\mathfrak{N}$ =
    [
      $\mathfrak{N}(\!|NTMap|\!)$,
      cf-smcf ($\mathfrak{N}(\!|NTDom|\!)$),
      cf-smcf ($\mathfrak{N}(\!|NTCod|\!)$),
      cat-smc ($\mathfrak{N}(\!|NTDGDom|\!)$),
      cat-smc ($\mathfrak{N}(\!|NTDGCod|\!)$)
    ]$_{\circ}$

Components.

lemma *ntcf-ntsmcf-components*:
  shows [*slicing-simps*]: *ntcf-ntsmcf* $\mathfrak{N}(\!|NTMap|\!)$ = $\mathfrak{N}(\!|NTMap|\!)$
    and [*slicing-commute*]: *ntcf-ntsmcf* $\mathfrak{N}(\!|NTDom|\!)$ = *cf-smcf* ($\mathfrak{N}(\!|NTDom|\!)$)
    and [*slicing-commute*]: *ntcf-ntsmcf* $\mathfrak{N}(\!|NTCod|\!)$ = *cf-smcf* ($\mathfrak{N}(\!|NTCod|\!)$)
    and [*slicing-commute*]: *ntcf-ntsmcf* $\mathfrak{N}(\!|NTDGDom|\!)$ = *cat-smc* ($\mathfrak{N}(\!|NTDGDom|\!)$)
    and [*slicing-commute*]: *ntcf-ntsmcf* $\mathfrak{N}(\!|NTDGCod|\!)$ = *cat-smc* ($\mathfrak{N}(\!|NTDGCod|\!)$)
  ⟨*proof*⟩

6.2 Definition and elementary properties

The definition of a natural transformation that is used in this work is is similar to the definition that can be found in Chapter I-4 in [7].

locale *is-ntcf* =
  $\mathcal{Z}$ α +
  vfsequence $\mathfrak{N}$ +
  NTDom: *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  NTCod: *is-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ +
  assumes *ntcf-length*[*cat-cs-simps*]: *vcard* $\mathfrak{N}$ = $5_{\mathbf{N}}$
    and *ntcf-is-ntsmcf*[*slicing-intros*]: *ntcf-ntsmcf* $\mathfrak{N}$:
      cf-smcf $\mathfrak{F}$ $\mapsto_{SMC}$ *cf-smcf* $\mathfrak{G}$: *cat-smc* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ *cat-smc* $\mathfrak{B}$
    and *ntcf-NTDom*[*cat-cs-simps*]: $\mathfrak{N}(\!|NTDom|\!)$ = $\mathfrak{F}$
    and *ntcf-NTCod*[*cat-cs-simps*]: $\mathfrak{N}(\!|NTCod|\!)$ = $\mathfrak{G}$
    and *ntcf-NTDGDom*[*cat-cs-simps*]: $\mathfrak{N}(\!|NTDGDom|\!)$ = $\mathfrak{A}$
    and *ntcf-NTDGCod*[*cat-cs-simps*]: $\mathfrak{N}(\!|NTDGCod|\!)$ = $\mathfrak{B}$

syntax *-is-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $\mapsto_{CF}$ - :/ - $\mapsto\!\!\mapsto_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-ntcf* ⇌ *is-ntcf*
translations $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$ ⇌ *CONST is-ntcf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

abbreviation *all-ntcfs* :: $V \Rightarrow V$
  where *all-ntcfs* α ≡ *set* {$\mathfrak{N}$. ∃$\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$}

abbreviation *ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\equiv$ *set* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$

abbreviation *these-ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *these-ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\equiv$ *set* $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$

lemmas [*cat-cs-simps*] =
is-ntcf.ntcf-length
is-ntcf.ntcf-NTDom
is-ntcf.ntcf-NTCod
is-ntcf.ntcf-NTDGDom
is-ntcf.ntcf-NTDGCod

lemma (**in** *is-ntcf*) *ntcf-is-ntsmcf'*:
assumes $\mathfrak{F}'$ = *cf-smcf* $\mathfrak{F}$
and $\mathfrak{G}'$ = *cf-smcf* $\mathfrak{G}$
and $\mathfrak{A}'$ = *cat-smc* $\mathfrak{A}$
and $\mathfrak{B}'$ = *cat-smc* $\mathfrak{B}$
shows *ntcf-ntsmcf* $\mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-ntcf.ntcf-is-ntsmcf'*

Rules.

lemma (**in** *is-ntcf*) *is-ntcf-axioms'*[*cat-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$ **and** $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{G}$
shows $\mathfrak{N} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-ntcf-def*[*unfolded is-ntcf-axioms-def*]
|*intro is-ntcfI*|
|*dest is-ntcfD*[*dest*]|
|*elim is-ntcfE*[*elim*]|

lemmas [*cat-cs-intros*] =
is-ntcfD(*3*,*4*)

lemma *is-ntcfI'*:
assumes $\mathcal{Z}$ α
and *vfsequence* $\mathfrak{N}$
and *vcard* $\mathfrak{N}$ = $5_{\mathbb{N}}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}(\!|NTDom|\!) = \mathfrak{F}$
and $\mathfrak{N}(\!|NTCod|\!) = \mathfrak{G}$
and $\mathfrak{N}(\!|NTDGDom|\!) = \mathfrak{A}$
and $\mathfrak{N}(\!|NTDGCod|\!) = \mathfrak{B}$
and *vsv* $(\mathfrak{N}(\!|NTMap|\!))$
and $\mathcal{D}_\circ\ (\mathfrak{N}(\!|NTMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
and $\bigwedge a.\ a \in_\circ \mathfrak{A}(\!|Obj|\!) \Longrightarrow \mathfrak{N}(\!|NTMap|\!)(\!|a|\!) : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{N}(\!|NTMap|\!)(\!|b|\!) \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{G}(\!|ArrMap|\!)(\!|f|\!) \circ_{A\mathfrak{B}} \mathfrak{N}(\!|NTMap|\!)(\!|a|\!)$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-ntcfD'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{Z}$ α
and *vfsequence* $\mathfrak{N}$

and $vcard\ \mathfrak{N} = 5_{\mathbb{N}}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}(\lfloor NTDom \rfloor) = \mathfrak{F}$
and $\mathfrak{N}(\lfloor NTCod \rfloor) = \mathfrak{G}$
and $\mathfrak{N}(\lfloor NTDGDom \rfloor) = \mathfrak{A}$
and $\mathfrak{N}(\lfloor NTDGCod \rfloor) = \mathfrak{B}$
and $vsv\ (\mathfrak{N}(\lfloor NTMap \rfloor))$
and $\mathcal{D}_\circ\ (\mathfrak{N}(\lfloor NTMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
and $\bigwedge a.\ a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
⟨*proof*⟩

lemma *is-ntcfE′*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
obtains $\mathcal{Z}\ \alpha$
and $vfsequence\ \mathfrak{N}$
and $vcard\ \mathfrak{N} = 5_{\mathbb{N}}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}(\lfloor NTDom \rfloor) = \mathfrak{F}$
and $\mathfrak{N}(\lfloor NTCod \rfloor) = \mathfrak{G}$
and $\mathfrak{N}(\lfloor NTDGDom \rfloor) = \mathfrak{A}$
and $\mathfrak{N}(\lfloor NTDGCod \rfloor) = \mathfrak{B}$
and $vsv\ (\mathfrak{N}(\lfloor NTMap \rfloor))$
and $\mathcal{D}_\circ\ (\mathfrak{N}(\lfloor NTMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
and $\bigwedge a.\ a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
$\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
⟨*proof*⟩

Slicing.

context *is-ntcf*
begin

interpretation *ntsmcf*:
is-ntsmcf α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$› ‹*cf-smcf* $\mathfrak{G}$› ‹*ntcf-ntsmcf* $\mathfrak{N}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
ntcf-NTMap-vsv = *ntsmcf.ntsmcf-NTMap-vsv*
and *ntcf-NTMap-vdomain*[*cat-cs-simps*] = *ntsmcf.ntsmcf-NTMap-vdomain*
and *ntcf-NTMap-is-arr* = *ntsmcf.ntsmcf-NTMap-is-arr*
and *ntcf-NTMap-is-arr′*[*cat-cs-intros*] = *ntsmcf.ntsmcf-NTMap-is-arr′*

sublocale *NTMap*: *vsv* ‹$\mathfrak{N}(\lfloor NTMap \rfloor)$›
rewrites $\mathcal{D}_\circ\ (\mathfrak{N}(\lfloor NTMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
ntcf-NTMap-app-in-Arr[*cat-cs-intros*] = *ntsmcf.ntsmcf-NTMap-app-in-Arr*
and *ntcf-NTMap-vrange-vifunion* = *ntsmcf.ntsmcf-NTMap-vrange-vifunion*
and *ntcf-NTMap-vrange* = *ntsmcf.ntsmcf-NTMap-vrange*
and *ntcf-NTMap-vsubset-Vset* = *ntsmcf.ntsmcf-NTMap-vsubset-Vset*
and *ntcf-NTMap-in-Vset* = *ntsmcf.ntsmcf-NTMap-in-Vset*
and *ntcf-is-ntsmcf-if-ge-Limit* = *ntsmcf.ntsmcf-is-ntsmcf-if-ge-Limit*

lemmas-with [*unfolded slicing-simps*]:
ntcf-Comp-commute[*cat-cs-intros*] = *ntsmcf.ntsmcf-Comp-commute*
and *ntcf-Comp-commute′* = *ntsmcf.ntsmcf-Comp-commute′*
and *ntcf-Comp-commute″* = *ntsmcf.ntsmcf-Comp-commute″*

end

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-NTMap-vdomain*

lemmas [*cat-cs-intros*] =
is-ntcf.ntcf-NTMap-vsv
is-ntcf.ntcf-NTMap-is-arr′
ntsmcf-hcomp-NTMap-vsv

Elementary properties.

lemma *ntcf-eqI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
and $\mathfrak{N}(\!|NTMap|\!) = \mathfrak{N}'(\!|NTMap|\!)$
and $\mathfrak{F} = \mathfrak{F}'$
and $\mathfrak{G} = \mathfrak{G}'$
and $\mathfrak{A} = \mathfrak{A}'$
and $\mathfrak{B} = \mathfrak{B}'$
shows $\mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

lemma *ntcf-ntsmcf-eqI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
and $\mathfrak{F} = \mathfrak{F}'$
and $\mathfrak{G} = \mathfrak{G}'$
and $\mathfrak{A} = \mathfrak{A}'$
and $\mathfrak{B} = \mathfrak{B}'$
and *ntcf-ntsmcf* $\mathfrak{N}$ = *ntcf-ntsmcf* $\mathfrak{N}'$
shows $\mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-def*:
$\mathfrak{N} = [\mathfrak{N}(\!|NTMap|\!), \mathfrak{N}(\!|NTDom|\!), \mathfrak{N}(\!|NTCod|\!), \mathfrak{N}(\!|NTDGDom|\!), \mathfrak{N}(\!|NTDGCod|\!)]_\circ$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $\mathfrak{N} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-is-ntcf-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\beta} \mathfrak{B}$
⟨*proof*⟩

lemma *small-all-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-ntcfs*[*simp*]: *small* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-these-ntcfs*[*simp*]: *small* {𝔑. 𝔑 : 𝔉 ↦$_{CF}$ 𝔊 : 𝔄 ↦↦$_{C\alpha}$ 𝔅}
⟨*proof*⟩

Further elementary results.

lemma *these-ntcfs-iff*:
𝔑 ∈$_\circ$ *these-ntcfs* α 𝔄 𝔅 𝔉 𝔊 ⟷ 𝔑 : 𝔉 ↦$_{CF}$ 𝔊 : 𝔄 ↦↦$_{C\alpha}$ 𝔅
⟨*proof*⟩

6.3 Opposite natural transformation

See section 1.5 in [3].

definition *op-ntcf* :: *V* ⇒ *V*
where *op-ntcf* 𝔑 =
[
𝔑(|*NTMap*|),
op-cf (𝔑(|*NTCod*|)),
op-cf (𝔑(|*NTDom*|)),
op-cat (𝔑(|*NTDGDom*|)),
op-cat (𝔑(|*NTDGCod*|))
]$_\circ$

Components.

lemma *op-ntcf-components*[*cat-op-simps*]:
shows *op-ntcf* 𝔑(|*NTMap*|) = 𝔑(|*NTMap*|)
and *op-ntcf* 𝔑(|*NTDom*|) = *op-cf* (𝔑(|*NTCod*|))
and *op-ntcf* 𝔑(|*NTCod*|) = *op-cf* (𝔑(|*NTDom*|))
and *op-ntcf* 𝔑(|*NTDGDom*|) = *op-cat* (𝔑(|*NTDGDom*|))
and *op-ntcf* 𝔑(|*NTDGCod*|) = *op-cat* (𝔑(|*NTDGCod*|))
⟨*proof*⟩

Slicing.

lemma *ntcf-ntsmcf-op-ntcf*[*slicing-commute*]:
op-ntsmcf (*ntcf-ntsmcf* 𝔑) = *ntcf-ntsmcf* (*op-ntcf* 𝔑)
⟨*proof*⟩

Elementary properties.

lemma *op-ntcf-vsv*[*cat-op-intros*]: *vsv* (*op-ntcf* 𝔉)
⟨*proof*⟩

6.3.1 Further properties

lemma (**in** *is-ntcf*) *is-ntcf-op*:
op-ntcf 𝔑 : *op-cf* 𝔊 ↦$_{CF}$ *op-cf* 𝔉 : *op-cat* 𝔄 ↦↦$_{C\alpha}$ *op-cat* 𝔅
⟨*proof*⟩

lemma (**in** *is-ntcf*) *is-ntcf-op'*[*cat-op-intros*]:
assumes 𝔊' = *op-cf* 𝔊
and 𝔉' = *op-cf* 𝔉
and 𝔄' = *op-cat* 𝔄
and 𝔅' = *op-cat* 𝔅
shows *op-ntcf* 𝔑 : 𝔊' ↦$_{CF}$ 𝔉' : 𝔄' ↦↦$_{C\alpha}$ 𝔅'
⟨*proof*⟩

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

lemma (**in** *is-ntcf*) *ntcf-op-ntcf-op-ntcf*[*cat-op-simps*]:
op-ntcf (*op-ntcf* 𝔑) = 𝔑

⟨*proof*⟩

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

lemma *eq-op-ntcf-iff*[*cat-op-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
shows *op-ntcf* $\mathfrak{N}$ = *op-ntcf* $\mathfrak{N}' \longleftrightarrow \mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

6.4 Vertical composition of natural transformations

6.4.1 Definition and elementary properties

See Chapter II-4 in [7].

abbreviation (*input*) *ntcf-vcomp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\bullet_{NTCF}\rangle$ *55*)
where *ntcf-vcomp* ≡ *ntsmcf-vcomp*

lemmas [*cat-cs-simps*] = *ntsmcf-vcomp-components*(*2–5*)

Slicing.

lemma *ntcf-ntsmcf-ntcf-vcomp*[*slicing-commute*]:
ntcf-ntsmcf $\mathfrak{M} \bullet_{NTSMCF}$ *ntcf-ntsmcf* $\mathfrak{N}$ = *ntcf-ntsmcf* $(\mathfrak{M} \bullet_{NTCF} \mathfrak{N})$
⟨*proof*⟩

6.4.2 Natural transformation map

lemma *ntcf-vcomp-NTMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ\ ((\mathfrak{M} \bullet_{NTCF} \mathfrak{N})(\!|NTMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *ntcf-vcomp-NTMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $(\mathfrak{M} \bullet_{NTCF} \mathfrak{N})(\!|NTMap|\!)(\!|a|\!) = \mathfrak{M}(\!|NTMap|\!)(\!|a|\!) \circ_{A\mathfrak{B}} \mathfrak{N}(\!|NTMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemma *ntcf-vcomp-NTMap-vrange*:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ\ ((\mathfrak{M} \bullet_{NTCF} \mathfrak{N})(\!|NTMap|\!)) \subseteq_\circ \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

6.4.3 Further properties

lemma *ntcf-vcomp-composable-commute*[*cat-cs-simps*]:
— See Chapter II-4 in [7].
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and [*intro*]: $f : a \mapsto_{\mathfrak{A}} b$
shows
$(\mathfrak{M}(\!|NTMap|\!)(\!|b|\!) \circ_{A\mathfrak{B}} \mathfrak{N}(\!|NTMap|\!)(\!|b|\!)) \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) =$
$\mathfrak{H}(\!|ArrMap|\!)(\!|f|\!) \circ_{A\mathfrak{B}} (\mathfrak{M}(\!|NTMap|\!)(\!|a|\!) \circ_{A\mathfrak{B}} \mathfrak{N}(\!|NTMap|\!)(\!|a|\!))$
⟨*proof*⟩

lemma *ntcf-vcomp-is-ntcf*[*cat-cs-intros*]:
— see Chapter II-4 in [7].

assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{M} \cdot_{NTCF} \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-assoc*[*cat-cs-simps*]:
— See Chapter II-4 in [7].
assumes $\mathfrak{L} : \mathfrak{H} \mapsto_{CF} \mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $(\mathfrak{L} \cdot_{NTCF} \mathfrak{M}) \cdot_{NTCF} \mathfrak{N} = \mathfrak{L} \cdot_{NTCF} (\mathfrak{M} \cdot_{NTCF} \mathfrak{N})$
⟨*proof*⟩

6.4.4 The opposite of the vertical composition of natural transformations

lemma *op-ntcf-ntcf-vcomp*[*cat-op-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *op-ntcf* $(\mathfrak{M} \cdot_{NTCF} \mathfrak{N})$ = *op-ntcf* $\mathfrak{N} \cdot_{NTCF}$ *op-ntcf* $\mathfrak{M}$
⟨*proof*⟩

6.5 Horizontal composition of natural transformations

6.5.1 Definition and elementary properties

See Chapter II-5 in [7].

abbreviation (*input*) *ntcf-hcomp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{NTCF}$› *55*)
where *ntcf-hcomp* ≡ *ntsmcf-hcomp*

lemmas [*cat-cs-simps*] = *ntsmcf-hcomp-components*(*2–5*)

Slicing.

lemma *ntcf-ntsmcf-ntcf-hcomp*[*slicing-commute*]:
ntcf-ntsmcf $\mathfrak{M} \circ_{NTSMCF}$ *ntcf-ntsmcf* $\mathfrak{N}$ = *ntcf-ntsmcf* $(\mathfrak{M} \circ_{NTCF} \mathfrak{N})$
⟨*proof*⟩

6.5.2 Natural transformation map

lemma *ntcf-hcomp-NTMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ\,((\mathfrak{M} \circ_{NTCF} \mathfrak{N})(\lfloor NTMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *ntcf-hcomp-NTMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $(\mathfrak{M} \circ_{NTCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor a \rfloor) =$
$\mathfrak{G}'(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) \rfloor) \circ_{A\mathfrak{C}} \mathfrak{M}(\lfloor NTMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \rfloor)$
⟨*proof*⟩

lemma *ntcf-hcomp-NTMap-vrange*:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ\,((\mathfrak{M} \circ_{NTCF} \mathfrak{N})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

6.5.3 Further properties

lemma *ntcf-hcomp-composable-commute*:

— See Chapter II-5 in [7].
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $f : a \mapsto_{\mathfrak{A}} b$
shows
$(\mathfrak{M} \circ_{NTCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{C}} (\mathfrak{F}' \circ_{CF} \mathfrak{F})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) =$
$(\mathfrak{G}' \circ_{CF} \mathfrak{G})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{C}} (\mathfrak{M} \circ_{NTCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
(**is** $\langle ?\mathfrak{M}\mathfrak{N}b \circ_{A\mathfrak{C}} ?\mathfrak{F}'\mathfrak{F}f = ?\mathfrak{G}'\mathfrak{G}f \circ_{A\mathfrak{C}} ?\mathfrak{M}\mathfrak{N}a \rangle$)
⟨*proof*⟩

lemma *ntcf-hcomp-is-ntcf*:
— See Chapter II-5 in [7].
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{M} \circ_{NTCF} \mathfrak{N} : \mathfrak{F}' \circ_{CF} \mathfrak{F} \mapsto_{CF} \mathfrak{G}' \circ_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-hcomp-is-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} = \mathfrak{F}' \circ_{CF} \mathfrak{F}$
and $\mathfrak{S}' = \mathfrak{G}' \circ_{CF} \mathfrak{G}$
shows $\mathfrak{M} \circ_{NTCF} \mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-hcomp-associativ*[*cat-cs-simps*]:
assumes $\mathfrak{L} : \mathfrak{F}'' \mapsto_{CF} \mathfrak{G}'' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $(\mathfrak{L} \circ_{NTCF} \mathfrak{M}) \circ_{NTCF} \mathfrak{N} = \mathfrak{L} \circ_{NTCF} (\mathfrak{M} \circ_{NTCF} \mathfrak{N})$
⟨*proof*⟩

6.5.4 The opposite of the horizontal composition of natural transformations

lemma *op-ntcf-ntcf-hcomp*[*cat-op-simps*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *op-ntcf* $(\mathfrak{M} \circ_{NTCF} \mathfrak{N})$ = *op-ntcf* $\mathfrak{M} \circ_{NTCF}$ *op-ntcf* $\mathfrak{N}$
⟨*proof*⟩

6.6 Interchange law

lemma *ntcf-comp-interchange-law*:
— See Chapter II-5 in [7].
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{M}' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $((\mathfrak{M}' \cdot_{NTCF} \mathfrak{N}') \circ_{NTCF} (\mathfrak{M} \cdot_{NTCF} \mathfrak{N})) = (\mathfrak{M}' \circ_{NTCF} \mathfrak{M}) \cdot_{NTCF} (\mathfrak{N}' \circ_{NTCF} \mathfrak{N})$
⟨*proof*⟩

6.7 Identity natural transformation

6.7.1 Definition and elementary properties

See Chapter II-4 in [7].

definition *ntcf-id* :: $V \Rightarrow V$
where *ntcf-id* $\mathfrak{F} = [\mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor CId \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap \rfloor), \mathfrak{F}, \mathfrak{F}, \mathfrak{F}(\lfloor HomDom \rfloor), \mathfrak{F}(\lfloor HomCod \rfloor)]_{\circ}$

Components.

lemma *ntcf-id-components*:
shows *ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor) = \mathfrak{F}(\lfloor HomCod \rfloor)(\lfloor CId \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap \rfloor)$
and [*dg-shared-cs-simps*, *cat-cs-simps*]: *ntcf-id* $\mathfrak{F}(\lfloor NTDom \rfloor) = \mathfrak{F}$
and [*dg-shared-cs-simps*, *cat-cs-simps*]: *ntcf-id* $\mathfrak{F}(\lfloor NTCod \rfloor) = \mathfrak{F}$
and [*dg-shared-cs-simps*, *cat-cs-simps*]: *ntcf-id* $\mathfrak{F}(\lfloor NTDGDom \rfloor) = \mathfrak{F}(\lfloor HomDom \rfloor)$
and [*dg-shared-cs-simps*, *cat-cs-simps*]: *ntcf-id* $\mathfrak{F}(\lfloor NTDGCod \rfloor) = \mathfrak{F}(\lfloor HomCod \rfloor)$
⟨*proof*⟩

lemma (**in** *is-functor*) *is-functor-ntcf-id-components*:
shows *ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor) = \mathfrak{B}(\lfloor CId \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap \rfloor)$
and *ntcf-id* $\mathfrak{F}(\lfloor NTDom \rfloor) = \mathfrak{F}$
and *ntcf-id* $\mathfrak{F}(\lfloor NTCod \rfloor) = \mathfrak{F}$
and *ntcf-id* $\mathfrak{F}(\lfloor NTDGDom \rfloor) = \mathfrak{A}$
and *ntcf-id* $\mathfrak{F}(\lfloor NTDGCod \rfloor) = \mathfrak{B}$
⟨*proof*⟩

6.7.2 Natural transformation map

lemma (**in** *is-functor*) *ntcf-id-NTMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_{\circ}$ (*ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor)$) = $\mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.ntcf-id-NTMap-vdomain*

lemma (**in** *is-functor*) *ntcf-id-NTMap-app-vdomain*[*cat-cs-simps*]:
assumes [*simp*]: $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
shows *ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) = \mathfrak{B}(\lfloor CId \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \rfloor)$
⟨*proof*⟩

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

lemma (**in** *is-functor*) *ntcf-id-NTMap-vsv*[*cat-cs-intros*]:
vsv (*ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor)$)
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.ntcf-id-NTMap-vsv*

lemma (**in** *is-functor*) *ntcf-id-NTMap-vrange*:
$\mathcal{R}_{\circ}$ (*ntcf-id* $\mathfrak{F}(\lfloor NTMap \rfloor)$) $\subseteq_{\circ}$ $\mathfrak{B}(\lfloor Arr \rfloor)$
⟨*proof*⟩

6.7.3 Further properties

lemma (**in** *is-functor*) *cf-ntcf-id-is-ntcf*[*cat-cs-intros*]:
ntcf-id $\mathfrak{F} : \mathfrak{F} \mapsto_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ntcf-id-is-ntcf'*:
assumes $\mathfrak{G}' = \mathfrak{F}$ **and** $\mathfrak{H}' = \mathfrak{F}$
shows *ntcf-id* $\mathfrak{F} : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-ntcf-id-is-ntcf'*

lemma (**in** *is-ntcf*) *ntcf-ntcf-vcomp-ntcf-id-left-left*[*cat-cs-simps*]:
— See Chapter II-4 in [7].
ntcf-id $\mathfrak{G} \cdot_{NTCF} \mathfrak{N} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-ntcf-vcomp-ntcf-id-left-left*

lemma (**in** *is-ntcf*) *ntcf-ntcf-vcomp-ntcf-id-right-left*[*cat-cs-simps*]:
— See Chapter II-4 in [7].
$\mathfrak{N} \bullet_{NTCF}$ *ntcf-id* $\mathfrak{F} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-ntcf-vcomp-ntcf-id-right-left*

lemma (**in** *is-ntcf*) *ntcf-ntcf-hcomp-ntcf-id-left-left*[*cat-cs-simps*]:
— See Chapter II-5 in [7].
ntcf-id (*cf-id* $\mathfrak{B}$) $\circ_{NTCF}$ $\mathfrak{N} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-ntcf-hcomp-ntcf-id-left-left*

lemma (**in** *is-ntcf*) *ntcf-ntcf-hcomp-ntcf-id-right-left*[*cat-cs-simps*]:
— See Chapter II-5 in [7].
$\mathfrak{N} \circ_{NTCF}$ *ntcf-id* (*cf-id* $\mathfrak{A}$) = $\mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-ntcf-hcomp-ntcf-id-right-left*

6.7.4 The opposite identity natural transformation

lemma (**in** *is-functor*) *cf-ntcf-id-op-cf*: *ntcf-id* (*op-cf* $\mathfrak{F}$) = *op-ntcf* (*ntcf-id* $\mathfrak{F}$)
⟨*proof*⟩

6.7.5 Identity natural transformation of a composition of functors

lemma *ntcf-id-cf-comp*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-id* ($\mathfrak{G} \circ_{CF} \mathfrak{F}$) = *ntcf-id* $\mathfrak{G} \circ_{NTCF}$ *ntcf-id* $\mathfrak{F}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *ntcf-id-cf-comp*[*symmetric*]

6.8 Composition of a natural transformation and a functor

6.8.1 Definition and elementary properties

abbreviation (*input*) *ntcf-cf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{NTCF-CF}$› 55)
where *ntcf-cf-comp* ≡ *tdghm-dghm-comp*

Slicing.

lemma *ntsmcf-tdghm-ntsmcf-smcf-comp*[*slicing-commute*]:
ntcf-ntsmcf $\mathfrak{N} \circ_{NTSMCF-SMCF}$ *cf-smcf* $\mathfrak{H}$ = *ntcf-ntsmcf* ($\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H}$)
⟨*proof*⟩

6.8.2 Natural transformation map

mk-VLambda (**in** *is-functor*)
tdghm-dghm-comp-components(*1*)[**where** $\mathfrak{H}$=$\mathfrak{F}$, *unfolded cf-HomDom*]
|*vdomain ntcf-cf-comp-NTMap-vdomain*[*cat-cs-simps*]|
|*app ntcf-cf-comp-NTMap-app*[*cat-cs-simps*]|

lemmas [*cat-cs-simps*] =
is-functor.ntcf-cf-comp-NTMap-vdomain

is-functor.ntcf-cf-comp-NTMap-app

lemma *ntcf-cf-comp-NTMap-vrange*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H})(\!|NTMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

6.8.3 Opposite of the composition of a natural transformation and a functor

lemma *op-ntcf-ntcf-cf-comp*[*cat-op-simps*]:
op-ntcf $(\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H})$ = *op-ntcf* $\mathfrak{N} \circ_{NTCF-CF}$ *op-cf* $\mathfrak{H}$
⟨*proof*⟩

6.8.4 Composition of a natural transformation and a functor is a natural transformation

lemma *ntcf-cf-comp-is-ntcf*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F} \circ_{CF} \mathfrak{H} \mapsto_{CF} \mathfrak{G} \circ_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-cf-comp-is-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{CF} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{CF} \mathfrak{H}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

6.8.5 Further properties

lemma *ntcf-cf-comp-ntcf-cf-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{H} \mapsto_{CF} \mathfrak{H}' : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $(\mathfrak{N} \circ_{NTCF-CF} \mathfrak{G}) \circ_{NTCF-CF} \mathfrak{F} = \mathfrak{N} \circ_{NTCF-CF} (\mathfrak{G} \circ_{CF} \mathfrak{F})$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-ntcf-cf-comp-cf-id*[*cat-cs-simps*]:
$\mathfrak{N} \circ_{NTCF-CF}$ *cf-id* $\mathfrak{A} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-ntcf-cf-comp-cf-id*

lemma *ntcf-vcomp-ntcf-cf-comp*[*cat-cs-simps*]:
assumes $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $(\mathfrak{M} \circ_{NTCF-CF} \mathfrak{K}) \cdot_{NTCF} (\mathfrak{N} \circ_{NTCF-CF} \mathfrak{K}) = (\mathfrak{M} \cdot_{NTCF} \mathfrak{N}) \circ_{NTCF-CF} \mathfrak{K}$
⟨*proof*⟩

6.9 Composition of a functor and a natural transformation

6.9.1 Definition and elementary properties

abbreviation (*input*) *cf-ntcf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\circ_{CF-NTCF}\rangle$ *55*)
where *cf-ntcf-comp* ≡ *dghm-tdghm-comp*

Slicing.

lemma *ntcf-ntsmcf-cf-ntcf-comp*[*slicing-commute*]:
cf-smcf $\mathfrak{H}$ $\circ_{SMCF-NTSMCF}$ *ntcf-ntsmcf* $\mathfrak{N}$ = *ntcf-ntsmcf* ($\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$)
⟨*proof*⟩

6.9.2 Natural transformation map

mk-VLambda (**in** *is-ntcf*)
dghm-tdghm-comp-components(*1*)[**where** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded ntcf-NTDGDom*]
|*vdomain cf-ntcf-comp-NTMap-vdomain*|
|*app cf-ntcf-comp-NTMap-app*|

lemmas [*cat-cs-simps*] =
is-ntcf.cf-ntcf-comp-NTMap-vdomain
is-ntcf.cf-ntcf-comp-NTMap-app

lemma *cf-ntcf-comp-NTMap-vrange*:
assumes $\mathfrak{H}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$ **and** $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
shows $\mathcal{R}_\circ$ (($\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$)(⟦*NTMap*⟧)) $\subseteq_\circ$ $\mathfrak{C}$(⟦*Arr*⟧)
⟨*proof*⟩

6.9.3 Opposite of the composition of a functor and a natural transformation

lemma *op-ntcf-cf-ntcf-comp*[*cat-op-simps*]:
op-ntcf ($\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$) = *op-cf* $\mathfrak{H}$ $\circ_{CF-NTCF}$ *op-ntcf* $\mathfrak{N}$
⟨*proof*⟩

6.9.4 Composition of a functor and a natural transformation is a natural transformation

lemma *cf-ntcf-comp-is-ntcf*:
assumes $\mathfrak{H}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$ **and** $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
shows $\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$: $\mathfrak{H}$ $\circ_{CF}$ $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{H}$ $\circ_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{H}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
and $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
and $\mathfrak{F}'$ = $\mathfrak{H}$ $\circ_{CF}$ $\mathfrak{F}$
and $\mathfrak{G}'$ = $\mathfrak{H}$ $\circ_{CF}$ $\mathfrak{G}$
shows $\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$: $\mathfrak{F}'$ $\mapsto_{CF}$ $\mathfrak{G}'$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

6.9.5 Further properties

lemma *cf-comp-cf-ntcf-comp-assoc*:
assumes $\mathfrak{N}$: $\mathfrak{H}$ $\mapsto_{CF}$ $\mathfrak{H}'$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
and $\mathfrak{F}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
and $\mathfrak{G}$: $\mathfrak{C}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{D}$
shows ($\mathfrak{G}$ $\circ_{CF}$ $\mathfrak{F}$) $\circ_{CF-NTCF}$ $\mathfrak{N}$ = $\mathfrak{G}$ $\circ_{CF-NTCF}$ ($\mathfrak{F}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$)
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-cf-ntcf-comp-cf-id*[*cat-cs-simps*]:
cf-id $\mathfrak{B}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$ = $\mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.ntcf-cf-ntcf-comp-cf-id*

lemma *cf-ntcf-comp-ntcf-cf-comp-assoc*:
assumes $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$

and $\mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $(\mathfrak{H} \circ_{CF-NTCF} \mathfrak{N}) \circ_{NTCF-CF} \mathfrak{K} = \mathfrak{H} \circ_{CF-NTCF} (\mathfrak{N} \circ_{NTCF-CF} \mathfrak{K})$
⟨*proof*⟩

lemma *ntcf-cf-comp-ntcf-id*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-id* $\mathfrak{F} \circ_{NTCF-CF} \mathfrak{K}$ = *ntcf-id* $\mathfrak{F} \circ_{NTCF}$ *ntcf-id* $\mathfrak{K}$
⟨*proof*⟩

lemma *cf-ntcf-comp-ntcf-vcomp*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{K} \circ_{CF-NTCF} (\mathfrak{M} \bullet_{NTCF} \mathfrak{N}) = (\mathfrak{K} \circ_{CF-NTCF} \mathfrak{M}) \bullet_{NTCF} (\mathfrak{K} \circ_{CF-NTCF} \mathfrak{N})$
⟨*proof*⟩

6.10 Constant natural transformation

6.10.1 Definition and elementary properties

See Chapter III in [7].

definition *ntcf-const* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f =
[
vconst-on $(\mathfrak{J}(\!|Obj|\!))$ f,
cf-const $\mathfrak{J}$ $\mathfrak{C}$ $(\mathfrak{C}(\!|Dom|\!)(\!|f|\!))$,
cf-const $\mathfrak{J}$ $\mathfrak{C}$ $(\mathfrak{C}(\!|Cod|\!)(\!|f|\!))$,
$\mathfrak{J}$,
$\mathfrak{C}$
]$_\circ$

Components.

lemma *ntcf-const-components*:
shows *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTMap|\!)$ = *vconst-on* $(\mathfrak{J}(\!|Obj|\!))$ f
and *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTDom|\!)$ = *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ $(\mathfrak{C}(\!|Dom|\!)(\!|f|\!))$
and *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTCod|\!)$ = *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ $(\mathfrak{C}(\!|Cod|\!)(\!|f|\!))$
and *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTDGDom|\!)$ = $\mathfrak{J}$
and *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTDGCod|\!)$ = $\mathfrak{C}$
⟨*proof*⟩

6.10.2 Natural transformation map

mk-VLambda *ntcf-const-components*(*1*)[*folded VLambda-vconst-on*]
|*vsv ntcf-const-ObjMap-vsv*[*cat-cs-intros*]|
|*vdomain ntcf-const-ObjMap-vdomain*[*cat-cs-simps*]|
|*app ntcf-const-ObjMap-app*[*cat-cs-simps*]|

lemma *ntcf-const-NTMap-ne-vrange*:
assumes $\mathfrak{J}(\!|Obj|\!) \neq 0$
shows $\mathcal{R}_\circ$ (*ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTMap|\!)$) = *set* $\{f\}$
⟨*proof*⟩

lemma *ntcf-const-NTMap-vempty-vrange*:
assumes $\mathfrak{J}(\!|Obj|\!) = 0$
shows $\mathcal{R}_\circ$ (*ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f(\!|NTMap|\!)$) = *0*
⟨*proof*⟩

6.10.3 Constant natural transformation is a natural transformation

lemma *ntcf-const-is-ntcf*:
assumes *category* α $\mathfrak{J}$ **and** *category* α $\mathfrak{C}$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f : *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ $a \mapsto_{CF}$ *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ $b : \mathfrak{J} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-const-is-ntcf'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{J}$
and *category* α $\mathfrak{C}$
and $f : a \mapsto_{\mathfrak{C}} b$
and $\mathfrak{A}$ = *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ a
and $\mathfrak{B}$ = *cf-const* $\mathfrak{J}$ $\mathfrak{C}$ b
and $\mathfrak{J}' = \mathfrak{J}$
and $\mathfrak{C}' = \mathfrak{C}$
shows *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ $f : \mathfrak{A} \mapsto_{CF} \mathfrak{B} : \mathfrak{J}' \mapsto\mapsto_{C\alpha} \mathfrak{C}'$
⟨*proof*⟩

6.10.4 Opposite constant natural transformation

lemma *op-ntcf-ntcf-const*[*cat-op-simps*]:
op-ntcf (*ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f) = *ntcf-const* (*op-cat* $\mathfrak{J}$) (*op-cat* $\mathfrak{C}$) f
⟨*proof*⟩

6.10.5 Further properties

lemma *ntcf-const-ntcf-vcomp*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{J}$
and *category* α $\mathfrak{C}$
and $g : b \mapsto_{\mathfrak{C}} c$
and $f : a \mapsto_{\mathfrak{C}} b$
shows *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ g $\bullet_{NTCF}$ *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f = *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ ($g \circ_{A\mathfrak{C}} f$)
⟨*proof*⟩

lemma *ntcf-id-cf-const*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{J}$ **and** *category* α $\mathfrak{C}$ **and** $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-id* (*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ c) = *ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ ($\mathfrak{C}(\!|CId|\!)(\!|c|\!)$)
⟨*proof*⟩

lemma *ntcf-cf-comp-cf-const-right*[*cat-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *category* α $\mathfrak{A}$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{N} \circ_{NTCF-CF}$ *cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b = *ntcf-const* $\mathfrak{A}$ $\mathfrak{C}$ ($\mathfrak{N}(\!|NTMap|\!)(\!|b|\!)$)
⟨*proof*⟩

lemma *cf-ntcf-comp-ntcf-id*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF-NTCF}$ *ntcf-id* $\mathfrak{F}$ = *ntcf-id* $\mathfrak{G}$ $\circ_{NTCF}$ *ntcf-id* $\mathfrak{F}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ntcf-cf-comp-ntcf-const*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows *ntcf-const* $\mathfrak{B}$ $\mathfrak{C}$ f $\circ_{NTCF-CF}$ $\mathfrak{F}$ = *ntcf-const* $\mathfrak{A}$ $\mathfrak{C}$ f
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-ntcf-cf-comp-ntcf-const*

lemma (**in** *is-functor*) *cf-ntcf-comp-cf-ntcf-const*[*cat-cs-simps*]:

assumes *category* α $\mathfrak{J}$
and $f : r' \mapsto_{\mathfrak{A}} r$
shows $\mathfrak{F} \circ_{CF-NTCF}$ *ntcf-const* $\mathfrak{J}$ $\mathfrak{A}$ f = *ntcf-const* $\mathfrak{J}$ $\mathfrak{B}$ ($\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-ntcf-comp-cf-ntcf-const*

6.11 Natural isomorphism

See Chapter I-4 in [7].

locale *is-iso-ntcf* = *is-ntcf* +
assumes *iso-ntcf-is-iso-arr*[*cat-arrow-cs-intros*]:
$a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{iso\mathfrak{B}} \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$

syntax *-is-iso-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- : - ↦$_{CF.iso}$ - : - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-iso-ntcf* ⇌ *is-iso-ntcf*
translations $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ ⇌
CONST is-iso-ntcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

lemma (**in** *is-iso-ntcf*) *iso-ntcf-is-iso-arr'*:
assumes $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
and $A = \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $B = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
shows $\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) : A \mapsto_{iso\mathfrak{B}} B$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] =
is-iso-ntcf.iso-ntcf-is-iso-arr'

lemma (**in** *is-iso-ntcf*) *iso-ntcf-is-iso-arr''*:
assumes $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
and $A = \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $B = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $F = \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
and $\mathfrak{B}' = \mathfrak{B}$
shows $F : A \mapsto_{iso\mathfrak{B}'} B$
⟨*proof*⟩

Rules.

lemma (**in** *is-iso-ntcf*) *is-iso-ntcf-axioms'*[*cat-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{G}$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-iso-ntcf-def*[*unfolded is-iso-ntcf-axioms-def*]
|*intro is-iso-ntcfI*|
|*dest is-iso-ntcfD*[*dest*]|
|*elim is-iso-ntcfE*[*elim*]|

lemmas [*ntcf-cs-intros*] = *is-iso-ntcfD*(*1*)

6.12 Inverse natural transformation

6.12.1 Definition and elementary properties

definition *inv-ntcf* :: $V \Rightarrow V$
where *inv-ntcf* $\mathfrak{N}$ =

[
(λa∈∘ℜ(|NTDGDom|)(|Obj|). SOME g. is-inverse (ℜ(|NTDGCod|)) g (ℜ(|NTMap|)(|a|))),
ℜ(|NTCod|),
ℜ(|NTDom|),
ℜ(|NTDGDom|),
ℜ(|NTDGCod|)
]∘

Slicing.

lemma *inv-ntcf-components*:
shows *inv-ntcf* ℜ(|*NTMap*|) =
(λa∈∘ℜ(|*NTDGDom*|)(|*Obj*|). *SOME g. is-inverse* (ℜ(|*NTDGCod*|)) *g* (ℜ(|*NTMap*|)(|*a*|)))
and [*cat-cs-simps*]: *inv-ntcf* ℜ(|*NTDom*|) = ℜ(|*NTCod*|)
and [*cat-cs-simps*]: *inv-ntcf* ℜ(|*NTCod*|) = ℜ(|*NTDom*|)
and [*cat-cs-simps*]: *inv-ntcf* ℜ(|*NTDGDom*|) = ℜ(|*NTDGDom*|)
and [*cat-cs-simps*]: *inv-ntcf* ℜ(|*NTDGCod*|) = ℜ(|*NTDGCod*|)
⟨*proof*⟩

Components.

lemma (in *is-iso-ntcf*) *is-iso-ntcf-inv-ntcf-components*[*cat-cs-simps*]:
inv-ntcf ℜ(|*NTDom*|) = 𝔊
inv-ntcf ℜ(|*NTCod*|) = 𝔉
inv-ntcf ℜ(|*NTDGDom*|) = 𝔄
inv-ntcf ℜ(|*NTDGCod*|) = 𝔅
⟨*proof*⟩

6.12.2 Natural transformation map

lemma *inv-ntcf-NTMap-vsv*[*cat-cs-intros*]: *vsv* (*inv-ntcf* ℜ(|*NTMap*|))
⟨*proof*⟩

lemma (in *is-iso-ntcf*) *iso-ntcf-inv-ntcf-NTMap-app-is-inverse*[*cat-cs-intros*]:
assumes *a* ∈∘ 𝔄(|*Obj*|)
shows *is-inverse* 𝔅 (*inv-ntcf* ℜ(|*NTMap*|)(|*a*|)) (ℜ(|*NTMap*|)(|*a*|))
⟨*proof*⟩

lemma (in *is-iso-ntcf*) *iso-ntcf-inv-ntcf-NTMap-app-is-the-inverse*[*cat-cs-intros*]:
assumes *a* ∈∘ 𝔄(|*Obj*|)
shows *inv-ntcf* ℜ(|*NTMap*|)(|*a*|) = (ℜ(|*NTMap*|)(|*a*|))$^{-1}{}_{C\mathfrak{B}}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-iso-ntcf.iso-ntcf-inv-ntcf-NTMap-app-is-the-inverse*

lemma (in *is-ntcf*) *inv-ntcf-NTMap-vdomain*[*cat-cs-simps*]:
𝒟∘ (*inv-ntcf* ℜ(|*NTMap*|)) = 𝔄(|*Obj*|)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-ntcf.inv-ntcf-NTMap-vdomain*

lemma (in *is-iso-ntcf*) *inv-ntcf-NTMap-vrange*:
ℛ∘ (*inv-ntcf* ℜ(|*NTMap*|)) ⊆∘ 𝔅(|*Arr*|)
⟨*proof*⟩

6.12.3 Opposite natural isomorphism

lemma (in *is-iso-ntcf*) *is-iso-ntcf-op*:
op-ntcf ℜ : *op-cf* 𝔊 ↦$_{CF.iso}$ *op-cf* 𝔉 : *op-cat* 𝔄 ↦↦$_{C\alpha}$ *op-cat* 𝔅
⟨*proof*⟩

lemma (in *is-iso-ntcf*) *is-iso-ntcf-op'*[*cat-op-intros*]:
  assumes $\mathfrak{G}' = op\text{-}cf\ \mathfrak{G}$
    and $\mathfrak{F}' = op\text{-}cf\ \mathfrak{F}$
    and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{A}$
    and $\mathfrak{B}' = op\text{-}cat\ \mathfrak{B}$
  shows $op\text{-}ntcf\ \mathfrak{N} : \mathfrak{G}' \mapsto_{CF.iso} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
  ⟨*proof*⟩

lemmas *is-iso-ntcf-op*[*cat-op-intros*] = *is-iso-ntcf.is-iso-ntcf-op*

6.13 A natural isomorphism is an isomorphism in the category *Funct*

The results that are presented in this subsection can be found in nLab (see [1][2]).

lemma *is-iso-arr-is-iso-ntcf*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{N} \bullet_{NTCF} \mathfrak{M} = ntcf\text{-}id\ \mathfrak{G}$
    and $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} = ntcf\text{-}id\ \mathfrak{F}$
  shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *iso-ntcf-is-iso-arr*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
  shows [*ntcf-cs-intros*]: $inv\text{-}ntcf\ \mathfrak{N} : \mathfrak{G} \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{N} \bullet_{NTCF} inv\text{-}ntcf\ \mathfrak{N} = ntcf\text{-}id\ \mathfrak{G}$
    and $inv\text{-}ntcf\ \mathfrak{N} \bullet_{NTCF} \mathfrak{N} = ntcf\text{-}id\ \mathfrak{F}$
⟨*proof*⟩

6.13.1 The operation of taking the inverse natural transformation is an involution

lemma (in *is-iso-ntcf*) *iso-ntcf-inv-ntcf-inv-ntcf*[*ntcf-cs-simps*]:
  $inv\text{-}ntcf\ (inv\text{-}ntcf\ \mathfrak{N}) = \mathfrak{N}$
⟨*proof*⟩

lemmas [*ntcf-cs-simps*] = *is-iso-ntcf.iso-ntcf-inv-ntcf-inv-ntcf*

6.13.2 Natural isomorphisms from natural transformations

lemma *iso-ntcf-if-is-inverse*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\bigwedge a.\ a \in_{\circ} \mathfrak{A}(\!|Obj|\!) \Longrightarrow is\text{-}inverse\ \mathfrak{B}\ (\mathfrak{M}(\!|NTMap|\!)(\!|a|\!))\ (\mathfrak{N}(\!|NTMap|\!)(\!|a|\!))$
  shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.iso} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{M} = inv\text{-}ntcf\ \mathfrak{N}$
    and $\mathfrak{N} = inv\text{-}ntcf\ \mathfrak{M}$
⟨*proof*⟩

6.13.3 Vertical composition of natural isomorphisms

lemma *ntcf-vcomp-is-iso-ntcf*[*cat-cs-intros*]:
  assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.iso} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
  shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

[2] https://ncatlab.org/nlab/show/natural+isomorphism

6.13.4 Horizontal composition of natural isomorphisms

lemma *ntcf-hcomp-is-iso-ntcf*:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto\mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{M} \circ_{NTCF} \mathfrak{N} : \mathfrak{F}' \circ_{CF} \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G}' \circ_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-hcomp-is-iso-ntcf'*[*ntcf-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto\mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{H}' = \mathfrak{F}' \circ_{CF} \mathfrak{F}$
and $\mathfrak{H}'' = \mathfrak{G}' \circ_{CF} \mathfrak{G}$
shows $\mathfrak{M} \circ_{NTCF} \mathfrak{N} : \mathfrak{H}' \mapsto\mapsto_{CF.iso} \mathfrak{H}'' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

6.13.5 Composition of a natural isomorphism and a functor

lemma *ntcf-cf-comp-is-iso-ntcf*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F} \circ_{CF} \mathfrak{H} \mapsto\mapsto_{CF.iso} \mathfrak{G} \circ_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-cf-comp-is-iso-ntcf'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{CF} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{CF} \mathfrak{H}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F}' \mapsto\mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

6.13.6 An identity natural transformation is a natural isomorphism

lemma (**in** *is-functor*) *cf-ntcf-id-is-iso-ntcf*:
ntcf-id $\mathfrak{F} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ntcf-id-is-iso-ntcf'*[*ntcf-cs-intros*]:
assumes $\mathfrak{G}' = \mathfrak{F}$ **and** $\mathfrak{H}' = \mathfrak{F}$
shows *ntcf-id* $\mathfrak{F} : \mathfrak{G}' \mapsto\mapsto_{CF.iso} \mathfrak{H}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*ntcf-cs-intros*] = *is-functor.cf-ntcf-id-is-iso-ntcf'*

6.14 Functor isomorphism

6.14.1 Definition and elementary properties

See subsection 1.5 in [3].

locale *iso-functor* =
fixes α $\mathfrak{F}$ $\mathfrak{G}$
assumes *iso-cf-is-iso-ntcf*: $\exists \mathfrak{A}\ \mathfrak{B}\ \mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$

notation *iso-functor* (**infixl** $\langle\approx_{CF^1}\rangle$ *50*)

Rules.

lemma *iso-functorI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto\mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$

shows $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
⟨*proof*⟩

lemma *iso-functorD*[*dest*]:
assumes $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
shows $\exists \mathfrak{A}\ \mathfrak{B}\ \mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *iso-functorE*[*elim*]:
assumes $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
obtains $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{N}$ **where** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

6.14.2 A functor isomorphism is an equivalence relation

lemma *iso-functor-refl*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{F} \approx_{CF\alpha} \mathfrak{F}$
⟨*proof*⟩

lemma *iso-functor-sym*[*sym*]:
assumes $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
shows $\mathfrak{G} \approx_{CF\alpha} \mathfrak{F}$
⟨*proof*⟩

lemma *iso-functor-trans*[*trans*, *intro*]:
assumes $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$ **and** $\mathfrak{G} \approx_{CF\alpha} \mathfrak{H}$
shows $\mathfrak{F} \approx_{CF\alpha} \mathfrak{H}$
⟨*proof*⟩

6.14.3 Opposite functor isomorphism

lemma (**in** *iso-functor*) *iso-functor-op*: *op-cf* $\mathfrak{F} \approx_{CF\alpha}$ *op-cf* $\mathfrak{G}$
⟨*proof*⟩

lemmas *iso-functor-op*[*cat-op-intros*] = *iso-functor.iso-functor-op*

7 Smallness for natural transformations

7.1 Natural transformation of functors with tiny maps

7.1.1 Definition and elementary properties

locale *is-tm-ntcf* = *is-ntcf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ +
  assumes *tm-ntcf-is-tm-ntsmcf*: *ntcf-ntsmcf* $\mathfrak{N}$:
    cf-smcf $\mathfrak{F}$ $\mapsto_{SMCF.tm}$ *cf-smcf* $\mathfrak{G}$: *cat-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC.tm\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-tm-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  ($\langle$- :/ - $\mapsto_{CF.tm}$ - :/ - $\mapsto\mapsto_{C.tm1}$ -)$\rangle$ [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-tm-ntcf* $\rightleftharpoons$ *is-tm-ntcf*
translations $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.tm\alpha}$ $\mathfrak{B}$ $\rightleftharpoons$
  CONST is-tm-ntcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

abbreviation *all-tm-ntcfs* :: $V \Rightarrow V$
  where *all-tm-ntcfs* $\alpha \equiv$
    set $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

abbreviation *tm-ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *tm-ntcfs* α $\mathfrak{A}$ $\mathfrak{B} \equiv$
    set $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

abbreviation *these-tm-ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *these-tm-ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G} \equiv$
    set $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

lemma (**in** *is-tm-ntcf*) *tm-ntcf-is-tm-ntsmcf'*:
  assumes $\mathfrak{F}' =$ *cf-smcf* $\mathfrak{F}$
    and $\mathfrak{G}' =$ *cf-smcf* $\mathfrak{G}$
    and $\mathfrak{A}' =$ *cat-smc* $\mathfrak{A}$
    and $\mathfrak{B}' =$ *cat-smc* $\mathfrak{B}$
  shows *ntcf-ntsmcf* $\mathfrak{N}$: $\mathfrak{F}'$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *is-tm-ntcf.tm-ntcf-is-tm-ntsmcf'*

Rules.

lemma (**in** *is-tm-ntcf*) *is-tm-ntcf-axioms'*[*cat-small-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$ **and** $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{G}$
  shows $\mathfrak{N}$: $\mathfrak{F}'$ $\mapsto_{CF.tm}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\mapsto_{C.tm\alpha}$ $\mathfrak{B}'$
  ⟨*proof*⟩

mk-ide rf *is-tm-ntcf-def*[*unfolded is-tm-ntcf-axioms-def*]
  |*intro is-tm-ntcfI*|
  |*dest is-tm-ntcfD*[*dest*]|
  |*elim is-tm-ntcfE*[*elim*]|

lemmas [*cat-small-cs-intros*] = *is-tm-ntcfD*(*1*)

context *is-tm-ntcf*
begin

interpretation *ntsmcf*: *is-tm-ntsmcf*
  α ‹*cat-smc* $\mathfrak{A}$› ‹*cat-smc* $\mathfrak{B}$› ‹*cf-smcf* $\mathfrak{F}$› ‹*cf-smcf* $\mathfrak{G}$› ‹*ntcf-ntsmcf* $\mathfrak{N}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:

tm-ntcf-NTMap-in-Vset = *ntsmcf.tm-ntsmcf-NTMap-in-Vset*

end

sublocale *is-tm-ntcf* ⊆ *NTDom*: *is-tm-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$
⟨*proof*⟩

sublocale *is-tm-ntcf* ⊆ *NTCod*: *is-tm-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$
⟨*proof*⟩

Further rules.

lemma *is-tm-ntcfI'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tm-ntcfD'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-ntcfD'*(*2*,*3*)

lemma *is-tm-ntcfE'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
obtains $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

The set of all natural transformations with tiny maps is small.

lemma *small-all-tm-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-tm-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-these-tm-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

Further elementary results.

lemma *these-tm-ntcfs-iff*:
$\mathfrak{N} \in_{\circ}$ *these-tm-ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G} \longleftrightarrow \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

7.1.2 Opposite natural transformation of functors with tiny maps

lemma (**in** *is-tm-ntcf*) *is-tm-ntcf-op*: *op-ntcf* $\mathfrak{N}$:
op-cf $\mathfrak{G} \mapsto_{CF.tm}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.tm\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tm-ntcf*) *is-tm-ntcf-op'*[*cat-op-intros*]:
assumes $\mathfrak{G}' = op\text{-}cf\ \mathfrak{G}$
and $\mathfrak{F}' = op\text{-}cf\ \mathfrak{F}$
and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{A}$
and $\mathfrak{B}' = op\text{-}cat\ \mathfrak{B}$
shows $op\text{-}ntcf\ \mathfrak{N} : \mathfrak{G}' \mapsto_{CF.tm} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}'$
⟨*proof*⟩

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

7.1.3 Vertical composition of natural transformations of functors with tiny maps

lemma *ntcf-vcomp-is-tm-ntcf*[*cat-small-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.tm} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

7.1.4 Identity natural transformation of a functor with tiny maps

lemma (in *is-tm-functor*) *tm-cf-ntcf-id-is-tm-ntcf*:
ntcf-id $\mathfrak{F} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tm-functor*) *tm-cf-ntcf-id-is-tm-ntcf'*:
assumes $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{F}$
shows *ntcf-id* $\mathfrak{F} : \mathfrak{F}' \mapsto_{CF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functor.tm-cf-ntcf-id-is-tm-ntcf'*

7.1.5 Constant natural transformation of functors with tiny maps

lemma *ntcf-const-is-tm-ntcf*:
assumes *tiny-category* $\alpha\ \mathfrak{J}$ **and** *category* $\alpha\ \mathfrak{C}$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f$:
cf-const $\mathfrak{J}\ \mathfrak{C}\ a \mapsto_{CF.tm}$ *cf-const* $\mathfrak{J}\ \mathfrak{C}\ b : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
(**is** ‹*?Cf* : *?Ca* $\mapsto_{CF.tm}$ *?Cb* : $\mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$›)
⟨*proof*⟩

lemma *ntcf-const-is-tm-ntcf'*[*cat-small-cs-intros*]:
assumes *tiny-category* $\alpha\ \mathfrak{J}$
and *category* $\alpha\ \mathfrak{C}$
and $f : a \mapsto_{\mathfrak{C}} b$
and $\mathfrak{A} = cf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ a$
and $\mathfrak{B} = cf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ b$
and $\mathfrak{J}' = \mathfrak{J}$
and $\mathfrak{C}' = \mathfrak{C}$
shows *ntcf-const* $\mathfrak{J}\ \mathfrak{C}\ f : \mathfrak{A} \mapsto_{CF.tm} \mathfrak{B} : \mathfrak{J}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

7.1.6 Natural isomorphisms of functors with tiny maps

locale *is-tm-iso-ntcf* = *is-iso-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$ + *is-tm-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$
for $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$

syntax *-is-tm-iso-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- : - $\mapsto_{CF.tm.iso}$ - : - $\mapsto\mapsto_{C.tm1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-tm-iso-ntcf* ⇌ *is-tm-iso-ntcf*

translations $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B} \rightleftharpoons$
CONST is-tm-iso-ntcf $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$

Rules.

mk-ide rf *is-tm-iso-ntcf-def*
|*intro is-tm-iso-ntcfI*|
|*dest is-tm-iso-ntcfD*[*dest*]|
|*elim is-tm-iso-ntcfE*[*elim*]|

lemmas [*ntcf-cs-intros*] = *is-tm-iso-ntcfD*

lemma *iso-tm-ntcf-is-iso-arr*:
assumes *category* $\alpha\ \mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows [*ntcf-cs-intros*]: *inv-ntcf* $\mathfrak{N} : \mathfrak{G} \mapsto_{CF.tm.iso} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N} \bullet_{NTCF}$ *inv-ntcf* $\mathfrak{N}$ = *ntcf-id* $\mathfrak{G}$
and *inv-ntcf* $\mathfrak{N} \bullet_{NTCF} \mathfrak{N}$ = *ntcf-id* $\mathfrak{F}$
⟨*proof*⟩

lemma *is-iso-arr-is-tm-iso-ntcf*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.tm} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and [*simp*]: $\mathfrak{N} \bullet_{NTCF} \mathfrak{M}$ = *ntcf-id* $\mathfrak{G}$
and [*simp*]: $\mathfrak{M} \bullet_{NTCF} \mathfrak{N}$ = *ntcf-id* $\mathfrak{F}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

7.1.7 Composition of a natural transformation of functors with tiny maps and a functor with tiny maps

lemma *ntcf-cf-comp-is-tm-ntcf*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F} \circ_{CF} \mathfrak{H} \mapsto_{CF.tm} \mathfrak{G} \circ_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-cf-comp-is-tm-ntcf'*[*cat-small-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{CF} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{CF} \mathfrak{H}$
shows $\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H} : \mathfrak{F}' \mapsto_{CF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

7.1.8 Composition of a functor with tiny maps and a natural transformation of functors with tiny maps

lemma *cf-ntcf-comp-is-tm-ntcf*:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{H} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{H} \circ_{CF} \mathfrak{F} \mapsto_{CF.tm} \mathfrak{H} \circ_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-is-tm-ntcf'*[*cat-small-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{H} \circ_{CF} \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{H} \circ_{CF} \mathfrak{G}$
shows $\mathfrak{H} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{F}' \mapsto_{CF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

7.2 Tiny natural transformation of functors

7.2.1 Definition and elementary properties

locale *is-tiny-ntcf* = *is-ntcf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ +
  assumes *tiny-ntcf-is-tiny-ntsmcf*:
    ntcf-ntsmcf $\mathfrak{N}$:
      cf-smcf $\mathfrak{F}$ $\mapsto_{SMCF.tiny}$ *cf-smcf* $\mathfrak{G}$: *cat-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC.tiny\alpha}$ *cat-smc* $\mathfrak{B}$

syntax *-is-tiny-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  ($\langle$(- :/ - $\mapsto_{CF.tiny}$ - :/ - $\mapsto\mapsto_{C.tiny1}$ -)$\rangle$ [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-tiny-ntcf* $\rightleftharpoons$ *is-tiny-ntcf*
translations $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF.tiny}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.tiny\alpha}$ $\mathfrak{B}$ $\rightleftharpoons$
  CONST is-tiny-ntcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

abbreviation *all-tiny-ntcfs* :: $V \Rightarrow V$
  where *all-tiny-ntcfs* $\alpha \equiv$
    set $\{\mathfrak{N}.\ \exists \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$

abbreviation *tiny-ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *tiny-ntcfs* α $\mathfrak{A}$ $\mathfrak{B} \equiv$
    set $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$

abbreviation *these-tiny-ntcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *these-tiny-ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G} \equiv$
    set $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$

lemma (**in** *is-tiny-ntcf*) *tiny-ntcf-is-tiny-ntsmcf'*:
  assumes $\alpha' = \alpha$
    and $\mathfrak{F}' =$ *cf-smcf* $\mathfrak{F}$
    and $\mathfrak{G}' =$ *cf-smcf* $\mathfrak{G}$
    and $\mathfrak{A}' =$ *cat-smc* $\mathfrak{A}$
    and $\mathfrak{B}' =$ *cat-smc* $\mathfrak{B}$
  shows *ntcf-ntsmcf* $\mathfrak{N}$: $\mathfrak{F}'$ $\mapsto_{SMCF.tiny}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\mapsto_{SMC.tiny_{\alpha'}}$ $\mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*slicing-intros*] = *is-tiny-ntcf.tiny-ntcf-is-tiny-ntsmcf'*

Rules.

lemma (**in** *is-tiny-ntcf*) *is-tiny-ntcf-axioms'*[*cat-small-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$ **and** $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{G}$
  shows $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF.tiny}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C.tiny\alpha}$ $\mathfrak{B}$
  ⟨*proof*⟩

mk-ide rf *is-tiny-ntcf-def*[*unfolded is-tiny-ntcf-axioms-def*]
  |*intro is-tiny-ntcfI*|
  |*dest is-tiny-ntcfD*[*dest*]|
  |*elim is-tiny-ntcfE*[*elim*]|

Elementary properties.

sublocale *is-tiny-ntcf* ⊆ *NTDom*: *is-tiny-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$
  ⟨*proof*⟩

sublocale *is-tiny-ntcf* ⊆ *NTCod*: *is-tiny-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$
  ⟨*proof*⟩

sublocale *is-tiny-ntcf* ⊆ *is-tm-ntcf*
  ⟨*proof*⟩

lemmas (**in** *is-tiny-ntcf*) *tiny-ntcf-is-tm-ntcf*[*cat-small-cs-intros*] =
is-tm-ntcf-axioms

lemmas [*cat-small-cs-intros*] = *is-tiny-ntcf.tiny-ntcf-is-tm-ntcf*

Further rules.

lemma *is-tiny-ntcfI'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-ntcfD'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tiny-ntcfD'*(*2*,*3*)

lemma *is-tiny-ntcfE'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
obtains $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-ntcf-iff*:
$\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \longleftrightarrow$
(
$\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B} \wedge$
$\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \wedge$
$\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
)
⟨*proof*⟩

lemma (**in** *is-tiny-ntcf*) *tiny-ntcf-in-Vset*: $\mathfrak{N} \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma *small-all-tiny-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-tiny-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-these-tiny-ntcfs*[*simp*]:
small $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *tiny-ntcfs-vsubset-Vset*[*simp*]:
set $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}\} \subseteq_{\circ} Vset\ \alpha$
(**is** ‹*set ?ntcfs* $\subseteq_{\circ}$ -›)
⟨*proof*⟩

Further elementary results.

lemma *these-tiny-ntcfs-iff*:
$\mathfrak{N} \in_{\circ} \textit{these-tiny-ntcfs}\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \longleftrightarrow \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

Size.

lemma (**in** *is-ntcf*) *ntcf-is-tiny-ntcf-if-ge-Limit*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\beta} \mathfrak{B}$
⟨*proof*⟩

7.2.2 Opposite natural transformation of tiny functors

lemma (**in** *is-tiny-ntcf*) *is-tm-ntcf-op*: *op-ntcf* $\mathfrak{N}$:
op-cf $\mathfrak{G} \mapsto_{CF.tiny}$ *op-cf* $\mathfrak{F}$: *op-cat* $\mathfrak{A} \mapsto\mapsto_{C.tiny\alpha}$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tiny-ntcf*) *is-tiny-ntcf-op'*[*cat-op-intros*]:
assumes $\mathfrak{G}' =$ *op-cf* $\mathfrak{G}$
and $\mathfrak{F}' =$ *op-cf* $\mathfrak{F}$
and $\mathfrak{A}' =$ *op-cat* $\mathfrak{A}$
and $\mathfrak{B}' =$ *op-cat* $\mathfrak{B}$
shows *op-ntcf* $\mathfrak{N} : \mathfrak{G}' \mapsto_{CF.tiny} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}'$
⟨*proof*⟩

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

7.2.3 Vertical composition of tiny natural transformations

lemma *ntsmcf-vcomp-is-tiny-ntsmcf*[*cat-small-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.tiny} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

7.2.4 Tiny identity natural transformation

lemma (**in** *is-tiny-functor*) *tiny-cf-ntcf-id-is-tiny-ntcf*:
ntcf-id $\mathfrak{F} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tiny-functor*) *tiny-cf-ntcf-id-is-tiny-ntcf'*[*cat-small-cs-intros*]:
assumes $\mathfrak{F}' = \mathfrak{F}$ **and** $\mathfrak{G}' = \mathfrak{F}$
shows *ntcf-id* $\mathfrak{F} : \mathfrak{F}' \mapsto_{CF.tiny} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tiny-functor.tiny-cf-ntcf-id-is-tiny-ntcf'*

7.3 Tiny natural isomorphisms

7.3.1 Definition and elementary properties

locale *is-tiny-iso-ntcf* = *is-iso-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$ + *is-tiny-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$
for $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$

syntax *-is-tiny-iso-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
$(\langle(\text{-} : \text{-} \mapsto_{CF.tiny.iso} \text{-} : \text{-} \mapsto\mapsto_{C.tiny^1} \text{-})\rangle\ [51, 51, 51, 51, 51]\ 51)$
syntax-consts *-is-tiny-iso-ntcf* ⇌ *is-tiny-iso-ntcf*

translations $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \rightleftharpoons$
CONST is-tiny-iso-ntcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

Rules.

mk-ide rf *is-tiny-iso-ntcf-def*
|*intro is-tiny-iso-ntcfI*|
|*dest is-tiny-iso-ntcfD*[*dest*]|
|*elim is-tiny-iso-ntcfE*[*elim*]|

lemmas [*ntcf-cs-intros*] = *is-tiny-iso-ntcfD*(*2*)

Elementary properties.

sublocale *is-tiny-iso-ntcf* ⊆ *is-tm-iso-ntcf*
⟨*proof*⟩

lemmas (**in** *is-tiny-iso-ntcf*) *is-tm-iso-ntcf-axioms'* = *is-tm-iso-ntcf-axioms*

lemmas [*ntcf-cs-intros*] = *is-tiny-iso-ntcf.is-tm-iso-ntcf-axioms'*

Further rules.

lemma *is-tiny-iso-ntcfI'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-iso-ntcfD'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-iso-ntcfE'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
obtains $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-iso-ntcf-iff*:
$\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \longleftrightarrow$
(
$\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B} \wedge$
$\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B} \wedge$
$\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
)
⟨*proof*⟩

7.3.2 Further properties

lemma *iso-tiny-ntcf-is-iso-arr*:
assumes *category* α $\mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
shows [*ntcf-cs-intros*]: *inv-ntcf* $\mathfrak{N} : \mathfrak{G} \mapsto_{CF.tiny.iso} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
and $\mathfrak{N} \bullet_{NTCF}$ *inv-ntcf* $\mathfrak{N}$ = *ntcf-id* $\mathfrak{G}$
and *inv-ntcf* $\mathfrak{N} \bullet_{NTCF} \mathfrak{N}$ = *ntcf-id* $\mathfrak{F}$
⟨*proof*⟩

lemma *is-iso-arr-is-tiny-iso-ntcf*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
    and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.tiny} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
    and [*simp*]: $\mathfrak{N} \bullet_{NTCF} \mathfrak{M} = ntcf\text{-}id\ \mathfrak{G}$
    and [*simp*]: $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} = ntcf\text{-}id\ \mathfrak{F}$
  shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tiny.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

8 Product category

8.1 Background

See Chapter II-3 in [7].

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

8.2 Product category: definition and elementary properties

definition *cat-prod* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  where *cat-prod* I $\mathfrak{A}$ =
    [
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$,
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!)(\!|f(\!|i|\!)|\!)))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!)(\!|f(\!|i|\!)|\!)))$,
      (
        $\lambda gf \in_{\circ} composable\text{-}arrs\ (dg\text{-}prod\ I\ \mathfrak{A})$.
          $(\lambda i \in_{\circ} I.\ vpfst\ gf(\!|i|\!) \circ_{A\mathfrak{A}\ i} vpsnd\ gf(\!|i|\!))$
      ),
      $(\lambda a \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|CId|\!)(\!|a(\!|i|\!)|\!)))$
    ]$_{\circ}$

syntax *-PCATEGORY* :: *pttrn* $\Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  ($\langle$($3\prod_{C}$ -$\in_{\circ}$-./ -)$\rangle$ [0, 0, 10] 10)
syntax-consts *-PCATEGORY* $\rightleftharpoons$ *cat-prod*
translations $\prod_{C} i \in_{\circ} I.\ \mathfrak{A} \rightleftharpoons CONST\ cat\text{-}prod\ I\ (\lambda i.\ \mathfrak{A})$

Components.

lemma *cat-prod-components*:
  shows $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Obj|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$
    and $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Arr|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$
    and $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Dom|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!)(\!|f(\!|i|\!)|\!)))$
    and $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Cod|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!)(\!|f(\!|i|\!)|\!)))$
    and $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Comp|\!) =$
      (
        $\lambda gf \in_{\circ} composable\text{-}arrs\ (dg\text{-}prod\ I\ \mathfrak{A})$.
          $(\lambda i \in_{\circ} I.\ vpfst\ gf(\!|i|\!) \circ_{A\mathfrak{A}\ i} vpsnd\ gf(\!|i|\!))$
)
    and $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|CId|\!) =$
      $(\lambda a \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|CId|\!)(\!|a(\!|i|\!)|\!)))$
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-cat-prod*[*slicing-commute*]:
  smc-prod I $(\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i)) = cat\text{-}smc\ (\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)$
  ⟨*proof*⟩

context
  fixes $\mathfrak{A}$ φ :: $V \Rightarrow V$
    and $\mathfrak{C}$ :: V
begin

lemmas-with [

where $\mathfrak{A}$=‹λi. *cat-smc* ($\mathfrak{A}$ i)›, *unfolded slicing-simps slicing-commute*
]:
cat-prod-ObjI = *smc-prod-ObjI*
and *cat-prod-ObjD* = *smc-prod-ObjD*
and *cat-prod-ObjE* = *smc-prod-ObjE*
and *cat-prod-Obj-cong* = *smc-prod-Obj-cong*
and *cat-prod-ArrI* = *smc-prod-ArrI*
and *cat-prod-ArrD* = *smc-prod-ArrD*
and *cat-prod-ArrE* = *smc-prod-ArrE*
and *cat-prod-Arr-cong* = *smc-prod-Arr-cong*
and *cat-prod-Dom-vsv*[*cat-cs-intros*] = *smc-prod-Dom-vsv*
and *cat-prod-Dom-vdomain*[*cat-cs-simps*] = *smc-prod-Dom-vdomain*
and *cat-prod-Dom-app* = *smc-prod-Dom-app*
and *cat-prod-Dom-app-component-app*[*cat-cs-simps*] =
smc-prod-Dom-app-component-app
and *cat-prod-Cod-vsv*[*cat-cs-intros*] = *smc-prod-Cod-vsv*
and *cat-prod-Cod-app* = *smc-prod-Cod-app*
and *cat-prod-Cod-vdomain*[*cat-cs-simps*] = *smc-prod-Cod-vdomain*
and *cat-prod-Cod-app-component-app*[*cat-cs-simps*] =
smc-prod-Cod-app-component-app
and *cat-prod-Comp* = *smc-prod-Comp*
and *cat-prod-Comp-vdomain*[*cat-cs-simps*] = *smc-prod-Comp-vdomain*
and *cat-prod-Comp-app* = *smc-prod-Comp-app*
and *cat-prod-Comp-app-component*[*cat-cs-simps*] =
smc-prod-Comp-app-component
and *cat-prod-Comp-app-vdomain* = *smc-prod-Comp-app-vdomain*
and *cat-prod-vunion-Obj-in-Obj* = *smc-prod-vunion-Obj-in-Obj*
and *cat-prod-vdiff-vunion-Obj-in-Obj* = *smc-prod-vdiff-vunion-Obj-in-Obj*
and *cat-prod-vunion-Arr-in-Arr* = *smc-prod-vunion-Arr-in-Arr*
and *cat-prod-vdiff-vunion-Arr-in-Arr* = *smc-prod-vdiff-vunion-Arr-in-Arr*

end

8.3 Local assumptions for a product category

locale *pcategory-base* = $\mathcal{Z}$ α **for** α I $\mathfrak{A}$ +
assumes *pcat-categories*: $i \in_\circ I \Longrightarrow$ *category* α ($\mathfrak{A}$ i)
and *pcat-index-in-Vset*[*cat-cs-intros*]: $I \in_\circ$ *Vset* α

lemma (**in** *pcategory-base*) *pcat-categories'*[*cat-prod-cs-intros*]:
assumes $i \in_\circ I$ **and** $\alpha' = \alpha$
shows *category* α' ($\mathfrak{A}$ i)
⟨*proof*⟩

Rules.

lemma (**in** *pcategory-base*) *pcategory-base-axioms'*[*cat-prod-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $I' = I$
shows *pcategory-base* α' I' $\mathfrak{A}$
⟨*proof*⟩

mk-ide rf *pcategory-base-def*[*unfolded pcategory-base-axioms-def*]
|*intro pcategory-baseI*|
|*dest pcategory-baseD*[*dest*]|
|*elim pcategory-baseE*[*elim*]|

lemma *pcategory-base-psemicategory-baseI*:
assumes *psemicategory-base* α I (λi. *cat-smc* ($\mathfrak{A}$ i))
and $\bigwedge i$. $i \in_\circ I \Longrightarrow$ *category* α ($\mathfrak{A}$ i)

shows *pcategory-base* α I $\mathfrak{A}$
⟨*proof*⟩

Product category is a product semicategory.

context *pcategory-base*
begin

lemma *pcat-psemicategory-base*: *psemicategory-base* α I $(\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))$
⟨*proof*⟩

interpretation *psmc*: *psemicategory-base* α I ‹$\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i)$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
pcat-Obj-in-Vset = *psmc.psmc-Obj-in-Vset*
and *pcat-Arr-in-Vset* = *psmc.psmc-Arr-in-Vset*
and *pcat-smc-prod-Obj-in-Vset* = *psmc.psmc-smc-prod-Obj-in-Vset*
and *pcat-smc-prod-Arr-in-Vset* = *psmc.psmc-smc-prod-Arr-in-Vset*
and *cat-prod-Dom-app-in-Obj*[*cat-cs-intros*] = *psmc.smc-prod-Dom-app-in-Obj*
and *cat-prod-Cod-app-in-Obj*[*cat-cs-intros*] = *psmc.smc-prod-Cod-app-in-Obj*
and *cat-prod-is-arrI* = *psmc.smc-prod-is-arrI*
and *cat-prod-is-arrD*[*dest*] = *psmc.smc-prod-is-arrD*
and *cat-prod-is-arrE*[*elim*] = *psmc.smc-prod-is-arrE*

end

lemma *cat-prod-dg-prod-is-arr*:
$g : b \mapsto_{dg\text{-}prod\ I\ \mathfrak{A}}\ c \longleftrightarrow g : b \mapsto_{(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)}\ c$
⟨*proof*⟩

lemma *smc-prod-composable-arrs-dg-prod*:
composable-arrs (*dg-prod* I $\mathfrak{A}$) = *composable-arrs* $(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Elementary properties.

lemma (in *pcategory-base*) *pcat-vsubset-index-pcategory-base*:
assumes $J \subseteq_\circ I$
shows *pcategory-base* α J $\mathfrak{A}$
⟨*proof*⟩

8.3.1 Identity

lemma *cat-prod-CId-vsv*[*cat-cs-intros*]: *vsv* $((\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!))$
⟨*proof*⟩

lemma *cat-prod-CId-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ\ ((\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!)) = (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-prod-CId-app*:
assumes $a \in_\circ (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
shows $(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!)(\!|a|\!) = (\lambda i \in_\circ I.\ \mathfrak{A}\ i(\!|CId|\!)(\!|a(\!|i|\!)|\!))$
⟨*proof*⟩

lemma *cat-prod-CId-app-component*[*cat-cs-simps*]:
assumes $a \in_\circ (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$ **and** $i \in_\circ I$
shows $(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!)(\!|a|\!)(\!|i|\!) = \mathfrak{A}\ i(\!|CId|\!)(\!|a(\!|i|\!)|\!)$
⟨*proof*⟩

lemma (in *pcategory-base*) *cat-prod-CId-vrange*:
$\mathcal{R}_\circ\ ((\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!)) \subseteq_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!))$
⟨*proof*⟩

8.3.2 A product α-category is a tiny β-category

lemma (in *pcategory-base*) *pcat-tiny-category-cat-prod*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *tiny-category* β $(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

8.4 Further local assumptions for product categories

8.4.1 Definition and elementary properties

locale *pcategory* = *pcategory-base* $\alpha\ I\ \mathfrak{A}$ **for** $\alpha\ I\ \mathfrak{A}$ +
assumes *pcat-Obj-vsubset-Vset*: $J \subseteq_\circ I \Longrightarrow (\prod_C i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and *pcat-Hom-vifunion-in-Vset*:
⟦
$J \subseteq_\circ I$;
$A \subseteq_\circ (\prod_C i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$B \subseteq_\circ (\prod_C i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$A \in_\circ Vset\ \alpha$;
$B \in_\circ Vset\ \alpha$
⟧ $\Longrightarrow (\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ (\prod_C i \in_\circ J.\ \mathfrak{A}\ i)\ a\ b) \in_\circ Vset\ \alpha$

Rules.

lemma (in *pcategory*) *pcategory-axioms'*[*cat-prod-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $I' = I$
shows *pcategory* $\alpha'\ I'\ \mathfrak{A}$
⟨*proof*⟩

mk-ide rf *pcategory-def*[*unfolded pcategory-axioms-def*]
|*intro pcategoryI*|
|*dest pcategoryD*[*dest*]|
|*elim pcategoryE*[*elim*]|

lemmas [*cat-prod-cs-intros*] = *pcategoryD*(*1*)

lemma *pcategory-psemicategoryI*:
assumes *psemicategory* $\alpha\ I\ (\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow category\ \alpha\ (\mathfrak{A}\ i)$
shows *pcategory* $\alpha\ I\ \mathfrak{A}$
⟨*proof*⟩

Product category is a product semicategory.

context *pcategory*
begin

lemma *pcat-psemicategory*: *psemicategory* $\alpha\ I\ (\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))$
⟨*proof*⟩

interpretation *psmc*: *psemicategory* $\alpha\ I\ \langle\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i)\rangle$
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
pcat-Obj-vsubset-Vset' = *psmc.psmc-Obj-vsubset-Vset'*

and *pcat-Hom-vifunion-in-Vset′ = psmc.psmc-Hom-vifunion-in-Vset′*
and *pcat-cat-prod-vunion-is-arr = psmc.psmc-smc-prod-vunion-is-arr*
and *pcat-cat-prod-vdiff-vunion-is-arr = psmc.psmc-smc-prod-vdiff-vunion-is-arr*

lemmas-with [*unfolded slicing-simps slicing-commute*]:
pcat-cat-prod-vunion-Comp = psmc.psmc-smc-prod-vunion-Comp
and *pcat-cat-prod-vdiff-vunion-Comp = psmc.psmc-smc-prod-vdiff-vunion-Comp*

end

Elementary properties.

lemma (**in** *pcategory*) *pcat-vsubset-index-pcategory*:
assumes $J \subseteq_{\circ} I$
shows *pcategory* α J $\mathfrak{A}$
⟨*proof*⟩

8.4.2 A product α-category is an α-category

lemma (**in** *pcategory*) *pcat-category-cat-prod*: *category* α $(\prod_{C} i \in_{\circ} I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

8.5 Local assumptions for a finite product category

8.5.1 Definition and elementary properties

locale *finite-pcategory = pcategory-base* α I $\mathfrak{A}$ **for** α I $\mathfrak{A}$ +
assumes *fin-pcat-index-vfinite*: *vfinite I*

Rules.

lemma (**in** *finite-pcategory*) *finite-pcategory-axioms*[*cat-prod-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $I' = I$
shows *finite-pcategory* α' I' $\mathfrak{A}$
⟨*proof*⟩

mk-ide rf *finite-pcategory-def*[*unfolded finite-pcategory-axioms-def*]
|*intro finite-pcategoryI*|
|*dest finite-pcategoryD*[*dest*]|
|*elim finite-pcategoryE*[*elim*]|

lemmas [*cat-prod-cs-intros*] = *finite-pcategoryD*(*1*)

lemma *finite-pcategory-finite-psemicategoryI*:
assumes *finite-psemicategory* α I $(\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))$
and $\bigwedge i.\ i \in_{\circ} I \Longrightarrow category\ \alpha\ (\mathfrak{A}\ i)$
shows *finite-pcategory* α I $\mathfrak{A}$
⟨*proof*⟩

8.5.2 Local assumptions for a finite product semicategory and local assumptions for an arbitrary product semicategory

sublocale *finite-pcategory* $\subseteq$ *pcategory* α I $\mathfrak{A}$
⟨*proof*⟩

8.6 Binary union and complement

lemma (**in** *pcategory*) *pcat-cat-prod-vunion-CId*:
assumes *vdisjnt J K*
and $J \subseteq_{\circ} I$

and $K \subseteq_\circ I$
and $a \in_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
and $b \in_\circ (\prod_C j \in_\circ K.\ \mathfrak{A}\ j)(\!|Obj|\!)$
shows
$(\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|a|\!) \cup_\circ (\prod_C j \in_\circ K.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|b|\!) =$
$(\prod_C i \in_\circ J \cup_\circ K.\ \mathfrak{A}\ i)(\!|CId|\!)(\!|a \cup_\circ b|\!)$
⟨*proof*⟩

lemma (in *pcategory*) *pcat-cat-prod-vdiff-vunion-CId*:
assumes $J \subseteq_\circ I$
and $a \in_\circ (\prod_C j \in_\circ I -_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
and $b \in_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
shows
$(\prod_C j \in_\circ I -_\circ J.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|a|\!) \cup_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|b|\!) =$
$(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|CId|\!)(\!|a \cup_\circ b|\!)$
⟨*proof*⟩

8.7 Projection

8.7.1 Definition and elementary properties

See Chapter II-3 in [7].

definition *cf-proj* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$ ($\langle \pi_C \rangle$)
where $\pi_C\ I\ \mathfrak{A}\ i =$
$[$
$(\lambda a \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ a(\!|i|\!))$,
$(\lambda f \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ f(\!|i|\!))$,
$(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$,
$\mathfrak{A}\ i$
$]_\circ$

Components.

lemma *cf-proj-components*:
shows $\pi_C\ I\ \mathfrak{A}\ i(\!|ObjMap|\!) = (\lambda a \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ a(\!|i|\!))$
and $\pi_C\ I\ \mathfrak{A}\ i(\!|ArrMap|\!) = (\lambda f \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ f(\!|i|\!))$
and $\pi_C\ I\ \mathfrak{A}\ i(\!|HomDom|\!) = (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
and $\pi_C\ I\ \mathfrak{A}\ i(\!|HomCod|\!) = \mathfrak{A}\ i$
⟨*proof*⟩

Slicing

lemma *cf-smcf-cf-proj*[*slicing-commute*]:
$\pi_{SMC}\ I\ (\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))\ i = cf\text{-}smcf\ (\pi_C\ I\ \mathfrak{A}\ i)$
⟨*proof*⟩

context *pcategory*
begin

interpretation *psmc*: *psemicategory* $\alpha\ I\ \langle \lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i) \rangle$
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
pcat-cf-proj-is-semifunctor = *psmc.psmc-smcf-proj-is-semifunctor*

end

8.7.2 Projection functor is a functor

lemma (in *pcategory*) *pcat-cf-proj-is-functor*:

assumes $i \in_\circ I$
shows $\pi_C\ I\ \mathfrak{A}\ i : (\prod_C i \in_\circ I.\ \mathfrak{A}\ i) \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
⟨*proof*⟩

lemma (in *pcategory*) *pcat-cf-proj-is-functor'*:
assumes $i \in_\circ I$ **and** $\mathfrak{C} = (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$ **and** $\mathfrak{D} = \mathfrak{A}\ i$
shows $\pi_C\ I\ \mathfrak{A}\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *pcategory.pcat-cf-proj-is-functor'*

8.8 Category product universal property functor

8.8.1 Definition and elementary properties

The functor that is presented in this section is used in the proof of the universal property of the product category later in this work.

definition *cf-up* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi$ =
[
$(\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ObjMap|\!)(\!|a|\!)))$,
$(\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ArrMap|\!)(\!|f|\!)))$,
$\mathfrak{C}$,
$(\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
]$_\circ$

Components.

lemma *cf-up-components*:
shows *cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!) = (\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ObjMap|\!)(\!|a|\!)))$
and *cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!) = (\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ArrMap|\!)(\!|f|\!)))$
and *cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomDom|\!) = \mathfrak{C}$
and *cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomCod|\!) = (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Slicing.

lemma *smcf-dghm-cf-up*[*slicing-commute*]:
smcf-up I $(\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i))$ $(cat\text{-}smc\ \mathfrak{C})$ $(\lambda i.\ cf\text{-}smcf\ (\varphi\ i))$ =
cf-smcf (*cf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi$)
⟨*proof*⟩

context
fixes $\mathfrak{A}\ \varphi :: V \Rightarrow V$
and $\mathfrak{C} :: V$
begin

lemmas-with
[
where $\mathfrak{A}$=‹$\lambda i.\ cat\text{-}smc\ (\mathfrak{A}\ i)$› **and** φ=‹$\lambda i.\ cf\text{-}smcf\ (\varphi\ i)$› **and** $\mathfrak{C}$ = ‹$cat\text{-}smc\ \mathfrak{C}$›,
unfolded slicing-simps slicing-commute
]:
cf-up-ObjMap-vdomain[*simp*] = *smcf-up-ObjMap-vdomain*
and *cf-up-ObjMap-app* = *smcf-up-ObjMap-app*
and *cf-up-ObjMap-app-vdomain*[*simp*] = *smcf-up-ObjMap-app-vdomain*
and *cf-up-ObjMap-app-component* = *smcf-up-ObjMap-app-component*
and *cf-up-ArrMap-vdomain*[*simp*] = *smcf-up-ArrMap-vdomain*
and *cf-up-ArrMap-app* = *smcf-up-ArrMap-app*
and *cf-up-ArrMap-app-vdomain*[*simp*] = *smcf-up-ArrMap-app-vdomain*

and *cf-up-ArrMap-app-component = smcf-up-ArrMap-app-component*

lemma *cf-up-ObjMap-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|\textit{ObjMap}|\!)) \subseteq_\circ (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|\textit{Obj}|\!)$
⟨*proof*⟩

lemma *cf-up-ObjMap-app-vrange*:
assumes $a \in_\circ \mathfrak{C}(\!|\textit{Obj}|\!)$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|\textit{ObjMap}|\!)(\!|a|\!)) \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|\textit{Obj}|\!))$
⟨*proof*⟩

lemma *cf-up-ArrMap-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|\textit{ArrMap}|\!)) \subseteq_\circ (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)(\!|\textit{Arr}|\!)$
⟨*proof*⟩

lemma *cf-up-ArrMap-app-vrange*:
assumes $a \in_\circ \mathfrak{C}(\!|\textit{Arr}|\!)$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|\textit{ArrMap}|\!)(\!|a|\!)) \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|\textit{Arr}|\!))$
⟨*proof*⟩

end

context *pcategory*
begin

interpretation *psmc*: *psemicategory* $\alpha\ I\ \langle\lambda i.\ \textit{cat-smc}\ (\mathfrak{A}\ i)\rangle$
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
pcat-smcf-comp-smcf-proj-smcf-up = psmc.psmc-Comp-smcf-proj-smcf-up
and *pcat-smcf-up-eq-smcf-proj = psmc.psmc-smcf-up-eq-smcf-proj*

end

8.8.2 Category product universal property functor is a functor

lemma (**in** *pcategory*) *pcat-cf-up-is-functor*:
assumes *category* $\alpha\ \mathfrak{C}$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
shows $\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi : \mathfrak{C} \mapsto\mapsto_{C\alpha} (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

8.8.3 Further properties

lemma (**in** *pcategory*) *pcat-Comp-cf-proj-cf-up*:
assumes *category* $\alpha\ \mathfrak{C}$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{A}\ i$
and $i \in_\circ I$
shows $\varphi\ i = \pi_C\ I\ \mathfrak{A}\ i \circ_{CF} (\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi)$
⟨*proof*⟩

lemma (**in** *pcategory*) *pcat-cf-up-eq-cf-proj*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} (\prod_C i \in_\circ I.\ \mathfrak{A}\ i)$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i = \pi_C\ I\ \mathfrak{A}\ i \circ_{CF} \mathfrak{F}$
shows $\textit{cf-up}\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi = \mathfrak{F}$
⟨*proof*⟩

8.9 Prodfunctor with respect to a fixed argument

A prodfunctor is a functor whose domain is a product category. It is a generalization of the concept of the bifunctor, as presented in Chapter II-3 in [7].

definition *prodfunctor-proj* :: $V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *prodfunctor-proj* $\mathfrak{S}$ I $\mathfrak{A}$ $\mathfrak{D}$ J c =
    [
      $(\lambda b \in_\circ (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!).\ \mathfrak{S}(\!|ObjMap|\!)(\!|b \cup_\circ c|\!))$,
      $(\lambda f \in_\circ (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)(\!|Arr|\!).\ \mathfrak{S}(\!|ArrMap|\!)(\!|f \cup_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|c|\!)|\!))$,
      $(\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)$,
      $\mathfrak{D}$
    ]$_\circ$

syntax *-PPRODFUNCTOR-PROJ* :: $V \Rightarrow$ *pttrn* $\Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V \Rightarrow V$
  $(\langle(\text{-}(3\prod_C \text{-} \in_\circ \text{-} -_\circ \text{-}./\text{-}),\text{-}/'(/\text{-},\text{-}/')')\rangle\ [51,\ 51,\ 51,\ 51,\ 51,\ 51,\ 51]\ 51)$
syntax-consts *-PPRODFUNCTOR-PROJ* $\rightleftharpoons$ *prodfunctor-proj*
translations $\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A},\mathfrak{D}}(-,c) \rightleftharpoons$
  CONST prodfunctor-proj $\mathfrak{S}$ I $(\lambda i.\ \mathfrak{A})$ $\mathfrak{D}$ J c

Components.

lemma *prodfunctor-proj-components*:
  shows $(\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i,\mathfrak{D}}(-,c))(\!|ObjMap|\!) =$
      $(\lambda b \in_\circ (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!).\ \mathfrak{S}(\!|ObjMap|\!)(\!|b \cup_\circ c|\!))$
    and $(\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i,\mathfrak{D}}(-,c))(\!|ArrMap|\!) =$
      $(\lambda f \in_\circ (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)(\!|Arr|\!).\ \mathfrak{S}(\!|ArrMap|\!)(\!|f \cup_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|CId|\!)(\!|c|\!)|\!))$
    and $(\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i,\mathfrak{D}}(-,c))(\!|HomDom|\!) = (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)$
    and $(\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i,\mathfrak{D}}(-,c))(\!|HomCod|\!) = \mathfrak{D}$
  ⟨*proof*⟩

8.9.1 Object map

mk-VLambda *prodfunctor-proj-components*(*1*)
  |*vsv prodfunctor-proj-ObjMap-vsv*[*cat-cs-intros*]|
  |*vdomain prodfunctor-proj-ObjMap-vdomain*[*cat-cs-simps*]|
  |*app prodfunctor-proj-ObjMap-app*[*cat-cs-simps*]|

8.9.2 Arrow map

mk-VLambda *prodfunctor-proj-components*(*2*)
  |*vsv prodfunctor-proj-ArrMap-vsv*[*cat-cs-intros*]|
  |*vdomain prodfunctor-proj-ArrMap-vdomain*[*cat-cs-simps*]|
  |*app prodfunctor-proj-ArrMap-app*[*cat-cs-simps*]|

8.9.3 Prodfunctor with respect to a fixed argument is a functor

lemma (**in** *pcategory*) *pcat-prodfunctor-proj-is-functor*:
  assumes $\mathfrak{S} : (\prod_C i \in_\circ I.\ \mathfrak{A}\ i) \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $c \in_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
    and $J \subseteq_\circ I$
  shows $(\mathfrak{S}_{\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i,\mathfrak{D}}(-,c)) : (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i) \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma (**in** *pcategory*) *pcat-prodfunctor-proj-is-functor'*:
  assumes $\mathfrak{S} : (\prod_C i \in_\circ I.\ \mathfrak{A}\ i) \mapsto\mapsto_{C\alpha} \mathfrak{D}$
    and $c \in_\circ (\prod_C j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
    and $J \subseteq_\circ I$
    and $\mathfrak{A}' = (\prod_C i \in_\circ I -_\circ J.\ \mathfrak{A}\ i)$

and $\mathfrak{B}' = \mathfrak{D}$
shows $(\mathfrak{S}_{\prod_C i \in_\circ I \, -_\circ \, J.\ \mathfrak{A}\ i, \mathfrak{D}(-,c)}) : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *pcategory.pcat-prodfunctor-proj-is-functor'*

8.10 Singleton category

8.10.1 Slicing

context
fixes $\mathfrak{C}$ :: *V*
begin

lemmas-with [**where** $\mathfrak{C}$=‹*cat-smc* $\mathfrak{C}$›, *unfolded slicing-simps slicing-commute*]:
cat-singleton-ObjI = *smc-singleton-ObjI*
and *cat-singleton-ObjE* = *smc-singleton-ObjE*
and *cat-singleton-ArrI* = *smc-singleton-ArrI*
and *cat-singleton-ArrE* = *smc-singleton-ArrE*

end

context *category*
begin

interpretation *smc*: *semicategory* α ‹*cat-smc* $\mathfrak{C}$› ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
cat-finite-psemicategory-cat-singleton =
smc.smc-finite-psemicategory-smc-singleton
and *cat-singleton-is-arrI* = *smc.smc-singleton-is-arrI*
and *cat-singleton-is-arrD* = *smc.smc-singleton-is-arrD*
and *cat-singleton-is-arrE* = *smc.smc-singleton-is-arrE*

end

8.10.2 Identity

lemma *cat-singleton-CId-app*:
assumes *set* $\{\langle j,\ a\rangle\} \in_\circ (\prod_C i \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|Obj|\!)$
shows $(\prod_C i \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|CId|\!)(\!|set\ \{\langle j,\ a\rangle\}|\!) = set\ \{\langle j,\ \mathfrak{C}(\!|CId|\!)(\!|a|\!)\rangle\}$
⟨*proof*⟩

8.10.3 Singleton category is a category

lemma (**in** *category*) *cat-finite-pcategory-cat-singleton*:
assumes $j \in_\circ Vset\ \alpha$
shows *finite-pcategory* α $(set\ \{j\})$ $(\lambda i.\ \mathfrak{C})$
⟨*proof*⟩

lemma (**in** *category*) *cat-category-cat-singleton*:
assumes $j \in_\circ Vset\ \alpha$
shows *category* α $(\prod_C i \in_\circ set\ \{j\}.\ \mathfrak{C})$
⟨*proof*⟩

8.11 Singleton functor

8.11.1 Definition and elementary properties

definition *cf-singleton* :: $V \Rightarrow V \Rightarrow V$
  where *cf-singleton* j $\mathfrak{C}$ =
    [
      $(\lambda a \in_{\circ} \mathfrak{C}(\!|Obj|\!).\ set\ \{\langle j,\ a\rangle\})$,
      $(\lambda f \in_{\circ} \mathfrak{C}(\!|Arr|\!).\ set\ \{\langle j,\ f\rangle\})$,
      $\mathfrak{C}$,
      $(\prod_{C} i \in_{\circ} set\ \{j\}.\ \mathfrak{C})$
    ]$_{\circ}$

Components.

lemma *cf-singleton-components*:
  shows *cf-singleton* j $\mathfrak{C}(\!|ObjMap|\!)$ = $(\lambda a \in_{\circ} \mathfrak{C}(\!|Obj|\!).\ set\ \{\langle j,\ a\rangle\})$
    and *cf-singleton* j $\mathfrak{C}(\!|ArrMap|\!)$ = $(\lambda f \in_{\circ} \mathfrak{C}(\!|Arr|\!).\ set\ \{\langle j,\ f\rangle\})$
    and *cf-singleton* j $\mathfrak{C}(\!|HomDom|\!)$ = $\mathfrak{C}$
    and *cf-singleton* j $\mathfrak{C}(\!|HomCod|\!)$ = $(\prod_{C} i \in_{\circ} set\ \{j\}.\ \mathfrak{C})$
  ⟨*proof*⟩

Slicing.

lemma *cf-smcf-cf-singleton*[*slicing-commute*]:
  smcf-singleton j (*cat-smc* $\mathfrak{C}$)= *cf-smcf* (*cf-singleton* j $\mathfrak{C}$)
  ⟨*proof*⟩

context
  fixes $\mathfrak{C}$ :: V
begin

lemmas-with [**where** $\mathfrak{C}$=‹*cat-smc* $\mathfrak{C}$›, *unfolded slicing-simps slicing-commute*]:
  cf-singleton-ObjMap-vsv[*cat-cs-intros*] = *smcf-singleton-ObjMap-vsv*
  and *cf-singleton-ObjMap-vdomain*[*cat-cs-simps*] = *smcf-singleton-ObjMap-vdomain*
  and *cf-singleton-ObjMap-vrange* = *smcf-singleton-ObjMap-vrange*
  and *cf-singleton-ObjMap-app*[*cat-prod-cs-simps*] = *smcf-singleton-ObjMap-app*
  and *cf-singleton-ArrMap-vsv*[*cat-cs-intros*] = *smcf-singleton-ArrMap-vsv*
  and *cf-singleton-ArrMap-vdomain*[*cat-cs-simps*] = *smcf-singleton-ArrMap-vdomain*
  and *cf-singleton-ArrMap-vrange* = *smcf-singleton-ArrMap-vrange*
  and *cf-singleton-ArrMap-app*[*cat-prod-cs-simps*] = *smcf-singleton-ArrMap-app*

end

8.11.2 Singleton functor is an isomorphism of categories

lemma (**in** *category*) *cat-cf-singleton-is-functor*:
  assumes $j \in_{\circ}$ *Vset* α
  shows *cf-singleton* j $\mathfrak{C}$: $\mathfrak{C} \mapsto\mapsto_{C.iso\alpha} (\prod_{C} i \in_{\circ} set\ \{j\}.\ \mathfrak{C})$
⟨*proof*⟩

8.12 Product of two categories

8.12.1 Definition and elementary properties.

See Chapter II-3 in [7].

definition *cat-prod-2* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\times_C$› *80*)
  where $\mathfrak{A} \times_C \mathfrak{B} \equiv$ *cat-prod* $(2_{\mathbb{N}})$ $(\lambda i.\ if\ i = 0\ then\ \mathfrak{A}\ else\ \mathfrak{B})$

Slicing.

lemma *cat-smc-cat-prod-2*[*slicing-commute*]:
cat-smc $\mathfrak{A}$ $\times_{SMC}$ *cat-smc* $\mathfrak{B}$ = *cat-smc* ($\mathfrak{A}$ $\times_{C}$ $\mathfrak{B}$)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

lemmas-with
[
where $\mathfrak{A}$=‹*cat-smc* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*cat-smc* $\mathfrak{B}$›,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*cat-semicategory* $\mathfrak{B}$.*cat-semicategory*
]:
cat-prod-2-ObjI = *smc-prod-2-ObjI*
and *cat-prod-2-ObjI'*[*cat-prod-cs-intros*] = *smc-prod-2-ObjI'*
and *cat-prod-2-ObjE* = *smc-prod-2-ObjE*
and *cat-prod-2-ArrI* = *smc-prod-2-ArrI*
and *cat-prod-2-ArrI'*[*cat-prod-cs-intros*] = *smc-prod-2-ArrI'*
and *cat-prod-2-ArrE* = *smc-prod-2-ArrE*
and *cat-prod-2-is-arrI* = *smc-prod-2-is-arrI*
and *cat-prod-2-is-arrI'*[*cat-prod-cs-intros*] = *smc-prod-2-is-arrI'*
and *cat-prod-2-is-arrE* = *smc-prod-2-is-arrE*
and *cat-prod-2-Dom-vsv* = *smc-prod-2-Dom-vsv*
and *cat-prod-2-Dom-vdomain*[*cat-cs-simps*] = *smc-prod-2-Dom-vdomain*
and *cat-prod-2-Dom-app*[*cat-prod-cs-simps*] = *smc-prod-2-Dom-app*
and *cat-prod-2-Dom-vrange* = *smc-prod-2-Dom-vrange*
and *cat-prod-2-Cod-vsv* = *smc-prod-2-Cod-vsv*
and *cat-prod-2-Cod-vdomain*[*cat-cs-simps*] = *smc-prod-2-Cod-vdomain*
and *cat-prod-2-Cod-app*[*cat-prod-cs-simps*] = *smc-prod-2-Cod-app*
and *cat-prod-2-Cod-vrange* = *smc-prod-2-Cod-vrange*
and *cat-prod-2-op-cat-cat-Obj*[*cat-op-simps*] = *smc-prod-2-op-smc-smc-Obj*
and *cat-prod-2-cat-op-cat-Obj*[*cat-op-simps*] = *smc-prod-2-smc-op-smc-Obj*
and *cat-prod-2-op-cat-cat-Arr*[*cat-op-simps*] = *smc-prod-2-op-smc-smc-Arr*
and *cat-prod-2-cat-op-cat-Arr*[*cat-op-simps*] = *smc-prod-2-smc-op-smc-Arr*

lemmas-with
[
where $\mathfrak{A}$=‹*cat-smc* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*cat-smc* $\mathfrak{B}$›,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*cat-semicategory* $\mathfrak{B}$.*cat-semicategory*
]:
cat-prod-2-Comp-app[*cat-prod-cs-simps*] = *smc-prod-2-Comp-app*

end

8.12.2 Product of two categories is a category

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *finite-pcategory-cat-prod-2*: *finite-pcategory* α ($2_{\mathbb{N}}$) (*if2* $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

interpretation *finite-pcategory* α ‹$2_{\mathbb{N}}$› ‹*if2* $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemma *category-cat-prod-2*[*cat-cs-intros*]: *category* α ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)
⟨*proof*⟩

end

8.12.3 Identity

lemma *cat-prod-2-CId-vsv*[*cat-cs-intros*]: *vsv* (($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*CId*|))
⟨*proof*⟩

lemma *cat-prod-2-CId-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*CId*|)) = ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*Obj*|)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

interpretation *finite-pcategory* α ‹$2_{\mathbb{N}}$› ‹(λ*i*. *if i = 0 then* $\mathfrak{A}$ *else* $\mathfrak{B}$)›
⟨*proof*⟩

lemma *cat-prod-2-CId-app*[*cat-prod-cs-simps*]:
assumes [*a*, *b*]$_\circ$ $\in_\circ$ ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*Obj*|)
shows ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*CId*|)(|*a*, *b*|)$_\bullet$ = [$\mathfrak{A}$(|*CId*|)(|*a*|), $\mathfrak{B}$(|*CId*|)(|*b*|)]$_\circ$
⟨*proof*⟩

lemma *cat-prod-2-CId-vrange*: $\mathcal{R}_\circ$ (($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*CId*|)) $\subseteq_\circ$ ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*Arr*|)
⟨*proof*⟩

end

8.12.4 Opposite product category

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *op-smc-smc-prod-2*[*smc-op-simps*]:
op-cat ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$) = *op-cat* $\mathfrak{A}$ $\times_C$ *op-cat* $\mathfrak{B}$
⟨*proof*⟩

end

8.12.5 Flip

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *cat-prod-2-Obj-fconverse*[*cat-cs-simps*]:
$((\mathfrak{A} \times_C \mathfrak{B})(\!|Obj|\!))^{-1}_{\bullet} = (\mathfrak{B} \times_C \mathfrak{A})(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-prod-2-Arr-fconverse*[*cat-cs-simps*]:
$((\mathfrak{A} \times_C \mathfrak{B})(\!|Arr|\!))^{-1}_{\bullet} = (\mathfrak{B} \times_C \mathfrak{A})(\!|Arr|\!)$
⟨*proof*⟩

end

8.13 Projections for the product of two categories

8.13.1 Definition and elementary properties

See Chapter II-3 in [7].

definition *cf-proj-fst* :: $V \Rightarrow V \Rightarrow V$ ($\langle\pi_{C.1}\rangle$)
where $\pi_{C.1}$ $\mathfrak{A}$ $\mathfrak{B}$ = *cf-proj* ($2_{\mathbb{N}}$) (λi. *if* i = *0* *then* $\mathfrak{A}$ *else* $\mathfrak{B}$) *0*
definition *cf-proj-snd* :: $V \Rightarrow V \Rightarrow V$ ($\langle\pi_{C.2}\rangle$)
where $\pi_{C.2}$ $\mathfrak{A}$ $\mathfrak{B}$ = *cf-proj* ($2_{\mathbb{N}}$) (λi. *if* i = *0* *then* $\mathfrak{A}$ *else* $\mathfrak{B}$) ($1_{\mathbb{N}}$)

Slicing

lemma *cf-smcf-cf-proj-fst*[*slicing-commute*]:
$\pi_{SMC.1}$ (*cat-smc* $\mathfrak{A}$) (*cat-smc* $\mathfrak{B}$) = *cf-smcf* ($\pi_{C.1}$ $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *cf-smcf-cf-proj-snd*[*slicing-commute*]:
$\pi_{SMC.2}$ (*cat-smc* $\mathfrak{A}$) (*cat-smc* $\mathfrak{B}$) = *cf-smcf* ($\pi_{C.2}$ $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

lemmas-with
[
where $\mathfrak{A}$=⟨*cat-smc* $\mathfrak{A}$⟩ **and** $\mathfrak{B}$=⟨*cat-smc* $\mathfrak{B}$⟩,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*cat-semicategory* $\mathfrak{B}$.*cat-semicategory*
]:
cf-proj-fst-ObjMap-app = *smcf-proj-fst-ObjMap-app*
and *cf-proj-snd-ObjMap-app* = *smcf-proj-snd-ObjMap-app*
and *cf-proj-fst-ArrMap-app* = *smcf-proj-fst-ArrMap-app*

and *cf-proj-snd-ArrMap-app* = *smcf-proj-snd-ArrMap-app*

end

8.13.2 Domain and codomain of a projection of a product of two categories

lemma *cf-proj-fst-HomDom*: $\pi_{C.1}\ \mathfrak{A}\ \mathfrak{B}(\!|HomDom|\!) = \mathfrak{A} \times_C \mathfrak{B}$
⟨*proof*⟩

lemma *cf-proj-fst-HomCod*: $\pi_{C.1}\ \mathfrak{A}\ \mathfrak{B}(\!|HomCod|\!) = \mathfrak{A}$
⟨*proof*⟩

lemma *cf-proj-snd-HomDom*: $\pi_{C.2}\ \mathfrak{A}\ \mathfrak{B}(\!|HomDom|\!) = \mathfrak{A} \times_C \mathfrak{B}$
⟨*proof*⟩

lemma *cf-proj-snd-HomCod*: $\pi_{C.2}\ \mathfrak{A}\ \mathfrak{B}(\!|HomCod|\!) = \mathfrak{B}$
⟨*proof*⟩

8.13.3 Projection of a product of two categories is a functor

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}$
assumes $\mathfrak{A}$: *category* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{B}$: *category* $\alpha\ \mathfrak{B}$
begin

interpretation $\mathcal{Z}\ \alpha$ ⟨*proof*⟩
interpretation $\mathfrak{A}$: *category* $\alpha\ \mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* $\alpha\ \mathfrak{B}$ ⟨*proof*⟩
interpretation *finite-pcategory* α ‹$2_{\mathbb{N}}$› ‹*if2* $\mathfrak{A}\ \mathfrak{B}$›
⟨*proof*⟩

lemma *cf-proj-fst-is-functor*:
assumes $i \in_\circ I$
shows $\pi_{C.1}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-proj-fst-is-functor'*[*cat-cs-intros*]:
assumes $i \in_\circ I$ **and** $\mathfrak{C} = \mathfrak{A} \times_C \mathfrak{B}$ **and** $\mathfrak{D} = \mathfrak{A}$
shows $\pi_{C.1}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-proj-snd-is-functor*:
assumes $i \in_\circ I$
shows $\pi_{C.2}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cf-proj-snd-is-functor'*[*cat-cs-intros*]:
assumes $i \in_\circ I$ **and** $\mathfrak{C} = \mathfrak{A} \times_C \mathfrak{B}$ **and** $\mathfrak{D} = \mathfrak{B}$
shows $\pi_{C.2}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

end

8.14 Product of three categories

8.14.1 Definition and elementary properties.

definition *cat-prod-3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹(- $\times_{C3}$ - $\times_{C3}$ -)› [*81*, *81*, *81*] *80*)
where $\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C} = (\prod_C i \in_\circ 3_{\mathbb{N}}.\ if3\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ i)$

abbreviation *cat-pow-3* :: $V \Rightarrow V$ ($\langle$-$\widehat{\ }_{C3}\rangle$ [*81*] *80*)
  where $\mathfrak{C}\widehat{\ }_{C3} \equiv \mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C}$

Slicing.

lemma *cat-smc-cat-prod-3*[*slicing-commute*]:
  cat-smc $\mathfrak{A} \times_{SMC3}$ *cat-smc* $\mathfrak{B} \times_{SMC3}$ *cat-smc* $\mathfrak{C}$ = *cat-smc* ($\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C}$)
  ⟨*proof*⟩

context
  fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
  assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *category* α $\mathfrak{C}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *category* α $\mathfrak{C}$ ⟨*proof*⟩

lemmas-with
  [
    where $\mathfrak{A}$=‹*cat-smc* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*cat-smc* $\mathfrak{B}$› **and** $\mathfrak{C}$=‹*cat-smc* $\mathfrak{C}$›,
    unfolded slicing-simps slicing-commute,
    OF $\mathfrak{A}$.*cat-semicategory* $\mathfrak{B}$.*cat-semicategory* $\mathfrak{C}$.*cat-semicategory*
  ]:
  cat-prod-3-ObjI = *smc-prod-3-ObjI*
  and *cat-prod-3-ObjI'*[*cat-prod-cs-intros*] = *smc-prod-3-ObjI'*
  and *cat-prod-3-ObjE* = *smc-prod-3-ObjE*
  and *cat-prod-3-ArrI* = *smc-prod-3-ArrI*
  and *cat-prod-3-ArrI'*[*cat-prod-cs-intros*] = *smc-prod-3-ArrI'*
  and *cat-prod-3-ArrE* = *smc-prod-3-ArrE*
  and *cat-prod-3-is-arrI* = *smc-prod-3-is-arrI*
  and *cat-prod-3-is-arrI'*[*cat-prod-cs-intros*] = *smc-prod-3-is-arrI'*
  and *cat-prod-3-is-arrE* = *smc-prod-3-is-arrE*
  and *cat-prod-3-Dom-vsv* = *smc-prod-3-Dom-vsv*
  and *cat-prod-3-Dom-vdomain*[*cat-cs-simps*] = *smc-prod-3-Dom-vdomain*
  and *cat-prod-3-Dom-app*[*cat-prod-cs-simps*] = *smc-prod-3-Dom-app*
  and *cat-prod-3-Dom-vrange* = *smc-prod-3-Dom-vrange*
  and *cat-prod-3-Cod-vsv* = *smc-prod-3-Cod-vsv*
  and *cat-prod-3-Cod-vdomain*[*cat-cs-simps*] = *smc-prod-3-Cod-vdomain*
  and *cat-prod-3-Cod-app*[*cat-prod-cs-simps*] = *smc-prod-3-Cod-app*
  and *cat-prod-3-Cod-vrange* = *smc-prod-3-Cod-vrange*

lemmas-with
  [
    where $\mathfrak{A}$=‹*cat-smc* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*cat-smc* $\mathfrak{B}$› **and** $\mathfrak{C}$=‹*cat-smc* $\mathfrak{C}$›,
    unfolded slicing-simps slicing-commute,
    OF $\mathfrak{A}$.*cat-semicategory* $\mathfrak{B}$.*cat-semicategory* $\mathfrak{C}$.*cat-semicategory*
  ]:
  cat-prod-3-Comp-app[*cat-prod-cs-simps*] = *smc-prod-3-Comp-app*

end

8.14.2 Product of three categories is a category

context
  fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
  assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *category* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *category* α $\mathfrak{C}$ ⟨*proof*⟩

lemma *finite-pcategory-cat-prod-3*: *finite-pcategory* α ($3_{\mathbb{N}}$) (*if3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$)
⟨*proof*⟩

interpretation *finite-pcategory* α ‹$3_{\mathbb{N}}$› ‹*if3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemma *category-cat-prod-3*[*cat-cs-intros*]: *category* α ($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)
⟨*proof*⟩

end

8.14.3 Identity

lemma *cat-prod-3-CId-vsv*[*cat-cs-intros*]: *vsv* (($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*CId*⦈))
⟨*proof*⟩

lemma *cat-prod-3-CId-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*CId*⦈)) = ($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*Obj*⦈)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *category* α $\mathfrak{C}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *category* α $\mathfrak{C}$ ⟨*proof*⟩

interpretation *finite-pcategory* α ‹$3_{\mathbb{N}}$› ‹*if3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemma *cat-prod-3-CId-app*[*cat-prod-cs-simps*]:
assumes [a, b, c]$_\circ$ $\in_\circ$ ($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*Obj*⦈)
shows ($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*CId*⦈)(⦇a, b, c⦈)$_\bullet$ = [$\mathfrak{A}$(⦇*CId*⦈)(⦇a⦈), $\mathfrak{B}$(⦇*CId*⦈)(⦇b⦈), $\mathfrak{C}$(⦇*CId*⦈)(⦇c⦈)]$_\circ$
⟨*proof*⟩

lemma *cat-prod-3-CId-vrange*:
$\mathcal{R}_\circ$ (($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*CId*⦈)) $\subseteq_\circ$ ($\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$)(⦇*Arr*⦈)
⟨*proof*⟩

end

8.15 Conversion of a product of three categories to products of two categories

definition *cf-cat-prod-21-of-3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ =
[
($\lambda A \in_\circ (\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C})$(⦇*Obj*⦈). [[A(⦇0⦈), A(⦇$1_{\mathbb{N}}$⦈)]$_\circ$, A(⦇$2_{\mathbb{N}}$⦈)]$_\circ$),
($\lambda F \in_\circ (\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C})$(⦇*Arr*⦈). [[F(⦇0⦈), F(⦇$1_{\mathbb{N}}$⦈)]$_\circ$, F(⦇$2_{\mathbb{N}}$⦈)]$_\circ$),
$\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$,
($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$) $\times_C$ $\mathfrak{C}$

$]_\circ$

definition *cf-cat-prod-12-of-3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ =
    [
      $(\lambda A \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Obj|\!).\ [A(\!|0|\!),\ [A(\!|1_{\mathbb{N}}|\!),\ A(\!|2_{\mathbb{N}}|\!)]_\circ]_\circ)$,
      $(\lambda F \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Arr|\!).\ [F(\!|0|\!),\ [F(\!|1_{\mathbb{N}}|\!),\ F(\!|2_{\mathbb{N}}|\!)]_\circ]_\circ)$,
      $\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C}$,
      $\mathfrak{A} \times_C (\mathfrak{B} \times_C \mathfrak{C})$
    ]$_\circ$

Components.

lemma *cf-cat-prod-21-of-3-components*:
  shows *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)$ =
    $(\lambda A \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Obj|\!).\ [[A(\!|0|\!)]_\circ,\ A(\!|1_{\mathbb{N}}|\!)]_\circ,\ A(\!|2_{\mathbb{N}}|\!)]_\circ)$
    and *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ArrMap|\!)$ =
    $(\lambda F \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Arr|\!).\ [[F(\!|0|\!)]_\circ,\ F(\!|1_{\mathbb{N}}|\!)]_\circ,\ F(\!|2_{\mathbb{N}}|\!)]_\circ)$
    and [*cat-cs-simps*]: *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|HomDom|\!)$ = $\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C}$
    and [*cat-cs-simps*]: *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|HomCod|\!)$ = $(\mathfrak{A} \times_C \mathfrak{B}) \times_C \mathfrak{C}$
  ⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-components*:
  shows *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)$ =
    $(\lambda A \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Obj|\!).\ [A(\!|0|\!),\ [A(\!|1_{\mathbb{N}}|\!),\ A(\!|2_{\mathbb{N}}|\!)]_\circ]_\circ)$
    and *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ArrMap|\!)$ =
    $(\lambda F \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Arr|\!).\ [F(\!|0|\!),\ [F(\!|1_{\mathbb{N}}|\!),\ F(\!|2_{\mathbb{N}}|\!)]_\circ]_\circ)$
    and [*cat-cs-simps*]: *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|HomDom|\!)$ = $\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C}$
    and [*cat-cs-simps*]: *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|HomCod|\!)$ = $\mathfrak{A} \times_C (\mathfrak{B} \times_C \mathfrak{C})$
  ⟨*proof*⟩

8.15.1 Object

mk-VLambda *cf-cat-prod-21-of-3-components*(*1*)
  |*vsv cf-cat-prod-21-of-3-ObjMap-vsv*[*cat-cs-intros*]|
  |*vdomain cf-cat-prod-21-of-3-ObjMap-vdomain*[*cat-cs-simps*]|
  |*app cf-cat-prod-21-of-3-ObjMap-app′*|

mk-VLambda *cf-cat-prod-12-of-3-components*(*1*)
  |*vsv cf-cat-prod-12-of-3-ObjMap-vsv*[*cat-cs-intros*]|
  |*vdomain cf-cat-prod-12-of-3-ObjMap-vdomain*[*cat-cs-simps*]|
  |*app cf-cat-prod-12-of-3-ObjMap-app′*|

lemma *cf-cat-prod-21-of-3-ObjMap-app*[*cat-cs-simps*]:
  assumes $A = [a,\ b,\ c]_\circ$ **and** $[a,\ b,\ c]_\circ \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Obj|\!)$
  shows *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)(\!|A|\!) = [[a,\ b]_\circ,\ c]_\circ$
  ⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-ObjMap-app*[*cat-cs-simps*]:
  assumes $A = [a,\ b,\ c]_\circ$ **and** $[a,\ b,\ c]_\circ \in_\circ (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})(\!|Obj|\!)$
  shows *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)(\!|A|\!) = [a,\ [b,\ c]_\circ]_\circ$
  ⟨*proof*⟩

lemma *cf-cat-prod-21-of-3-ObjMap-vrange*:
  assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$ **and** *category* α $\mathfrak{C}$
  shows $\mathcal{R}_\circ$ (*cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)$) $\subseteq_\circ ((\mathfrak{A} \times_C \mathfrak{B}) \times_C \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-ObjMap-vrange*:

assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$ **and** *category* α $\mathfrak{C}$
shows $\mathcal{R}_{\circ}$ (*cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$(⟦*ObjMap*⟧)) $\subseteq_{\circ}$ ($\mathfrak{A}$ $\times_C$ ($\mathfrak{B}$ $\times_C$ $\mathfrak{C}$))(⟦*Obj*⟧)
⟨*proof*⟩

8.15.2 Arrow

mk-VLambda *cf-cat-prod-21-of-3-components*(*2*)
|*vsv cf-cat-prod-21-of-3-ArrMap-vsv*[*cat-cs-intros*]|
|*vdomain cf-cat-prod-21-of-3-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-cat-prod-21-of-3-ArrMap-app′*|

mk-VLambda *cf-cat-prod-12-of-3-components*(*2*)
|*vsv cf-cat-prod-12-of-3-ArrMap-vsv*[*cat-cs-intros*]|
|*vdomain cf-cat-prod-12-of-3-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-cat-prod-12-of-3-ArrMap-app′*|

lemma *cf-cat-prod-21-of-3-ArrMap-app*[*cat-cs-simps*]:
assumes $F = [h, g, f]_{\circ}$ **and** $[h, g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})$(⟦*Arr*⟧)
shows *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$(⟦*ArrMap*⟧)(⟦F⟧) = $[[h, g]_{\circ}, f]_{\circ}$
⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-ArrMap-app*[*cat-cs-simps*]:
assumes $F = [h, g, f]_{\circ}$ **and** $[h, g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{C3} \mathfrak{B} \times_{C3} \mathfrak{C})$(⟦*Arr*⟧)
shows *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$(⟦*ArrMap*⟧)(⟦F⟧) = $[h, [g, f]_{\circ}]_{\circ}$
⟨*proof*⟩

8.15.3 Conversion of a product of three categories to products of two categories is a functor

lemma *cf-cat-prod-21-of-3-is-functor*:
assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$ **and** *category* α $\mathfrak{C}$
shows *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$: $\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$) $\times_C$ $\mathfrak{C}$
⟨*proof*⟩

lemma *cf-cat-prod-21-of-3-is-functor′*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and *category* α $\mathfrak{C}$
and $\mathfrak{A}'$ = $\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$
and $\mathfrak{B}'$ = ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$) $\times_C$ $\mathfrak{C}$
shows *cf-cat-prod-21-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-is-functor*:
assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$ **and** *category* α $\mathfrak{C}$
shows *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$: $\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{A}$ $\times_C$ ($\mathfrak{B}$ $\times_C$ $\mathfrak{C}$)
⟨*proof*⟩

lemma *cf-cat-prod-12-of-3-is-functor′*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and *category* α $\mathfrak{C}$
and $\mathfrak{A}'$ = $\mathfrak{A}$ $\times_{C3}$ $\mathfrak{B}$ $\times_{C3}$ $\mathfrak{C}$
and $\mathfrak{B}'$ = $\mathfrak{A}$ $\times_C$ ($\mathfrak{B}$ $\times_C$ $\mathfrak{C}$)
shows *cf-cat-prod-12-of-3* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

8.16 Bifunctors

A bifunctor is defined as a functor from a product of two categories to a category (see Chapter II-3 in [7]). This subsection exposes the elementary properties of the projections of the bifunctors established by fixing an argument in a functor (see Chapter II-3 in [7] for further information).

8.16.1 Definitions and elementary properties

definition *bifunctor-proj-fst* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
($\langle$-,-/$'$(/-,-/$'$)/$_{CF}$)$\rangle$ [*51*, *51*, *51*, *51*] *51*)
where $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$ =
($\mathfrak{S}_{\prod_C i \in_\circ 2_{\mathbb{N}} -_\circ set\ \{1_{\mathbb{N}}\}.\ (i = 0\ ?\ \mathfrak{A} : \mathfrak{B}),\mathfrak{S}(\!|HomCod|\!)}(-,set\ \{\langle 1_{\mathbb{N}},\ b\rangle\})) \circ_{CF}$
cf-singleton 0 $\mathfrak{A}$

definition *bifunctor-proj-snd* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
($\langle$-,-/$'$(/-,-/$'$)/$_{CF}$)$\rangle$ [*51*, *51*, *51*, *51*] *51*)
where $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$ =
($\mathfrak{S}_{\prod_C i \in_\circ 2_{\mathbb{N}} -_\circ set\ \{0\}.\ (i = 0\ ?\ \mathfrak{A} : \mathfrak{B}),\mathfrak{S}(\!|HomCod|\!)}(-,set\ \{\langle 0,\ a\rangle\})) \circ_{CF}$
cf-singleton ($1_{\mathbb{N}}$) $\mathfrak{B}$

abbreviation *bcf-ObjMap-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\otimes_{HM.O^1}\rangle$ *55*)
where $a \otimes_{HM.O\mathfrak{S}} b \equiv \mathfrak{S}(\!|ObjMap|\!)(\!|a,\ b|\!)_\bullet$
abbreviation *bcf-ArrMap-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\otimes_{HM.A^1}\rangle$ *55*)
where $g \otimes_{HM.A\mathfrak{S}} f \equiv \mathfrak{S}(\!|ArrMap|\!)(\!|g,\ f|\!)_\bullet$

Elementary properties.

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ $\langle$*proof*$\rangle$
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ $\langle$*proof*$\rangle$
interpretation *finite-pcategory* α $\langle 2_{\mathbb{N}}\rangle$ $\langle$*if2* $\mathfrak{A}$ $\mathfrak{B}\rangle$
$\langle$*proof*$\rangle$

lemma *cat-singleton-qm-fst-def*[*simp*]:
($\prod_C i \in_\circ set\ \{0\}.\ (i = 0\ ?\ \mathfrak{A} : \mathfrak{B})$) = ($\prod_C i \in_\circ set\ \{0\}.\ \mathfrak{A}$)
$\langle$*proof*$\rangle$

lemma *cat-singleton-qm-snd-def*[*simp*]:
($\prod_C i \in_\circ set\ \{1_{\mathbb{N}}\}.\ (i = 0\ ?\ \mathfrak{A} : \mathfrak{B})$) = ($\prod_C i \in_\circ set\ \{1_{\mathbb{N}}\}.\ \mathfrak{B}$)
$\langle$*proof*$\rangle$

end

8.16.2 Object map

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ $\langle$*proof*$\rangle$
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ $\langle$*proof*$\rangle$

interpretation *finite-pcategory* α $\langle 2_{\mathbb{N}}\rangle$ $\langle$*if2* $\mathfrak{A}$ $\mathfrak{B}\rangle$

⟨*proof*⟩

lemmas-with [*OF* $\mathfrak{A}$.*category-axioms* $\mathfrak{B}$.*category-axioms*, *simp*]:
cat-singleton-qm-fst-def **and** *cat-singleton-qm-snd-def*

lemma *bifunctor-proj-fst-ObjMap-app*[*cat-cs-simps*]:
assumes [a, b]$_{\circ}$ $\in_{\circ}$ ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*Obj*|)
shows ($\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$)(|*ObjMap*|)(|$a$|) = $\mathfrak{S}$(|*ObjMap*|)(|a, b|)$_{\bullet}$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ObjMap-app*[*cat-cs-simps*]:
assumes [a, b]$_{\circ}$ $\in_{\circ}$ ($\mathfrak{A}$ $\times_C$ $\mathfrak{B}$)(|*Obj*|)
shows ($\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$)(|*ObjMap*|)(|$b$|) = $\mathfrak{S}$(|*ObjMap*|)(|a, b|)$_{\bullet}$
⟨*proof*⟩

end

8.16.3 Arrow map

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

interpretation *finite-pcategory* α ‹$2_{\mathbb{N}}$› ‹*if2* $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemmas-with [*OF* $\mathfrak{A}$.*category-axioms* $\mathfrak{B}$.*category-axioms*, *simp*]:
cat-singleton-qm-fst-def **and** *cat-singleton-qm-snd-def*

lemma *bifunctor-proj-fst-ArrMap-app*[*cat-cs-simps*]:
assumes b $\in_{\circ}$ $\mathfrak{B}$(|*Obj*|) **and** f $\in_{\circ}$ $\mathfrak{A}$(|*Arr*|)
shows ($\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$)(|*ArrMap*|)(|$f$|) = $\mathfrak{S}$(|*ArrMap*|)(|f, $\mathfrak{B}$(|*CId*|)(|b|)|)$_{\bullet}$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ArrMap-app*[*cat-cs-simps*]:
assumes a $\in_{\circ}$ $\mathfrak{A}$(|*Obj*|) **and** g $\in_{\circ}$ $\mathfrak{B}$(|*Arr*|)
shows ($\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$)(|*ArrMap*|)(|$g$|) = $\mathfrak{S}$(|*ArrMap*|)(|$\mathfrak{A}$(|*CId*|)(|a|), g|)$_{\bullet}$
⟨*proof*⟩

end

8.16.4 Bifunctor projections are functors

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *category* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *category* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *category* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *category* α $\mathfrak{B}$ ⟨*proof*⟩

interpretation *finite-pcategory* α ‹$2_{\mathbb{N}}$› ‹*if2* $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemmas-with [*OF* $\mathfrak{A}$.*category-axioms* $\mathfrak{B}$.*category-axioms*, *simp*]:
cat-singleton-qm-fst-def **and** *cat-singleton-qm-snd-def*

lemma *bifunctor-proj-fst-is-functor*:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *bifunctor-proj-fst-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $\mathfrak{A}' = \mathfrak{A}$
shows $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *bifunctor-proj-fst-ObjMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *vsv* $((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF})(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *bifunctor-proj-fst-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{D}_\circ((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF})(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *bifunctor-proj-fst-ArrMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *vsv* $((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF})(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *bifunctor-proj-fst-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{D}_\circ((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF})(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

lemma *bifunctor-proj-snd-is-functor*:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *bifunctor-proj-snd-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ObjMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *vsv* $((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF})(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathcal{D}_\circ((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF})(\!|ObjMap|\!)) = \mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ArrMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *vsv* $((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF})(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *bifunctor-proj-snd-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathcal{D}_\circ\ ((\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF})(\!|ArrMap|\!)) = \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

end

8.17 Bifunctor flip

8.17.1 Definition and elementary properties

definition *bifunctor-flip* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ =
$[\mathit{fflip}\ (\mathfrak{F}(\!|ObjMap|\!)),\ \mathit{fflip}\ (\mathfrak{F}(\!|ArrMap|\!)),\ \mathfrak{B} \times_C \mathfrak{A},\ \mathfrak{F}(\!|HomCod|\!)]_\circ$

Components

lemma *bifunctor-flip-components*:
shows *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ObjMap|\!)$ = *fflip* $(\mathfrak{F}(\!|ObjMap|\!))$
and *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ArrMap|\!)$ = *fflip* $(\mathfrak{F}(\!|ArrMap|\!))$
and *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|HomDom|\!) = \mathfrak{B} \times_C \mathfrak{A}$
and *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|HomCod|\!) = \mathfrak{F}(\!|HomCod|\!)$
⟨*proof*⟩

8.17.2 Bifunctor flip object map

lemma *bifunctor-flip-ObjMap-vsv*[*cat-cs-intros*]:
vsv (*bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *bifunctor-flip-ObjMap-app*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ObjMap|\!)(\!|b,\ a|\!)_\bullet = \mathfrak{F}(\!|ObjMap|\!)(\!|a,\ b|\!)_\bullet$
⟨*proof*⟩

lemma *bifunctor-flip-ObjMap-app'*[*cat-cs-simps*]:
assumes $ba = [b,\ a]_\circ$
and *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ObjMap|\!)(\!|ba|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|a,\ b|\!)_\bullet$
⟨*proof*⟩

lemma *bifunctor-flip-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*bifunctor-flip* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}(\!|ObjMap|\!)$) = $(\mathfrak{B} \times_C \mathfrak{A})(\!|Obj|\!)$
⟨*proof*⟩

lemma *bifunctor-flip-ObjMap-vrange*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{A}$

and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ$ (*bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ObjMap|\!)$) = $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

8.17.3 Bifunctor flip arrow map

lemma *bifunctor-flip-ArrMap-vsv*[*cat-cs-intros*]:
vsv (*bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *bifunctor-flip-ArrMap-app*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $g \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{B}(\!|Arr|\!)$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)(\!|f,\ g|\!)_\bullet$ = $\mathfrak{F}(\!|ArrMap|\!)(\!|g,\ f|\!)_\bullet$
⟨*proof*⟩

lemma *bifunctor-flip-ArrMap-app'*[*cat-cs-simps*]:
assumes $fg = [f,\ g]_\circ$
and *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $g \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{B}(\!|Arr|\!)$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)(\!|fg|\!)$ = $\mathfrak{F}(\!|ArrMap|\!)(\!|g,\ f|\!)_\bullet$
⟨*proof*⟩

lemma *bifunctor-flip-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)$) = $(\mathfrak{B} \times_C \mathfrak{A})(\!|Arr|\!)$
⟨*proof*⟩

lemma *bifunctor-flip-ArrMap-vrange*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ$ (*bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)$) = $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

8.17.4 Bifunctor flip is a bifunctor

lemma *bifunctor-flip-is-functor*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} : \mathfrak{B} \times_C \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bifunctor-flip-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{D} = \mathfrak{B} \times_C \mathfrak{A}$

shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.17.5 Double-flip of a bifunctor

lemma *bifunctor-flip-flip*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *bifunctor-flip* $\mathfrak{B}$ $\mathfrak{A}$ (*bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$) = $\mathfrak{F}$
⟨*proof*⟩

8.17.6 A projection of a bifunctor flip

lemma *bifunctor-flip-proj-snd*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}_{\mathfrak{B},\mathfrak{A}}(b,-)_{CF} = \mathfrak{F}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
⟨*proof*⟩

lemma *bifunctor-flip-proj-fst*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}_{\mathfrak{B},\mathfrak{A}}(-,a)_{CF} = \mathfrak{F}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
⟨*proof*⟩

8.17.7 A flip of a bifunctor isomorphism

lemma *bifunctor-flip-is-iso-functor*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C.iso\alpha} \mathfrak{C}$
shows *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} : \mathfrak{B} \times_C \mathfrak{A} \mapsto\mapsto_{C.iso\alpha} \mathfrak{C}$
⟨*proof*⟩

8.18 Array bifunctor

8.18.1 Definition and elementary properties

See Chapter II-3 in [7].

definition *cf-array* :: $V \Rightarrow V \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}$ =
[
(λa∈$_\circ$($\mathfrak{B}$ ×$_C$ $\mathfrak{C}$)(|*Obj*|). $\mathfrak{G}$ (*vpfst a*)(|*ObjMap*|)(|*vpsnd a*|)),
(
λf∈$_\circ$($\mathfrak{B}$ ×$_C$ $\mathfrak{C}$)(|*Arr*|).
$\mathfrak{G}$ ($\mathfrak{B}$(|*Cod*|)(|*vpfst f*|))(|*ArrMap*|)(|*vpsnd f*|) ∘$_{A\mathfrak{D}}$
$\mathfrak{F}$ ($\mathfrak{C}$(|*Dom*|)(|*vpsnd f*|))(|*ArrMap*|)(|*vpfst f*|)
),
$\mathfrak{B}$ ×$_C$ $\mathfrak{C}$,
$\mathfrak{D}$
]$_\circ$

Components.

lemma *cf-array-components*:
shows *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ObjMap \rfloor)$ =
$(\lambda a \in_\circ (\mathfrak{B} \times_C \mathfrak{C})(\lfloor Obj \rfloor).\ \mathfrak{G}\ (vpfst\ a)(\lfloor ObjMap \rfloor)(\lfloor vpsnd\ a \rfloor))$
and *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ArrMap \rfloor)$ =
(
$\lambda f \in_\circ (\mathfrak{B} \times_C \mathfrak{C})(\lfloor Arr \rfloor).$
$\mathfrak{G}\ (\mathfrak{B}(\lfloor Cod \rfloor)(\lfloor vpfst\ f \rfloor))(\lfloor ArrMap \rfloor)(\lfloor vpsnd\ f \rfloor) \circ_{A\mathfrak{D}}$
$\mathfrak{F}\ (\mathfrak{C}(\lfloor Dom \rfloor)(\lfloor vpsnd\ f \rfloor))(\lfloor ArrMap \rfloor)(\lfloor vpfst\ f \rfloor)$
)
and *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor HomDom \rfloor) = \mathfrak{B} \times_C \mathfrak{C}$
and *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor HomCod \rfloor) = \mathfrak{D}$
⟨*proof*⟩

8.18.2 Object map

lemma *cf-array-ObjMap-vsv*: *vsv* (*cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ObjMap \rfloor)$)
⟨*proof*⟩

lemma *cf-array-ObjMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ObjMap \rfloor)$) = $(\mathfrak{B} \times_C \mathfrak{C})(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *cf-array-ObjMap-app*[*cat-cs-simps*]:
assumes $[b, c]_\circ \in_\circ (\mathfrak{B} \times_C \mathfrak{C})(\lfloor Obj \rfloor)$
shows *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor b, c \rfloor)_\bullet = \mathfrak{G}\ b(\lfloor ObjMap \rfloor)(\lfloor c \rfloor)$
⟨*proof*⟩

lemma *cf-array-ObjMap-vrange*:
assumes *category* α $\mathfrak{B}$
and *category* α $\mathfrak{C}$
and $\bigwedge b.\ b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{G}\ b : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ObjMap \rfloor)$) $\subseteq_\circ \mathfrak{D}(\lfloor Obj \rfloor)$
⟨*proof*⟩

8.18.3 Arrow map

lemma *cf-array-ArrMap-vsv*: *vsv* (*cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ArrMap \rfloor)$)
⟨*proof*⟩

lemma *cf-array-ArrMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ArrMap \rfloor)$) = $(\mathfrak{B} \times_C \mathfrak{C})(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *cf-array-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and *category* α $\mathfrak{C}$
and $g : a \mapsto_{\mathfrak{B}} b$
and $f : a' \mapsto_{\mathfrak{C}} b'$
shows *cf-array* $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{D}$ $\mathfrak{F}$ $\mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor g, f \rfloor)_\bullet =$
$\mathfrak{G}\ b(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{D}} \mathfrak{F}\ a'(\lfloor ArrMap \rfloor)(\lfloor g \rfloor)$
⟨*proof*⟩

lemma *cf-array-ArrMap-vrange*:
assumes *category* α $\mathfrak{B}$
and *category* α $\mathfrak{C}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{F}\ c : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge b.\ b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor) \Longrightarrow \mathfrak{G}\ b : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and [*cat-cs-simps*]:

$\bigwedge b\ c.\ b \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{G}\ b(\!|ObjMap|\!)(\!|c|\!) = \mathfrak{F}\ c(\!|ObjMap|\!)(\!|b|\!)$
shows $\mathcal{R}_\circ$ (*cf-array* $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}(\!|ArrMap|\!)) \subseteq_\circ \mathfrak{D}(\!|Arr|\!)$
⟨*proof*⟩

8.18.4 Array bifunctor is a bifunctor

lemma *cf-array-specification*:
— See Proposition 1 from Chapter II-3 in [7].
assumes *category* $\alpha\ \mathfrak{B}$
and *category* $\alpha\ \mathfrak{C}$
and *category* $\alpha\ \mathfrak{D}$
and $\bigwedge c.\ c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{F}\ c : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge b.\ b \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow \mathfrak{G}\ b : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\bigwedge b\ c.\ b \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow c \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow \mathfrak{G}\ b(\!|ObjMap|\!)(\!|c|\!) = \mathfrak{F}\ c(\!|ObjMap|\!)(\!|b|\!)$
and
$\bigwedge b\ c\ b'\ c'\ f\ g.\ [\![\ f : b \mapsto_{\mathfrak{B}} b';\ g : c \mapsto_{\mathfrak{C}} c'\ ]\!] \Longrightarrow$
$\mathfrak{G}\ b'(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{D}} \mathfrak{F}\ c(\!|ArrMap|\!)(\!|f|\!) =$
$\mathfrak{F}\ c'(\!|ArrMap|\!)(\!|f|\!) \circ_{A\mathfrak{D}} \mathfrak{G}\ b(\!|ArrMap|\!)(\!|g|\!)$
shows *cf-array-is-functor*: *cf-array* $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and *cf-array-ObjMap-app-fst*: $\bigwedge b\ c.\ [\![\ b \in_\circ \mathfrak{B}(\!|Obj|\!);\ c \in_\circ \mathfrak{C}(\!|Obj|\!)\ ]\!] \Longrightarrow$
cf-array $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}(\!|ObjMap|\!)(\!|b,\ c|\!)_\bullet = \mathfrak{F}\ c(\!|ObjMap|\!)(\!|b|\!)$
and *cf-array-ObjMap-app-snd*: $\bigwedge b\ c.\ [\![\ b \in_\circ \mathfrak{B}(\!|Obj|\!);\ c \in_\circ \mathfrak{C}(\!|Obj|\!)\ ]\!] \Longrightarrow$
cf-array $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}(\!|ObjMap|\!)(\!|b,\ c|\!)_\bullet = \mathfrak{G}\ b(\!|ObjMap|\!)(\!|c|\!)$
and *cf-array-ArrMap-app-fst*: $\bigwedge a\ b\ f\ c.\ [\![\ f : a \mapsto_{\mathfrak{B}} b;\ c \in_\circ \mathfrak{C}(\!|Obj|\!)]\!] \Longrightarrow$
cf-array $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}(\!|ArrMap|\!)(\!|f,\ \mathfrak{C}(\!|CId|\!)(\!|c|\!)|\!)_\bullet = \mathfrak{F}\ c(\!|ArrMap|\!)(\!|f|\!)$
and *cf-array-ArrMap-app-snd*: $\bigwedge a\ b\ g\ c.\ [\![\ g : a \mapsto_{\mathfrak{C}} b;\ c \in_\circ \mathfrak{B}(\!|Obj|\!)\ ]\!] \Longrightarrow$
cf-array $\mathfrak{B}\ \mathfrak{C}\ \mathfrak{D}\ \mathfrak{F}\ \mathfrak{G}(\!|ArrMap|\!)(\!|\mathfrak{B}(\!|CId|\!)(\!|c|\!),\ g|\!)_\bullet = \mathfrak{G}\ c(\!|ArrMap|\!)(\!|g|\!)$
⟨*proof*⟩

8.19 Composition of a covariant bifunctor and covariant functors

8.19.1 Definition and elementary properties.

definition *cf-bcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-bcomp* $\mathfrak{S}\ \mathfrak{F}\ \mathfrak{G} =$
$[$
$($
$\lambda a \in_\circ (\mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!))(\!|Obj|\!).$
$\mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|vpfst\ a|\!),\ \mathfrak{G}(\!|ObjMap|\!)(\!|vpsnd\ a|\!)|\!)_\bullet$
$),$
$($
$\lambda f \in_\circ (\mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!))(\!|Arr|\!).$
$\mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|vpfst\ f|\!),\ \mathfrak{G}(\!|ArrMap|\!)(\!|vpsnd\ f|\!)|\!)_\bullet$
$),$
$\mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!),$
$\mathfrak{S}(\!|HomCod|\!)$
$]_\circ$

Components.

lemma *cf-bcomp-components*:
shows *cf-bcomp* $\mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|ObjMap|\!) =$
$($
$\lambda a \in_\circ (\mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!))(\!|Obj|\!).$
$\mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|vpfst\ a|\!),\ \mathfrak{G}(\!|ObjMap|\!)(\!|vpsnd\ a|\!)|\!)_\bullet$
$)$
and *cf-bcomp* $\mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|ArrMap|\!) =$
$($
$\lambda f \in_\circ (\mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!))(\!|Arr|\!).$
$\mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|vpfst\ f|\!),\ \mathfrak{G}(\!|ArrMap|\!)(\!|vpsnd\ f|\!)|\!)_\bullet$

)
and *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|HomDom|\!) = \mathfrak{F}(\!|HomDom|\!) \times_C \mathfrak{G}(\!|HomDom|\!)$
and *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|HomCod|\!) = \mathfrak{S}(\!|HomCod|\!)$
⟨*proof*⟩

8.19.2 Object map

lemma *cf-bcomp-ObjMap-vsv*: *vsv* (*cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *cf-bcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$) = $(\mathfrak{B}' \times_C \mathfrak{C}')(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-bcomp-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[a, b]_\circ \in_\circ (\mathfrak{B}' \times_C \mathfrak{C}')(\!|Obj|\!)$
shows *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)(\!|a, b|\!)_\bullet = \mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!), \mathfrak{G}(\!|ObjMap|\!)(\!|b|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-bcomp-ObjMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Obj|\!)$
⟨*proof*⟩

8.19.3 Arrow map

lemma *cf-bcomp-ArrMap-vsv*: *vsv* (*cf-bcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-bcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ArrMap|\!)$) = $(\mathfrak{B}' \times_C \mathfrak{C}')(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-bcomp-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[g, f]_\circ \in_\circ (\mathfrak{B}' \times_C \mathfrak{C}')(\!|Arr|\!)$
shows *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ArrMap|\!)(\!|g, f|\!)_\bullet = \mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!), \mathfrak{G}(\!|ArrMap|\!)(\!|f|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-bcomp-ArrMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ArrMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Arr|\!)$
⟨*proof*⟩

8.19.4 Composition of a covariant bifunctor and covariant functors is a functor

lemma *cf-bcomp-is-functor*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$

shows *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}$: $\mathfrak{B}' \times_C \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-bcomp-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{A}' = \mathfrak{B}' \times_C \mathfrak{C}'$
shows *cf-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}$: $\mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

8.20 Composition of a contracovariant bifunctor and covariant functors

The term *contracovariant bifunctor* is used to refer to a bifunctor that is contravariant in the first argument and covariant in the second argument.

definition *cf-cn-cov-bcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}$ =
[
(
$\lambda a \in_\circ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!))(\!|Obj|\!)$.
$\mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|vpfst\ a|\!),\ \mathfrak{G}(\!|ObjMap|\!)(\!|vpsnd\ a|\!)|\!)_\bullet$
),
(
$\lambda f \in_\circ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!))(\!|Arr|\!)$.
$\mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|vpfst\ f|\!),\ \mathfrak{G}(\!|ArrMap|\!)(\!|vpsnd\ f|\!)|\!)_\bullet$
),
$op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!)$,
$\mathfrak{S}(\!|HomCod|\!)$
]$_\circ$

Components.

lemma *cf-cn-cov-bcomp-components*:
shows *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$ =
(
$\lambda a \in_\circ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!))(\!|Obj|\!)$.
$\mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|vpfst\ a|\!),\ \mathfrak{G}(\!|ObjMap|\!)(\!|vpsnd\ a|\!)|\!)_\bullet$
)
and *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ArrMap|\!)$ =
(
$\lambda f \in_\circ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!))(\!|Arr|\!)$.
$\mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|vpfst\ f|\!),\ \mathfrak{G}(\!|ArrMap|\!)(\!|vpsnd\ f|\!)|\!)_\bullet$
)
and *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|HomDom|\!)$ = $op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!))\ \times_C\ \mathfrak{G}(\!|HomDom|\!)$
and *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|HomCod|\!)$ = $\mathfrak{S}(\!|HomCod|\!)$
⟨*proof*⟩

8.20.1 Object map

lemma *cf-cn-cov-bcomp-ObjMap-vsv*: *vsv* (*cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ $\mathfrak{G}(\!|ObjMap|\!)$) = $(op\text{-}cat\ \mathfrak{B}' \times_C \mathfrak{C}')(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ObjMap-app*[*cat-cs-simps*]:

assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[a, b]_\circ \in_\circ (\textit{op-cat}\ \mathfrak{B}' \times_C \mathfrak{C}')(\!|\textit{Obj}|\!)$
shows
$\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|\textit{ObjMap}|\!)(\!|a, b|\!)_\bullet =$
$\mathfrak{S}(\!|\textit{ObjMap}|\!)(\!|\mathfrak{F}(\!|\textit{ObjMap}|\!)(\!|a|\!), \mathfrak{G}(\!|\textit{ObjMap}|\!)(\!|b|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ObjMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ\ (\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|\textit{ObjMap}|\!)) \subseteq_\circ \mathfrak{D}(\!|\textit{Obj}|\!)$
⟨*proof*⟩

8.20.2 Arrow map

lemma *cf-cn-cov-bcomp-ArrMap-vsv*: $\textit{vsv}\ (\textit{cf-cn-cov-bcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\textit{ArrMap}|\!))$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|\textit{ArrMap}|\!)) = (\textit{op-cat}\ \mathfrak{B}' \times_C \mathfrak{C}')(\!|\textit{Arr}|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[g, f]_\circ \in_\circ (\textit{op-cat}\ \mathfrak{B}' \times_C \mathfrak{C}')(\!|\textit{Arr}|\!)$
shows $\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|\textit{ArrMap}|\!)(\!|g, f|\!)_\bullet =$
$\mathfrak{S}(\!|\textit{ArrMap}|\!)(\!|\mathfrak{F}(\!|\textit{ArrMap}|\!)(\!|g|\!), \mathfrak{G}(\!|\textit{ArrMap}|\!)(\!|f|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-ArrMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ\ (\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}(\!|\textit{ArrMap}|\!)) \subseteq_\circ \mathfrak{D}(\!|\textit{Arr}|\!)$
⟨*proof*⟩

8.20.3 Composition of a contracovariant bifunctor and functors is a functor

lemma *cf-cn-cov-bcomp-is-functor*:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G} : \textit{op-cat}\ \mathfrak{B}' \times_C \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-cn-cov-bcomp-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{A}' = \textit{op-cat}\ \mathfrak{B}' \times_C \mathfrak{C}'$
shows $\textit{cf-cn-cov-bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

8.20.4 Projection of a contracovariant bifunctor and functors

lemma *cf-cn-cov-bcomp-bifunctor-proj-snd*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathit{op\text{-}cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $b \in_\circ \mathfrak{B}'(\!|\mathit{Obj}|\!)$
shows
$\mathit{cf\text{-}cn\text{-}cov\text{-}bcomp}\ \mathfrak{S}\ \mathfrak{F}\ \mathfrak{G}_{\mathit{op\text{-}cat}\ \mathfrak{B}',\mathfrak{C}'}(b,-)_{CF} =$
$(\mathfrak{S}_{\mathit{op\text{-}cat}\ \mathfrak{B},\mathfrak{C}}(\mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|b|\!),-)_{CF}) \circ_{CF} \mathfrak{G}$
⟨*proof*⟩

8.21 Composition of a covariant bifunctor and a covariant functor

8.21.1 Definition and elementary properties

definition *cf-lcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where $\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F} = \mathit{cf\text{-}bcomp}\ \mathfrak{S}\ \mathfrak{F}\ (\mathit{cf\text{-}id}\ \mathfrak{C})$

definition *cf-rcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where $\mathit{cf\text{-}rcomp}\ \mathfrak{B}\ \mathfrak{S}\ \mathfrak{G} = \mathit{cf\text{-}bcomp}\ \mathfrak{S}\ (\mathit{cf\text{-}id}\ \mathfrak{B})\ \mathfrak{G}$

Components.

lemma *cf-lcomp-components*:
shows $\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{HomDom}|\!) = \mathfrak{F}(\!|\mathit{HomDom}|\!) \times_C \mathfrak{C}$
and $\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{HomCod}|\!) = \mathfrak{S}(\!|\mathit{HomCod}|\!)$
⟨*proof*⟩

lemma *cf-rcomp-components*:
shows $\mathit{cf\text{-}rcomp}\ \mathfrak{B}\ \mathfrak{S}\ \mathfrak{G}(\!|\mathit{HomDom}|\!) = \mathfrak{B} \times_C \mathfrak{G}(\!|\mathit{HomDom}|\!)$
and $\mathit{cf\text{-}rcomp}\ \mathfrak{B}\ \mathfrak{S}\ \mathfrak{G}(\!|\mathit{HomCod}|\!) = \mathfrak{S}(\!|\mathit{HomCod}|\!)$
⟨*proof*⟩

8.21.2 Object map

lemma *cf-lcomp-ObjMap-vsv*: $\mathit{vsv}\ (\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!))$
⟨*proof*⟩

lemma *cf-rcomp-ObjMap-vsv*: $\mathit{vsv}\ (\mathit{cf\text{-}rcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!))$
⟨*proof*⟩

lemma *cf-lcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ\ (\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!)) = (\mathfrak{A} \times_C \mathfrak{C})(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemma *cf-rcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (\mathit{cf\text{-}rcomp}\ \mathfrak{B}\ \mathfrak{S}\ \mathfrak{G}(\!|\mathit{ObjMap}|\!)) = (\mathfrak{B} \times_C \mathfrak{A})(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemma *cf-lcomp-ObjMap-app*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$
and $c \in_\circ \mathfrak{C}(\!|\mathit{Obj}|\!)$
shows $\mathit{cf\text{-}lcomp}\ \mathfrak{C}\ \mathfrak{S}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a,\ c|\!)_\bullet = \mathfrak{S}(\!|\mathit{ObjMap}|\!)(\!|\mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!),\ c|\!)_\bullet$
⟨*proof*⟩

lemma *cf-rcomp-ObjMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ObjMap|\!)(\!|b,\ a|\!)_\bullet = \mathfrak{S}(\!|ObjMap|\!)(\!|b,\ \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-lcomp-ObjMap-vrange*:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-rcomp-ObjMap-vrange*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ObjMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Obj|\!)$
⟨*proof*⟩

8.21.3 Arrow map

lemma *cf-lcomp-ArrMap-vsv*: *vsv* (*cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-rcomp-ArrMap-vsv*: *vsv* (*cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-lcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ$ (*cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$) = $(\mathfrak{A} \times_C \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-rcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)$) = $(\mathfrak{B} \times_C \mathfrak{A})(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-lcomp-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $f \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows *cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)(\!|f,\ g|\!)_\bullet = \mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!),\ g|\!)_\bullet$
⟨*proof*⟩

lemma *cf-rcomp-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $f \in_\circ \mathfrak{B}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{A}(\!|Arr|\!)$
shows *cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)(\!|f,\ g|\!)_\bullet = \mathfrak{S}(\!|ArrMap|\!)(\!|f,\ \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-lcomp-ArrMap-vrange*:
assumes *category* α $\mathfrak{C}$

and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\lfloor ArrMap \rfloor)$) $\subseteq_\circ \mathfrak{D}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *cf-rcomp-ArrMap-vrange*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\lfloor ArrMap \rfloor)$) $\subseteq_\circ \mathfrak{D}(\lfloor Arr \rfloor)$
⟨*proof*⟩

8.21.4 Composition of a covariant bifunctor and a covariant functor is a functor

lemma *cf-lcomp-is-functor*:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows *cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F} : \mathfrak{A} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-lcomp-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{A}' = \mathfrak{A} \times_C \mathfrak{C}$
shows *cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-rcomp-is-functor*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows *cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G} : \mathfrak{B} \times_C \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-rcomp-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{A}' = \mathfrak{B} \times_C \mathfrak{A}$
shows *cf-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

8.22 Composition of a contracovariant bifunctor and a covariant functor

definition *cf-cn-cov-lcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}$ = *cf-cn-cov-bcomp* $\mathfrak{S}$ $\mathfrak{F}$ (*cf-id* $\mathfrak{C}$)

definition *cf-cn-cov-rcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}$ = *cf-cn-cov-bcomp* $\mathfrak{S}$ (*cf-id* $\mathfrak{B}$) $\mathfrak{G}$

Components.

lemma *cf-cn-cov-lcomp-components*:
shows *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\lfloor HomDom \rfloor)$ = *op-cat* ($\mathfrak{F}(\lfloor HomDom \rfloor)$) $\times_C$ $\mathfrak{C}$
and *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\lfloor HomCod \rfloor)$ = $\mathfrak{S}(\lfloor HomCod \rfloor)$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-components*:
shows *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|HomDom|\!) = op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{G}(\!|HomDom|\!)$
and *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|HomCod|\!) = \mathfrak{S}(\!|HomCod|\!)$
⟨*proof*⟩

8.22.1 Object map

lemma *cf-cn-cov-lcomp-ObjMap-vsv*: *vsv* (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ObjMap-vsv*: *vsv* (*cf-cn-cov-rcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ$ (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)$) = (*op-cat* $\mathfrak{A} \times_C \mathfrak{C}$)$(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ObjMap|\!)$) = (*op-cat* $\mathfrak{B} \times_C \mathfrak{A}$)$(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ObjMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $a \in_\circ op\text{-}cat\ \mathfrak{A}(\!|Obj|\!)$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)(\!|a,\ c|\!)_\bullet = \mathfrak{S}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!),\ c|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ObjMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ op\text{-}cat\ \mathfrak{B}(\!|Obj|\!)$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ObjMap|\!)(\!|b,\ a|\!)_\bullet = \mathfrak{S}(\!|ObjMap|\!)(\!|b,\ \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ObjMap-vrange*:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S} : op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ObjMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ObjMap-vrange*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S} : op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ObjMap|\!)$) $\subseteq_\circ \mathfrak{D}(\!|Obj|\!)$
⟨*proof*⟩

8.22.2 Arrow map

lemma *cf-cn-cov-lcomp-ArrMap-vsv*: *vsv* (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ArrMap-vsv*: *vsv* (*cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ$ (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$) = (*op-cat* $\mathfrak{A}$ $\times_C$ $\mathfrak{C}$)$(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)$) = (*op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{A}$)$(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $f \in_\circ$ *op-cat* $\mathfrak{A}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)(\!|f,\ g|\!)_\bullet$ = $\mathfrak{S}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!),\ g|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $f \in_\circ$ *op-cat* $\mathfrak{B}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{A}(\!|Arr|\!)$
shows *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)(\!|f,\ g|\!)_\bullet$ = $\mathfrak{S}(\!|ArrMap|\!)(\!|f,\ \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!)|\!)_\bullet$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-ArrMap-vrange*:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}(\!|ArrMap|\!)$) $\subseteq_\circ$ $\mathfrak{D}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-ArrMap-vrange*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{D}$
shows $\mathcal{R}_\circ$ (*cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}(\!|ArrMap|\!)$) $\subseteq_\circ$ $\mathfrak{D}(\!|Arr|\!)$
⟨*proof*⟩

8.22.3 Composition of a contracovariant bifunctor and a covariant functor is a functor

lemma *cf-cn-cov-lcomp-is-functor*:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{D}$
shows *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}$: *op-cat* $\mathfrak{A}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{D}$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{D}$
and $\mathfrak{AC}$ = *op-cat* $\mathfrak{A}$ $\times_C$ $\mathfrak{C}$

shows *cf-cn-cov-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}$: $\mathfrak{AC} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-is-functor*:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G}$: $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
shows *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *cf-cn-cov-rcomp-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G}$: $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $\mathfrak{BA}$ = *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{A}$
shows *cf-cn-cov-rcomp* $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}$: $\mathfrak{BA} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
⟨*proof*⟩

8.22.4 Projection of a composition of a contracovariant bifunctor and a covariant functor

lemma *cf-cn-cov-rcomp-bifunctor-proj-snd*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{B}$
and $\mathfrak{G}$: $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows
cf-cn-cov-rcomp $\mathfrak{B}$ $\mathfrak{S}$ $\mathfrak{G}_{op\text{-}cat\ \mathfrak{B},\mathfrak{A}}(b,-)_{CF}$ =
($\mathfrak{S}_{op\text{-}cat\ \mathfrak{B},\mathfrak{C}}(b,-)_{CF}$) $\circ_{CF}$ $\mathfrak{G}$
⟨*proof*⟩

lemma *cf-cn-cov-lcomp-bifunctor-proj-snd*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F}$: $\mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{S}$: *op-cat* $\mathfrak{B}$ $\times_C$ $\mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$
and $b \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows
cf-cn-cov-lcomp $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{F}_{op\text{-}cat\ \mathfrak{A},\mathfrak{C}}(b,-)_{CF}$ =
($\mathfrak{S}_{op\text{-}cat\ \mathfrak{B},\mathfrak{C}}(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!),-)_{CF}$)
⟨*proof*⟩

8.23 Composition of bifunctors

8.23.1 Definitions and elementary properties

definition *cf-blcomp* :: $V \Rightarrow V$
where *cf-blcomp* $\mathfrak{S}$ =
cf-lcomp ($\mathfrak{S}(\!|HomCod|\!)$) $\mathfrak{S}$ $\mathfrak{S}$ $\circ_{CF}$
cf-cat-prod-21-of-3 ($\mathfrak{S}(\!|HomCod|\!)$) ($\mathfrak{S}(\!|HomCod|\!)$) ($\mathfrak{S}(\!|HomCod|\!)$)

definition *cf-brcomp* :: $V \Rightarrow V$
where *cf-brcomp* $\mathfrak{S}$ =
cf-rcomp ($\mathfrak{S}(\!|HomCod|\!)$) $\mathfrak{S}$ $\mathfrak{S}$ $\circ_{CF}$
cf-cat-prod-12-of-3 ($\mathfrak{S}(\!|HomCod|\!)$) ($\mathfrak{S}(\!|HomCod|\!)$) ($\mathfrak{S}(\!|HomCod|\!)$)

Alternative forms of the definitions.

lemma *cf-blcomp-def'*:
assumes $\mathfrak{S}$: $\mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows *cf-blcomp* $\mathfrak{S}$ = *cf-lcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{S}$ $\circ_{CF}$ *cf-cat-prod-21-of-3* $\mathfrak{C}$ $\mathfrak{C}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *cf-brcomp-def'*:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cf-brcomp* $\mathfrak{S}$ = *cf-rcomp* $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{S}$ $\circ_{CF}$ *cf-cat-prod-12-of-3* $\mathfrak{C}$ $\mathfrak{C}$ $\mathfrak{C}$
⟨*proof*⟩

8.23.2 Compositions of bifunctors are functors

lemma *cf-blcomp-is-functor*:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cf-blcomp* $\mathfrak{S} : \mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-blcomp-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{A}' = \mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C}$
shows *cf-blcomp* $\mathfrak{S} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-brcomp-is-functor*:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *cf-brcomp* $\mathfrak{S} : \mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-brcomp-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{A}' = \mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C}$
shows *cf-brcomp* $\mathfrak{S} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.23.3 Object map

lemma *cf-blcomp-ObjMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *vsv* (*cf-blcomp* $\mathfrak{S}$(|*ObjMap*|))
⟨*proof*⟩

lemma *cf-brcomp-ObjMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *vsv* (*cf-brcomp* $\mathfrak{S}$(|*ObjMap*|))
⟨*proof*⟩

lemma *cf-blcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-blcomp* $\mathfrak{S}$(|*ObjMap*|)) = ($\mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C}$)(|*Obj*|)
⟨*proof*⟩

lemma *cf-brcomp-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-brcomp* $\mathfrak{S}$(|*ObjMap*|)) = ($\mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C}$)(|*Obj*|)
⟨*proof*⟩

lemma *cf-blcomp-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $A = [a, b, c]_\circ$
and $a \in_\circ \mathfrak{C}$(|*Obj*|)
and $b \in_\circ \mathfrak{C}$(|*Obj*|)
and $c \in_\circ \mathfrak{C}$(|*Obj*|)

shows *cf-blcomp* $\mathfrak{S}(\!|ObjMap|\!)(\!|A|\!) = (a \otimes_{HM.O\mathfrak{S}} b) \otimes_{HM.O\mathfrak{S}} c$
⟨*proof*⟩

lemma *cf-brcomp-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $A = [a, b, c]_\circ$
and $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *cf-brcomp* $\mathfrak{S}(\!|ObjMap|\!)(\!|A|\!) = a \otimes_{HM.O\mathfrak{S}} (b \otimes_{HM.O\mathfrak{S}} c)$
⟨*proof*⟩

8.23.4 Arrow map

lemma *cf-blcomp-ArrMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *vsv* (*cf-blcomp* $\mathfrak{S}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-brcomp-ArrMap-vsv*[*cat-cs-intros*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *vsv* (*cf-brcomp* $\mathfrak{S}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-blcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-blcomp* $\mathfrak{S}(\!|ArrMap|\!)$) = $(\mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-brcomp-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ (*cf-brcomp* $\mathfrak{S}(\!|ArrMap|\!)$) = $(\mathfrak{C} \times_{C3} \mathfrak{C} \times_{C3} \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-blcomp-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $F = [h, g, f]_\circ$
and $h \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows *cf-blcomp* $\mathfrak{S}(\!|ArrMap|\!)(\!|F|\!) = (h \otimes_{HM.A\mathfrak{S}} g) \otimes_{HM.A\mathfrak{S}} f$
⟨*proof*⟩

lemma *cf-brcomp-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{S} : \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $F = [h, g, f]_\circ$
and $h \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows *cf-brcomp* $\mathfrak{S}(\!|ArrMap|\!)(\!|F|\!) = h \otimes_{HM.A\mathfrak{S}} (g \otimes_{HM.A\mathfrak{S}} f)$
⟨*proof*⟩

8.24 Binatural transformation

8.24.1 Definitions and elementary properties

In this work, a *binatural transformation* is used to denote a natural transformation of bifunctors.

definition *bnt-proj-fst* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

$(\langle$-,-/$'$(/-,-/$'$)/$_{NTCF})\rangle$ [51, 51, 51, 51] 51)
where $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF}$ =
[
$(\lambda a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor).\ \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a,\ b \rfloor_\bullet)$,
$\mathfrak{N}(\lfloor NTDom \rfloor)_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$,
$\mathfrak{N}(\lfloor NTCod \rfloor)_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$,
$\mathfrak{A}$,
$\mathfrak{N}(\lfloor NTDGCod \rfloor)$
]$_\circ$

definition *bnt-proj-snd* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
$(\langle$-,-/$'$(/-,-/$'$)/$_{NTCF})\rangle$ [51, 51, 51, 51] 51)
where $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF}$ =
[
$(\lambda b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor).\ \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a,\ b \rfloor_\bullet)$,
$\mathfrak{N}(\lfloor NTDom \rfloor)_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$,
$\mathfrak{N}(\lfloor NTCod \rfloor)_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$,
$\mathfrak{B}$,
$\mathfrak{N}(\lfloor NTDGCod \rfloor)$
]$_\circ$

Components

lemma *bnt-proj-fst-components*:
shows $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTMap \rfloor) = (\lambda a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor).\ \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a,\ b \rfloor_\bullet)$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTDom \rfloor) = \mathfrak{N}(\lfloor NTDom \rfloor)_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTCod \rfloor) = \mathfrak{N}(\lfloor NTCod \rfloor)_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTDGDom \rfloor) = \mathfrak{A}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTDGCod \rfloor) = \mathfrak{N}(\lfloor NTDGCod \rfloor)$
⟨*proof*⟩

lemma *bnt-proj-snd-components*:
shows $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTMap \rfloor) = (\lambda b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor).\ \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a,\ b \rfloor_\bullet)$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTDom \rfloor) = \mathfrak{N}(\lfloor NTDom \rfloor)_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTCod \rfloor) = \mathfrak{N}(\lfloor NTCod \rfloor)_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTDGDom \rfloor) = \mathfrak{B}$
and $(\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTDGCod \rfloor) = \mathfrak{N}(\lfloor NTDGCod \rfloor)$
⟨*proof*⟩

8.24.2 Natural transformation maps

mk-VLambda *bnt-proj-fst-components*(*1*)[*folded VLambda-vconst-on*]
|*vsv bnt-proj-fst-NTMap-vsv*[*cat-cs-intros*]|
|*vdomain bnt-proj-fst-NTMap-vdomain*[*cat-cs-simps*]|
|*app bnt-proj-fst-NTMap-app*[*cat-cs-simps*]|

lemma *bnt-proj-fst-vrange*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows $\mathcal{R}_\circ\ ((\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

mk-VLambda *bnt-proj-snd-components*(*1*)[*folded VLambda-vconst-on*]
|*vsv bnt-proj-snd-NTMap-vsv*[*intro*]|
|*vdomain bnt-proj-snd-NTMap-vdomain*[*cat-cs-simps*]|
|*app bnt-proj-snd-NTMap-app*[*cat-cs-simps*]|

lemma *bnt-proj-snd-vrange*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $\mathcal{R}_\circ\ ((\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

8.24.3 Binatural transformation projection is a natural transformation

lemma *bnt-proj-snd-is-ntcf*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} \mapsto_{CF} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-snd-is-ntcf'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
and $\mathfrak{F} = \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
and $\mathfrak{G} = \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-fst-is-ntcf*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} \mapsto_{CF} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-fst-is-ntcf'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
and $\mathfrak{F} = \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $\mathfrak{G} = \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $\mathfrak{A}' = \mathfrak{A}$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.24.4 Array binatural transformation is a natural transformation

lemma *ntcf-array-is-ntcf*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{S} : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *vfsequence* $\mathfrak{N}$
and *vcard* $\mathfrak{N} = 5_{\mathbb{N}}$
and $\mathfrak{N}(\lfloor NTDom \rfloor) = \mathfrak{S}$

and $\mathfrak{N}(\lfloor NTCod \rfloor) = \mathfrak{S}'$
and $\mathfrak{N}(\lfloor NTDGDom \rfloor) = \mathfrak{A} \times_C \mathfrak{B}$
and $\mathfrak{N}(\lfloor NTDGCod \rfloor) = \mathfrak{C}$
and $vsv\ (\mathfrak{N}(\lfloor NTMap \rfloor))$
and $\mathcal{D}_\circ\ (\mathfrak{N}(\lfloor NTMap \rfloor)) = (\mathfrak{A} \times_C \mathfrak{B})(\lfloor Obj \rfloor)$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor);\ b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)\ ]\!] \Longrightarrow$
$\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a,\ b \rfloor)_\bullet : \mathfrak{S}(\lfloor ObjMap \rfloor)(\lfloor a,\ b \rfloor)_\bullet \mapsto_{\mathfrak{C}} \mathfrak{S}'(\lfloor ObjMap \rfloor)(\lfloor a,\ b \rfloor)_\bullet$
and $\bigwedge a.\ a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow$
$\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} \mapsto_{CF} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge b.\ b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor) \Longrightarrow$
$\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} \mapsto_{CF} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.24.5 Binatural transformation projections and isomorphisms

lemma *is-iso-ntcf-if-bnt-proj-snd-is-iso-ntcf*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge a.\ a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor) \Longrightarrow$
$\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} \mapsto_{CF.iso} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *is-iso-ntcf-if-bnt-proj-fst-is-iso-ntcf*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge b.\ b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor) \Longrightarrow$
$\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} : \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} \mapsto_{CF.iso} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-snd-is-iso-ntcf-if-is-iso-ntcf*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} :$
$\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} \mapsto_{CF.iso} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-snd-is-iso-ntcf-if-is-iso-ntcf'*[*cat-cs-intros*]:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} = \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
and $\mathfrak{G} = \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(a,-)_{CF}$
and $\mathfrak{B}' = \mathfrak{B}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-fst-is-iso-ntcf-if-is-iso-ntcf*:
assumes *category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$

and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} :$
$\mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} \mapsto_{CF.iso} \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-proj-fst-is-iso-ntcf-if-is-iso-ntcf'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} = \mathfrak{S}_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $\mathfrak{G} = \mathfrak{S}'_{\mathfrak{A},\mathfrak{B}}(-,b)_{CF}$
and $\mathfrak{A}' = \mathfrak{A}$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

8.25 Binatural transformation flip

8.25.1 Definition and elementary properties

definition *bnt-flip* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$ =
[
fflip ($\mathfrak{N}(\!|NTMap|\!)$),
bifunctor-flip $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}(\!|NTDom|\!)$),
bifunctor-flip $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}(\!|NTCod|\!)$),
$\mathfrak{B} \times_C \mathfrak{A}$,
$\mathfrak{N}(\!|NTDGCod|\!)$
]$_\circ$

Components.

lemma *bnt-flip-components*:
shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)$ = *fflip* ($\mathfrak{N}(\!|NTMap|\!)$)
and *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTDom|\!)$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}(\!|NTDom|\!)$)
and *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTCod|\!)$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}(\!|NTCod|\!)$)
and *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTDGDom|\!)$ = $\mathfrak{B} \times_C \mathfrak{A}$
and *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTDGCod|\!)$ = $\mathfrak{N}(\!|NTDGCod|\!)$
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{S}'$ $\mathfrak{N}$
assumes $\mathfrak{N}$: $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
begin

interpretation $\mathfrak{N}$: *is-ntcf* α ‹$\mathfrak{A} \times_C \mathfrak{B}$› $\mathfrak{C}$ $\mathfrak{S}$ $\mathfrak{S}'$ $\mathfrak{N}$ ⟨*proof*⟩

lemmas *bnt-flip-components'* =
bnt-flip-components[**where** $\mathfrak{A}$=$\mathfrak{A}$ **and** $\mathfrak{B}$=$\mathfrak{B}$ **and** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded cat-cs-simps*]

lemmas [*cat-cs-simps*] = *bnt-flip-components'*(*2–5*)

end

8.25.2 Natural transformation map

lemma *bnt-flip-NTMap-vsv*[*cat-cs-intros*]: *vsv* (*bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)$)
⟨*proof*⟩

lemma *bnt-flip-NTMap-app*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
  shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)(\!|b,\ a|\!)_\bullet = \mathfrak{N}(\!|NTMap|\!)(\!|a,\ b|\!)_\bullet$
  ⟨*proof*⟩

lemma *bnt-flip-NTMap-app'*[*cat-cs-simps*]:
  assumes $ba = [b,\ a]_\circ$
    and *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
    and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
  shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)(\!|ba|\!) = \mathfrak{N}(\!|NTMap|\!)(\!|a,\ b|\!)_\bullet$
  ⟨*proof*⟩

lemma *bnt-flip-NTMap-vdomain*[*cat-cs-simps*]:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\mathcal{D}_\circ$ (*bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)$) = $(\mathfrak{B} \times_C \mathfrak{A})(\!|Obj|\!)$
  ⟨*proof*⟩

lemma *bnt-flip-NTMap-vrange*[*cat-cs-simps*]:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows $\mathcal{R}_\circ$ (*bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}(\!|NTMap|\!)$) = $\mathcal{R}_\circ$ ($\mathfrak{N}(\!|NTMap|\!)$)
⟨*proof*⟩

8.25.3 Binatural transformation flip natural transformation map

lemma *bnt-flip-NTMap-is-ntcf*:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$:
    bifunctor-flip $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}$ $\mapsto_{CF}$ *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}'$:
    $\mathfrak{B} \times_C \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-flip-NTMap-is-ntcf'*[*cat-cs-intros*]:
  assumes *category* α $\mathfrak{A}$
    and *category* α $\mathfrak{B}$
    and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathcal{T}$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}$
    and $\mathcal{T}'$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}'$
    and $\mathfrak{D} = \mathfrak{B} \times_C \mathfrak{A}$
  shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N} : \mathcal{T} \mapsto_{CF} \mathcal{T}' : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  ⟨*proof*⟩

8.25.4 Double-flip of a binatural transformation

lemma *bnt-flip-flip*[*cat-cs-simps*]:

assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *bnt-flip* $\mathfrak{B}$ $\mathfrak{A}$ (*bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$) = $\mathfrak{N}$
⟨*proof*⟩

8.25.5 A projection of a flip of a binatural transformation

lemma *bnt-flip-proj-snd*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}_{\mathfrak{B},\mathfrak{A}}(b,-)_{NTCF}$ = $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(-,b)_{NTCF}$
⟨*proof*⟩

lemma *bnt-flip-proj-fst*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}_{\mathfrak{B},\mathfrak{A}}(-,a)_{NTCF}$ = $\mathfrak{N}_{\mathfrak{A},\mathfrak{B}}(a,-)_{NTCF}$
⟨*proof*⟩

8.25.6 A flip of a binatural isomorphism

lemma *bnt-flip-is-iso-ntcf*:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$:
bifunctor-flip $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}$ $\mapsto_{CF.iso}$ *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}'$:
$\mathfrak{B} \times_C \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *bnt-flip-is-iso-ntcf'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\mathfrak{N} : \mathfrak{S} \mapsto_{CF.iso} \mathfrak{S}' : \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F}$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}$
and $\mathfrak{G}$ = *bifunctor-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{S}'$
and $\mathfrak{D}$ = $\mathfrak{B} \times_C \mathfrak{A}$
shows *bnt-flip* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{D} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

9 Subcategory

9.1 Background

named-theorems *cat-sub-cs-intros*
named-theorems *cat-sub-bw-cs-intros*
named-theorems *cat-sub-fw-cs-intros*
named-theorems *cat-sub-bw-cs-simps*

9.2 Simple subcategory

9.2.1 Definition and elementary properties

See Chapter I-3 in [7].

locale *subcategory* = *sdg*: *category* α $\mathfrak{B}$ + *dg*: *category* α $\mathfrak{C}$ **for** α $\mathfrak{B}$ $\mathfrak{C}$ +
  assumes *subcat-subsemicategory*: *cat-smc* $\mathfrak{B}$ $\subseteq_{SMC\alpha}$ *cat-smc* $\mathfrak{C}$
    and *subcat-CId*: $a \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow \mathfrak{B}(\!|CId|\!)(\!|a|\!) = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$

abbreviation *is-subcategory* ($\langle$(-/ $\subseteq_{C^1}$ -)$\rangle$ [*51*, *51*] *50*)
  where $\mathfrak{B} \subseteq_{C\alpha} \mathfrak{C} \equiv$ *subcategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

lemma (**in** *subcategory*) *subcategory-axioms'*[*cat-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
  shows $\mathfrak{B}' \subseteq_{C\alpha'} \mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *subcategory*) *subcategory-axioms''*[*cat-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $\mathfrak{B} \subseteq_{C\alpha'} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *subcategory-def*[*unfolded subcategory-axioms-def*]
  |*intro subcategoryI*[*intro*!]|
  |*dest subcategoryD*[*dest*]|
  |*elim subcategoryE*[*elim*!]|

lemmas [*cat-sub-cs-intros*] = *subcategoryD*(*1*,*2*)

lemma *subcategoryI'*:
  assumes *category* α $\mathfrak{B}$
    and *category* α $\mathfrak{C}$
    and $\bigwedge a.\ a \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow a \in_\circ \mathfrak{C}(\!|Obj|\!)$
    and $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{B}} b \Longrightarrow f : a \mapsto_{\mathfrak{C}} b$
    and $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{B}} c;\ f : a \mapsto_{\mathfrak{B}} b\ ]\!] \Longrightarrow$
      $g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$
    and $\bigwedge a.\ a \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow \mathfrak{B}(\!|CId|\!)(\!|a|\!) = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
  shows $\mathfrak{B} \subseteq_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

A subcategory is a subsemicategory.

context *subcategory*
begin

interpretation *subsmc*: *subsemicategory* α ‹*cat-smc* $\mathfrak{B}$› ‹*cat-smc* $\mathfrak{C}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:

subcat-Obj-vsubset = *subsmc.subsmc-Obj-vsubset*
and *subcat-is-arr-vsubset* = *subsmc.subsmc-is-arr-vsubset*
and *subcat-subdigraph-op-dg-op-dg* = *subsmc.subsmc-subdigraph-op-dg-op-dg*
and *subcat-objD* = *subsmc.subsmc-objD*
and *subcat-arrD* = *subsmc.subsmc-arrD*
and *subcat-dom-simp* = *subsmc.subsmc-dom-simp*
and *subcat-cod-simp* = *subsmc.subsmc-cod-simp*
and *subcat-is-arrD* = *subsmc.subsmc-is-arrD*

lemmas-with [*unfolded slicing-simps slicing-commute*]:
subcat-Comp-simp = *subsmc.subsmc-Comp-simp*
and *subcat-is-idem-arrD* = *subsmc.subsmc-is-idem-arrD*

end

lemmas [*cat-sub-fw-cs-intros*] =
subcategory.subcat-Obj-vsubset
subcategory.subcat-is-arr-vsubset
subcategory.subcat-objD
subcategory.subcat-arrD
subcategory.subcat-is-arrD

lemmas [*cat-sub-bw-cs-simps*] =
subcategory.subcat-dom-simp
subcategory.subcat-cod-simp

lemmas [*cat-sub-fw-cs-intros*] =
subcategory.subcat-is-idem-arrD

lemmas [*cat-sub-bw-cs-simps*] =
subcategory.subcat-Comp-simp

The opposite subcategory.

lemma (in *subcategory*) *subcat-subcategory-op-cat*: *op-cat* $\mathfrak{B} \subseteq_{C\alpha}$ *op-cat* $\mathfrak{C}$
⟨*proof*⟩

lemmas *subcat-subcategory-op-cat*[*intro*] = *subcategory.subcat-subcategory-op-cat*

Elementary properties.

lemma (in *subcategory*) *subcat-CId-is-arr*[*intro*]:
assumes $a \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{C}(\!|CId|\!)(\!|a|\!) : a \mapsto_{\mathfrak{B}} a$
⟨*proof*⟩

Further rules.

lemma (in *subcategory*) *subcat-CId-simp*[*cat-sub-bw-cs-simps*]:
assumes $a \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{B}(\!|CId|\!)(\!|a|\!) = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
⟨*proof*⟩

lemmas [*cat-sub-bw-cs-simps*] = *subcategory.subcat-CId-simp*

lemma (in *subcategory*) *subcat-is-right-inverseD*[*cat-sub-fw-cs-intros*]:
assumes *is-right-inverse* $\mathfrak{B}$ *g f*
shows *is-right-inverse* $\mathfrak{C}$ *g f*
⟨*proof*⟩

lemmas [*cat-sub-fw-cs-intros*] = *subcategory.subcat-is-right-inverseD*

lemma (**in** *subcategory*) *subcat-is-left-inverseD*[*cat-sub-fw-cs-intros*]:
  assumes *is-left-inverse* $\mathfrak{B}$ *g f*
  shows *is-left-inverse* $\mathfrak{C}$ *g f*
⟨*proof*⟩

lemmas [*cat-sub-fw-cs-intros*] = *subcategory.subcat-is-left-inverseD*

lemma (**in** *subcategory*) *subcat-is-inverseD*[*cat-sub-fw-cs-intros*]:
  assumes *is-inverse* $\mathfrak{B}$ *g f*
  shows *is-inverse* $\mathfrak{C}$ *g f*
  ⟨*proof*⟩

lemmas [*cat-sub-fw-cs-intros*] = *subcategory.subcat-is-inverseD*

lemma (**in** *subcategory*) *subcat-is-iso-arrD*[*cat-sub-fw-cs-intros*]:
  assumes $f : a \mapsto_{iso_{\mathfrak{B}}} b$
  shows $f : a \mapsto_{iso_{\mathfrak{C}}} b$
⟨*proof*⟩

lemmas [*cat-sub-fw-cs-intros*] = *subcategory.subcat-is-iso-arrD*

lemma (**in** *subcategory*) *subcat-the-inverse-simp*[*cat-sub-bw-cs-simps*]:
  assumes $f : a \mapsto_{iso_{\mathfrak{B}}} b$
  shows ${f^{-1}}_{C\mathfrak{B}} = {f^{-1}}_{C\mathfrak{C}}$
⟨*proof*⟩

lemmas [*cat-sub-bw-cs-simps*] = *subcategory.subcat-the-inverse-simp*

lemma (**in** *subcategory*) *subcat-obj-isoD*:
  assumes $a \approx_{obj\mathfrak{B}} b$
  shows $a \approx_{obj\mathfrak{C}} b$
  ⟨*proof*⟩

lemmas [*cat-sub-fw-cs-intros*] = *subcategory.subcat-obj-isoD*

9.2.2 Subcategory relation is a partial order

lemma *subcat-refl*:
  assumes *category* α $\mathfrak{A}$
  shows $\mathfrak{A} \subseteq_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *subcat-trans*:
  assumes $\mathfrak{A} \subseteq_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C\alpha} \mathfrak{C}$
  shows $\mathfrak{A} \subseteq_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *subcat-antisym*:
  assumes $\mathfrak{A} \subseteq_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C\alpha} \mathfrak{A}$
  shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

9.3 Inclusion functor

9.3.1 Definition and elementary properties

See Chapter I-3 in [7].

abbreviation (*input*) *cf-inc* :: $V \Rightarrow V \Rightarrow V$
  where *cf-inc* ≡ *dghm-inc*

Slicing.

lemma *dghm-smcf-inc*[*slicing-commute*]:
  dghm-inc (*cat-smc* $\mathfrak{B}$) (*cat-smc* $\mathfrak{C}$) = *cf-smcf* (*cf-inc* $\mathfrak{B}$ $\mathfrak{C}$)
  ⟨*proof*⟩

Elementary properties.

lemmas [*cat-cs-simps*] =
  dghm-inc-ObjMap-app
  dghm-inc-ArrMap-app

9.3.2 Canonical inclusion functor associated with a subcategory

sublocale *subcategory* ⊆ *inc*: *is-ft-functor* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*cf-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemmas (**in** *subcategory*) *subcat-cf-inc-is-ft-functor* = *inc.is-ft-functor-axioms*

9.3.3 Inclusion functor for the opposite categories

lemma (**in** *subcategory*) *subcat-cf-inc-op-cat-is-functor*:
  cf-inc (*op-cat* $\mathfrak{B}$) (*op-cat* $\mathfrak{C}$) : *op-cat* $\mathfrak{B}$ $\mapsto\!\!\mapsto_{C.faithful\alpha}$ *op-cat* $\mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *subcategory*) *subcat-op-cat-cf-inc*:
  cf-inc (*op-cat* $\mathfrak{B}$) (*op-cat* $\mathfrak{C}$) = *op-cf* (*cf-inc* $\mathfrak{B}$ $\mathfrak{C}$)
  ⟨*proof*⟩

9.4 Full subcategory

See Chapter I-3 in [7].

locale *fl-subcategory* = *subcategory* +
  assumes *fl-subcat-fl-subsemicategory*: *cat-smc* $\mathfrak{B}$ $\subseteq_{SMC.full\alpha}$ *cat-smc* $\mathfrak{C}$

abbreviation *is-fl-subcategory* (‹(-/ $\subseteq_{C.full1}$ -)› [*51*, *51*] *50*)
  where $\mathfrak{B}$ $\subseteq_{C.full\alpha}$ $\mathfrak{C}$ ≡ *fl-subcategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

mk-ide rf *fl-subcategory-def*[*unfolded fl-subcategory-axioms-def*]
  |*intro fl-subcategoryI*|
  |*dest fl-subcategoryD*[*dest*]|
  |*elim fl-subcategoryE*[*elim*!]|

lemmas [*cat-sub-cs-intros*] = *fl-subcategoryD*(*1*)

Elementary properties.

sublocale *fl-subcategory* ⊆ *inc*: *is-fl-functor* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*cf-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

9.5 Wide subcategory

9.5.1 Definition and elementary properties

See [1][3].

locale *wide-subcategory* = *subcategory* +
  assumes *wide-subcat-wide-subsemicategory*: *cat-smc* $\mathfrak{B}$ $\subseteq_{SMC.wide\alpha}$ *cat-smc* $\mathfrak{C}$

abbreviation *is-wide-subcategory* (‹(-/ $\subseteq_{C.wide1}$ -)› [*51*, *51*] *50*)
  where $\mathfrak{B}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{C}$ ≡ *wide-subcategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

mk-ide rf *wide-subcategory-def*[*unfolded wide-subcategory-axioms-def*]
  |*intro wide-subcategoryI*|
  |*dest wide-subcategoryD*[*dest*]|
  |*elim wide-subcategoryE*[*elim*!]|

lemmas [*cat-sub-cs-intros*] = *wide-subcategoryD*(*1*)

Wide subcategory is wide subsemicategory.

context *wide-subcategory*
begin

interpretation *wide-subsmc*: *wide-subsemicategory* α ‹*cat-smc* $\mathfrak{B}$› ‹*cat-smc* $\mathfrak{C}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
  wide-subcat-Obj[*dg-sub-bw-cs-intros*] = *wide-subsmc.wide-subsmc-Obj*
  and *wide-subcat-obj-eq*[*dg-sub-bw-cs-simps*] = *wide-subsmc.wide-subsmc-obj-eq*

end

lemmas [*cat-sub-bw-cs-simps*] = *wide-subcategory.wide-subcat-obj-eq*[*symmetric*]
lemmas [*cat-sub-bw-cs-simps*] = *wide-subsemicategory.wide-subsmc-obj-eq*

9.5.2 The wide subcategory relation is a partial order

lemma *wide-subcat-refl*:
  assumes *category* α $\mathfrak{A}$
  shows $\mathfrak{A}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *wide-subcat-trans*[*trans*]:
  assumes $\mathfrak{A}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{C}$
  shows $\mathfrak{A}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *wide-subcat-antisym*:
  assumes $\mathfrak{A}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B}$ $\subseteq_{C.wide\alpha}$ $\mathfrak{A}$
  shows $\mathfrak{A}$ = $\mathfrak{B}$
⟨*proof*⟩

[3] https://ncatlab.org/nlab/show/wide+subcategory

9.6 Replete subcategory

9.6.1 Definition and elementary properties

See nLab [1][4].

locale *replete-subcategory* = *subcategory* α $\mathfrak{B}$ $\mathfrak{C}$ **for** α $\mathfrak{B}$ $\mathfrak{C}$ +
  assumes *rep-subcat-is-iso-arr-is-arr*:
    $a \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow f : a \mapsto_{iso\mathfrak{C}} b \Longrightarrow f : a \mapsto_{\mathfrak{B}} b$

abbreviation *is-replete-subcategory* $(\langle(\text{-}/\subseteq_{C.rep^1}\text{-})\rangle\ [51, 51]\ 50)$
  where $\mathfrak{B} \subseteq_{C.rep\alpha} \mathfrak{C} \equiv$ *replete-subcategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

mk-ide rf *replete-subcategory-def*[*unfolded replete-subcategory-axioms-def*]
  |*intro replete-subcategoryI*|
  |*dest replete-subcategoryD*[*dest*]|
  |*elim replete-subcategoryE*[*elim*!]|

lemmas [*cat-sub-cs-intros*] = *replete-subcategoryD*(*1*)

Elementary properties.

lemma (**in** *replete-subcategory*)
  rep-subcat-is-iso-arr-is-iso-arr-left:
  assumes $a \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $f : a \mapsto_{iso\mathfrak{C}} b$
  shows $f : a \mapsto_{iso\mathfrak{B}} b$
⟨*proof*⟩

lemma (**in** *replete-subcategory*)
  rep-subcat-is-iso-arr-is-iso-arr-right:
  assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $f : a \mapsto_{iso\mathfrak{C}} b$
  shows $f : a \mapsto_{iso\mathfrak{B}} b$
⟨*proof*⟩

lemma (**in** *replete-subcategory*)
  rep-subcat-is-iso-arr-is-iso-arr-left-iff:
  assumes $a \in_\circ \mathfrak{B}(\!|Obj|\!)$
  shows $f : a \mapsto_{iso\mathfrak{B}} b \longleftrightarrow f : a \mapsto_{iso\mathfrak{C}} b$
  ⟨*proof*⟩

lemma (**in** *replete-subcategory*)
  rep-subcat-is-iso-arr-is-iso-arr-right-iff:
  assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
  shows $f : a \mapsto_{iso\mathfrak{B}} b \longleftrightarrow f : a \mapsto_{iso\mathfrak{C}} b$
  ⟨*proof*⟩

9.6.2 The replete subcategory relation is a partial order

lemma *rep-subcat-refl*:
  assumes *category* α $\mathfrak{A}$
  shows $\mathfrak{A} \subseteq_{C.rep\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *rep-subcat-trans*[*trans*]:
  assumes $\mathfrak{A} \subseteq_{C.rep\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C.rep\alpha} \mathfrak{C}$
  shows $\mathfrak{A} \subseteq_{C.rep\alpha} \mathfrak{C}$
⟨*proof*⟩

[4] https://ncatlab.org/nlab/show/replete+subcategory

lemma *rep-subcat-antisym*:
assumes $\mathfrak{A} \subseteq_{C.rep\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C.rep\alpha} \mathfrak{A}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

9.7 Wide replete subcategory

9.7.1 Definition and elementary properties

locale *wide-replete-subcategory* =
wide-subcategory α $\mathfrak{B}$ $\mathfrak{C}$ + *replete-subcategory* α $\mathfrak{B}$ $\mathfrak{C}$ **for** α $\mathfrak{B}$ $\mathfrak{C}$

abbreviation *is-wide-replete-subcategory* (‹(-/ $\subseteq_{C.wr^1}$ -)› [*51*, *51*] *50*)
where $\mathfrak{B} \subseteq_{C.wr\alpha} \mathfrak{C} \equiv$ *wide-replete-subcategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

mk-ide rf *wide-replete-subcategory-def*
|*intro wide-replete-subcategoryI*|
|*dest wide-replete-subcategoryD*[*dest*]|
|*elim wide-replete-subcategoryE*[*elim*!]|

lemmas [*cat-sub-cs-intros*] = *wide-replete-subcategoryD*

Wide replete subcategory preserves isomorphisms.

lemma (**in** *wide-replete-subcategory*)
wr-subcat-is-iso-arr-is-iso-arr:
$f : a \mapsto_{iso\mathfrak{B}} b \longleftrightarrow f : a \mapsto_{iso\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-sub-bw-cs-simps*] =
wide-replete-subcategory.wr-subcat-is-iso-arr-is-iso-arr

9.7.2 The wide replete subcategory relation is a partial order

lemma *wr-subcat-refl*:
assumes *category* α $\mathfrak{A}$
shows $\mathfrak{A} \subseteq_{C.wr\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *wr-subcat-trans*[*trans*]:
assumes $\mathfrak{A} \subseteq_{C.wr\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C.wr\alpha} \mathfrak{C}$
shows $\mathfrak{A} \subseteq_{C.wr\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *wr-subcat-antisym*:
assumes $\mathfrak{A} \subseteq_{C.wr\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{C.wr\alpha} \mathfrak{A}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

10 Simple categories

10.1 Background

The section presents a variety of simple categories, (such as the empty category *0* and the singleton category *1*) and functors between them (see [7] for further information).

10.2 Empty category *0*

10.2.1 Definition and elementary properties

See Chapter I-2 in [7].

definition *cat-0* :: *V*
  where *cat-0* = [*0*, *0*, *0*, *0*, *0*, *0*]$_\circ$

Components.

lemma *cat-0-components*:
  shows *cat-0*(|*Obj*|) = *0*
    and *cat-0*(|*Arr*|) = *0*
    and *cat-0*(|*Dom*|) = *0*
    and *cat-0*(|*Cod*|) = *0*
    and *cat-0*(|*Comp*|) = *0*
    and *cat-0*(|*CId*|) = *0*
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-cat-0*: *cat-smc cat-0* = *smc-0*
  ⟨*proof*⟩

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-cat-0*, *unfolded slicing-simps*]:
  cat-0-is-arr-iff = *smc-0-is-arr-iff*

10.2.2 *0* is a category

lemma (in $\mathcal{Z}$**)** *category-cat-0*[*cat-cs-intros*]: *category* α *cat-0*
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*category-cat-0*

10.2.3 Opposite of the category *0*

lemma *op-cat-cat-0*[*cat-op-simps*]: *op-cat* (*cat-0*) = *cat-0*
⟨*proof*⟩

10.3 Empty functors

10.3.1 Definition and elementary properties

definition *cf-0* :: *V* ⇒ *V*
  where *cf-0* $\mathfrak{A}$ = [*0*, *0*, *cat-0*, $\mathfrak{A}$]$_\circ$

Components.

lemma *cf-0-components*:
  shows *cf-0* $\mathfrak{A}$(|*ObjMap*|) = *0*
    and *cf-0* $\mathfrak{A}$(|*ArrMap*|) = *0*
    and *cf-0* $\mathfrak{A}$(|*HomDom*|) = *cat-0*
    and *cf-0* $\mathfrak{A}$(|*HomCod*|) = $\mathfrak{A}$

⟨*proof*⟩

Slicing.

lemma *cf-smcf-cf-0*: *cf-smcf* (*cf-0* $\mathfrak{A}$) = *smcf-0* (*cat-smc* $\mathfrak{A}$)
⟨*proof*⟩

Opposite empty category homomorphism.

lemma *op-cf-cf-0*: *op-cf* (*cf-0* $\mathfrak{C}$) = *cf-0* (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

10.3.2 Object map

lemma *cf-0-ObjMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-0* $\mathfrak{C}$(|*ObjMap*|))
⟨*proof*⟩

10.3.3 Arrow map

lemma *cf-0-ArrMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-0* $\mathfrak{C}$(|*ArrMap*|))
⟨*proof*⟩

10.3.4 Empty functor is a faithful functor

lemma *cf-0-is-ft-functor*:
assumes *category* α $\mathfrak{A}$
shows *cf-0* $\mathfrak{A}$: *cat-0* $\mapsto\mapsto_{C.faithful\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *cf-0-is-ft-functor'*[*cf-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and $\mathfrak{B}'$ = $\mathfrak{A}$
and $\mathfrak{A}'$ = *cat-0*
shows *cf-0* $\mathfrak{A}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C.faithful\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemma *cf-0-is-functor*:
assumes *category* α $\mathfrak{A}$
shows *cf-0* $\mathfrak{A}$: *cat-0* $\mapsto\mapsto_{C\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *cf-0-is-functor'*[*cat-cs-intros*]:
assumes *category* α $\mathfrak{A}$
and $\mathfrak{B}'$ = $\mathfrak{A}$
and $\mathfrak{A}'$ = *cat-0*
shows *cf-0* $\mathfrak{A}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

10.3.5 Further properties

lemma *is-functor-is-cf-0-if-cat-0*:
assumes $\mathfrak{F}$: *cat-0* $\mapsto\mapsto_{C\alpha}$ $\mathfrak{C}$
shows $\mathfrak{F}$ = *cf-0* $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-comp-cf-cf-0*[*cat-cs-simps*]: $\mathfrak{F}$ $\circ_{CF}$ *cf-0* $\mathfrak{A}$ = *cf-0* $\mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-comp-cf-cf-0*

10.4 Empty natural transformation

10.4.1 Definition and elementary properties

See Chapter X-1 in [7].

definition *ntcf-0* :: $V \Rightarrow V$
  where *ntcf-0* $\mathfrak{C}$ = $[0, \mathit{cf\text{-}0}\ \mathfrak{C}, \mathit{cf\text{-}0}\ \mathfrak{C}, \mathit{cat\text{-}0}, \mathfrak{C}]_{\circ}$

Components.

lemma *ntcf-0-components*:
  shows *ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTMap} \rfloor)$ = *0*
    and [*cat-cs-simps*]: *ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTDom} \rfloor)$ = *cf-0* $\mathfrak{C}$
    and [*cat-cs-simps*]: *ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTCod} \rfloor)$ = *cf-0* $\mathfrak{C}$
    and [*cat-cs-simps*]: *ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTDGDom} \rfloor)$ = *cat-0*
    and [*cat-cs-simps*]: *ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTDGCod} \rfloor)$ = $\mathfrak{C}$
  ⟨*proof*⟩

Slicing.

lemma *ntcf-ntsmcf-ntcf-0*: *ntcf-ntsmcf* (*ntcf-0* $\mathfrak{A}$) = *ntsmcf-0* (*cat-smc* $\mathfrak{A}$)
  ⟨*proof*⟩

Duality.

lemma *op-ntcf-ntcf-0*: *op-ntcf* (*ntcf-0* $\mathfrak{C}$) = *ntcf-0* (*op-cat* $\mathfrak{C}$)
  ⟨*proof*⟩

10.4.2 Natural transformation map

lemma *ntcf-0-NTMap-vsv*[*cat-cs-intros*]: *vsv* (*ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTMap} \rfloor)$)
  ⟨*proof*⟩

lemma *ntcf-0-NTMap-vdomain*[*cat-cs-simps*]: $\mathcal{D}_{\circ}$ (*ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTMap} \rfloor)$) = *0*
  ⟨*proof*⟩

lemma *ntcf-0-NTMap-vrange*[*cat-cs-simps*]: $\mathcal{R}_{\circ}$ (*ntcf-0* $\mathfrak{C}(\lfloor \mathit{NTMap} \rfloor)$) = *0*
  ⟨*proof*⟩

10.4.3 Empty natural transformation is a natural transformation

lemma (**in** *category*) *cat-ntcf-0-is-ntcfI*:
  ntcf-0 $\mathfrak{C}$: *cf-0* $\mathfrak{C}$ $\mapsto_{CF}$ *cf-0* $\mathfrak{C}$: *cat-0* $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-0-is-ntcfI'*[*cat-cs-intros*]:
  assumes $\mathfrak{F}'$ = *cf-0* $\mathfrak{C}$
    and $\mathfrak{G}'$ = *cf-0* $\mathfrak{C}$
    and $\mathfrak{A}'$ = *cat-0*
    and $\mathfrak{B}'$ = $\mathfrak{C}$
    and $\mathfrak{F}'$ = $\mathfrak{F}$
    and $\mathfrak{G}'$ = $\mathfrak{G}$
  shows *ntcf-0* $\mathfrak{C}$: $\mathfrak{F}'$ $\mapsto_{CF}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-0-is-ntcfI'*

lemma *is-ntcf-is-ntcf-0-if-cat-0*:
  assumes $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: *cat-0* $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{C}$
  shows $\mathfrak{N}$ = *ntcf-0* $\mathfrak{C}$ **and** $\mathfrak{F}$ = *cf-0* $\mathfrak{C}$ **and** $\mathfrak{G}$ = *cf-0* $\mathfrak{C}$
⟨*proof*⟩

10.4.4 Further properties

lemma *ntcf-vcomp-ntcf-ntcf-0*[*cat-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathit{cat\text{-}0} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{N} \bullet_{NTCF} \mathit{ntcf\text{-}0}\ \mathfrak{C} = \mathit{ntcf\text{-}0}\ \mathfrak{C}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-0-ntcf*[*cat-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathit{cat\text{-}0} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathit{ntcf\text{-}0}\ \mathfrak{C} \bullet_{NTCF} \mathfrak{N} = \mathit{ntcf\text{-}0}\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-ntcf-comp-cf-ntcf-0*[*cat-cs-simps*]:
$\mathfrak{F} \circ_{CF-NTCF} \mathit{ntcf\text{-}0}\ \mathfrak{A} = \mathit{ntcf\text{-}0}\ \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-ntcf-comp-cf-ntcf-0*

10.5 *1*: category with one object and one arrow

10.5.1 Definition and elementary properties

See Chapter I-2 in [7].

definition *cat-1* :: $V \Rightarrow V \Rightarrow V$
where *cat-1* $\mathfrak{a}\ \mathfrak{f}$ =
[
set $\{\mathfrak{a}\}$,
set $\{\mathfrak{f}\}$,
set $\{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$,
set $\{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$,
set $\{\langle [\mathfrak{f}, \mathfrak{f}]_\circ, \mathfrak{f}\rangle\}$,
set $\{\langle \mathfrak{a}, \mathfrak{f}\rangle\}$
]$_\circ$

Components.

lemma *cat-1-components*:
shows *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{Obj} \rfloor)$ = *set* $\{\mathfrak{a}\}$
and *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{Arr} \rfloor)$ = *set* $\{\mathfrak{f}\}$
and *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{Dom} \rfloor)$ = *set* $\{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$
and *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{Cod} \rfloor)$ = *set* $\{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$
and *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{Comp} \rfloor)$ = *set* $\{\langle [\mathfrak{f}, \mathfrak{f}]_\circ, \mathfrak{f}\rangle\}$
and *cat-1* $\mathfrak{a}\ \mathfrak{f}(\lfloor \mathit{CId} \rfloor)$ = *set* $\{\langle \mathfrak{a}, \mathfrak{f}\rangle\}$
⟨*proof*⟩

Slicing.

lemma *smc-cat-1*: *cat-smc* (*cat-1* $\mathfrak{a}\ \mathfrak{f}$) = *smc-1* $\mathfrak{a}\ \mathfrak{f}$
⟨*proof*⟩

lemmas-with [*folded smc-cat-1*, *unfolded slicing-simps*]:
cat-1-is-arrI = *smc-1-is-arrI*
and *cat-1-is-arrD* = *smc-1-is-arrD*
and *cat-1-is-arrE* = *smc-1-is-arrE*
and *cat-1-is-arr-iff* = *smc-1-is-arr-iff*
and *cat-1-Comp-app*[*cat-cs-simps*] = *smc-1-Comp-app*

10.5.2 Object

lemma *cat-1-ObjI*[*cat-cs-intros*]:

assumes $a = \mathfrak{a}$
shows $a \in_{\circ} \mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f}\ (\!|\mathit{Obj}|\!)$
⟨*proof*⟩

10.5.3 Identity

lemma *cat-1-CId-app*: $\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f}(\!|\mathit{CId}|\!)(\!|\mathfrak{a}|\!) = \mathfrak{f}$
⟨*proof*⟩

10.5.4 *1* is a category

lemma (in $\mathcal{Z}$**)** *category-cat-1*:
assumes $\mathfrak{a} \in_{\circ} \mathit{Vset}\ \alpha$ **and** $\mathfrak{f} \in_{\circ} \mathit{Vset}\ \alpha$
shows $\mathit{category}\ \alpha\ (\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f})$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*category-cat-1*

lemma (in $\mathcal{Z}$**)** *finite-category-cat-1*:
assumes $\mathfrak{a} \in_{\circ} \mathit{Vset}\ \alpha$ **and** $\mathfrak{f} \in_{\circ} \mathit{Vset}\ \alpha$
shows $\mathit{finite\text{-}category}\ \alpha\ (\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f})$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = $\mathcal{Z}$.*finite-category-cat-1*

10.5.5 Opposite of the category *1*

lemma (in $\mathcal{Z}$**)** *cat-1-op*[*cat-op-simps*]:
assumes $\mathfrak{a} \in_{\circ} \mathit{Vset}\ \alpha$ **and** $\mathfrak{f} \in_{\circ} \mathit{Vset}\ \alpha$
shows $\mathit{op\text{-}cat}\ (\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f}) = \mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f}$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cat-1-op-0*[*cat-op-simps*]: $\mathit{op\text{-}cat}\ (\mathit{cat\text{-}1}\ 0\ 0) = \mathit{cat\text{-}1}\ 0\ 0$
⟨*proof*⟩

10.5.6 Further properties

lemma *cf-const-if-HomCod-is-cat-1*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f}$
shows $\mathfrak{K} = \mathit{cf\text{-}const}\ \mathfrak{B}\ (\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f})\ \mathfrak{a}$
⟨*proof*⟩

lemma *cf-const-if-HomDom-is-cat-1*:
assumes $\mathfrak{K} : \mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{K} = \mathit{cf\text{-}const}\ (\mathit{cat\text{-}1}\ \mathfrak{a}\ \mathfrak{f})\ \mathfrak{C}\ (\mathfrak{K}(\!|\mathit{ObjMap}|\!)(\!|\mathfrak{a}|\!))$
⟨*proof*⟩

11 Discrete category

11.1 Abstract discrete category

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

11.1.1 Definition and elementary properties

See Chapter I-2 in [7].

locale *cat-discrete* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  assumes *cat-discrete-Arr*: $f \in_{\circ} \mathfrak{C}(\!|Arr|\!) \Longrightarrow f \in_{\circ} \mathcal{R}_{\circ}\ (\mathfrak{C}(\!|CId|\!))$

Rules.

lemma (**in** *cat-discrete*)
  assumes $\alpha' = \alpha$ $\mathfrak{C}' = \mathfrak{C}$
  shows *cat-discrete* α' $\mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *cat-discrete-def*[*unfolded cat-discrete-axioms-def*]
  |*intro cat-discreteI*|
  |*dest cat-discreteD*[*dest*]|
  |*elim cat-discreteE*[*elim*]|

lemmas [*cat-discrete-cs-intros*] = *cat-discreteD*(*1*)

Elementary properties.

lemma (**in** *cat-discrete*) *cat-discrete-is-arrD*[*dest*]:
  assumes $f : a \mapsto_{\mathfrak{C}} b$
  shows $b = a$ **and** $f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
⟨*proof*⟩

lemma (**in** *cat-discrete*) *cat-discrete-is-arrE*[*elim*]:
  assumes $f : b \mapsto_{\mathfrak{C}} c$
  obtains a **where** $f : a \mapsto_{\mathfrak{C}} a$ **and** $f = \mathfrak{C}(\!|CId|\!)(\!|a|\!)$
  ⟨*proof*⟩

11.2 The discrete category

As explained in Chapter I-2 in [7], every discrete category is identified with its set of objects. In this work, it is assumed that the set of objects and the set of arrows in the canonical discrete category coincide; the domain and the codomain functions are identities.

11.2.1 Definition and elementary properties

definition *the-cat-discrete* :: $V \Rightarrow V$ $(\langle\cdot_C\rangle)$
  where $:_C\ I = [I,\ I,\ \textit{vid-on}\ I,\ \textit{vid-on}\ I,\ (\lambda fg \in_{\circ} \textit{fid-on}\ I.\ fg(\!|0|\!)),\ \textit{vid-on}\ I]_{\circ}$

Components.

lemma *the-cat-discrete-components*:
  shows $:_C\ I(\!|Obj|\!) = I$
    and $:_C\ I(\!|Arr|\!) = I$
    and $:_C\ I(\!|Dom|\!) = \textit{vid-on}\ I$
    and $:_C\ I(\!|Cod|\!) = \textit{vid-on}\ I$
    and $:_C\ I(\!|Comp|\!) = (\lambda fg \in_{\circ} \textit{fid-on}\ I.\ fg(\!|0|\!))$
    and $:_C\ I(\!|CId|\!) = \textit{vid-on}\ I$
  ⟨*proof*⟩

11.2.2 Domain

mk-VLambda *the-cat-discrete-components*(*3*)[*folded VLambda-vid-on*]
|*vsv the-cat-discrete-Dom-vsv*[*cat-discrete-cs-intros*]|
|*vdomain the-cat-discrete-Dom-vdomain*[*cat-discrete-cs-simps*]|
|*app the-cat-discrete-Dom-app*[*cat-discrete-cs-simps*]|

11.2.3 Codomain

mk-VLambda *the-cat-discrete-components*(*4*)[*folded VLambda-vid-on*]
|*vsv the-cat-discrete-Cod-vsv*[*cat-discrete-cs-intros*]|
|*vdomain the-cat-discrete-Cod-vdomain*[*cat-discrete-cs-simps*]|
|*app the-cat-discrete-Cod-app*[*cat-discrete-cs-simps*]|

11.2.4 Composition

lemma *the-cat-discrete-Comp-vsv*[*cat-discrete-cs-intros*]: *vsv* (:$_C$ *I*(|*Comp*|))
⟨*proof*⟩

lemma *the-cat-discrete-Comp-vdomain*: $\mathcal{D}_\circ$ (:$_C$ *I*(|*Comp*|)) = *fid-on I*
⟨*proof*⟩

lemma *the-cat-discrete-Comp-vrange*:
$\mathcal{R}_\circ$ (:$_C$ *I*(|*Comp*|)) = *I*
⟨*proof*⟩

lemma *the-cat-discrete-Comp-app*[*cat-discrete-cs-simps*]:
assumes *i* $\in_\circ$ *I*
shows *i* $\circ_{A:_C\ I}$ *i* = *i*
⟨*proof*⟩

11.2.5 Identity

mk-VLambda *the-cat-discrete-components*(*6*)[*folded VLambda-vid-on*]
|*vsv the-cat-discrete-CId-vsv*[*cat-discrete-cs-intros*]|
|*vdomain the-cat-discrete-CId-vdomain*[*cat-discrete-cs-simps*]|
|*app the-cat-discrete-CId-app*[*cat-discrete-cs-simps*]|

11.2.6 Arrow with a domain and a codomain

lemma *the-cat-discrete-is-arrI*:
assumes *i* $\in_\circ$ *I*
shows *i* : *i* $\mapsto_{:_C\ I}$ *i*
⟨*proof*⟩

lemma *the-cat-discrete-is-arrI'*[*cat-discrete-cs-intros*]:
assumes *i* $\in_\circ$ *I*
and *a* = *i*
and *b* = *i*
shows *i* : *a* $\mapsto_{:_C\ I}$ *b*
⟨*proof*⟩

lemma *the-cat-discrete-is-arrD*:
assumes *f* : *a* $\mapsto_{:_C\ I}$ *b*
shows *f* : *f* $\mapsto_{:_C\ I}$ *f*
and *a* : *a* $\mapsto_{:_C\ I}$ *a*
and *b* : *b* $\mapsto_{:_C\ I}$ *b*
and *f* $\in_\circ$ *I*
and *a* $\in_\circ$ *I*

and $b \in_{\circ} I$
and $f = a$
and $f = b$
and $b = a$
⟨*proof*⟩

11.2.7 The discrete category is a discrete category

lemma (in $\mathcal{Z}$**)** *cat-discrete-the-cat-discrete*:
assumes $I \subseteq_{\circ} Vset\ \alpha$
shows *cat-discrete* $\alpha\ (:_C\ I)$
⟨*proof*⟩

lemmas [*cat-discrete-cs-intros*] = $\mathcal{Z}$.*cat-discrete-the-cat-discrete*

11.2.8 Opposite discrete category

lemma (in $\mathcal{Z}$**)** *the-cat-discrete-op*[*cat-op-simps*]:
assumes $I \subseteq_{\circ} Vset\ \alpha$
shows *op-cat* $(:_C\ I) = :_C\ I$
⟨*proof*⟩

11.3 Discrete functor

11.3.1 Local assumptions for the discrete functor

See Chapter III in [7]).

locale *cf-discrete* = *category* $\alpha\ \mathfrak{C}$ **for** $\alpha\ I\ F\ \mathfrak{C}$ +
assumes *cf-discrete-selector-vrange*[*cat-discrete-cs-intros*]:
$i \in_{\circ} I \Longrightarrow F\ i \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and *cf-discrete-vdomain-vsubset-Vset*: $I \subseteq_{\circ} Vset\ \alpha$

lemmas (in *cf-discrete***)** *cf-discrete-category* = *category-axioms*

lemmas [*cat-discrete-cs-intros*] = *cf-discrete.cf-discrete-category*

Rules.

lemma (in *cf-discrete***)** *cf-discrete-axioms'*[*cat-discrete-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $I' = I$ **and** $F' = F$
shows *cf-discrete* $\alpha'\ I'\ F'\ \mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cf-discrete-def*[*unfolded cf-discrete-axioms-def*]
|*intro cf-discreteI*|
|*dest cf-discreteD*[*dest*]|
|*elim cf-discreteE*[*elim*]|

Elementary properties.

lemma (in *cf-discrete***)** *cf-discrete-is-functor-cf-CId-selector-is-arr*:
assumes $i \in_{\circ} I$
shows $\mathfrak{C}(\!|CId|\!)(\!|F\ i|\!) : F\ i \mapsto_{\mathfrak{C}} F\ i$
⟨*proof*⟩

lemma (in *cf-discrete***)**
cf-discrete-is-functor-cf-CId-selector-is-arr'[*cat-discrete-cs-intros*]:
assumes $i \in_{\circ} I$ **and** $a = F\ i$ **and** $b = F\ i$
shows $\mathfrak{C}(\!|CId|\!)(\!|F\ i|\!) : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

11.3.2 Definition and elementary properties

definition *the-cf-discrete* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$ $(\langle\colon\!\!\rightarrow\!\colon\rangle)$
  where $:\!\!\rightarrow\colon I\ F\ \mathfrak{C} = [\mathit{VLambda}\ I\ F,\ (\lambda i \in_{\circ} I.\ \mathfrak{C}(\!|\mathit{CId}|\!)(\!|F\ i|\!)),\ :_{C}\ I,\ \mathfrak{C}]_{\circ}$

Components.

lemma *the-cf-discrete-components*:
  shows $:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{ObjMap}|\!) = (\lambda i \in_{\circ} I.\ F\ i)$
    and $:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{ArrMap}|\!) = (\lambda i \in_{\circ} I.\ \mathfrak{C}(\!|\mathit{CId}|\!)(\!|F\ i|\!))$
    and [*cat-discrete-cs-simps*]: $:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{HomDom}|\!) = :_{C}\ I$
    and [*cat-discrete-cs-simps*]: $:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{HomCod}|\!) = \mathfrak{C}$
  ⟨*proof*⟩

11.3.3 Object map

mk-VLambda *the-cf-discrete-components*(*1*)
  |*vsv the-cf-discrete-ObjMap-vsv*[*cat-discrete-cs-intros*]|
  |*vdomain the-cf-discrete-ObjMap-vdomain*[*cat-discrete-cs-simps*]|
  |*app the-cf-discrete-ObjMap-app*[*cat-discrete-cs-simps*]|

lemma (**in** *cf-discrete*) *cf-discrete-the-cf-discrete-ObjMap-vrange*:
  $\mathcal{R}_{\circ}\ (:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{ObjMap}|\!)) \subseteq_{\circ} \mathfrak{C}(\!|\mathit{Obj}|\!)$
  ⟨*proof*⟩

11.3.4 Arrow map

mk-VLambda *the-cf-discrete-components*(*2*)
  |*vsv the-cf-discrete-ArrMap-vsv*[*cat-discrete-cs-intros*]|
  |*vdomain the-cf-discrete-ArrMap-vdomain*[*cat-discrete-cs-simps*]|
  |*app the-cf-discrete-ArrMap-app*[*cat-discrete-cs-simps*]|

lemma (**in** *cf-discrete*) *cf-discrete-the-cf-discrete-ArrMap-vrange*:
  $\mathcal{R}_{\circ}\ (:\!\!\rightarrow\colon I\ F\ \mathfrak{C}(\!|\mathit{ArrMap}|\!)) \subseteq_{\circ} \mathfrak{C}(\!|\mathit{Arr}|\!)$
  ⟨*proof*⟩

11.3.5 Discrete functor is a functor

lemma (**in** *cf-discrete*) *cf-discrete-the-cf-discrete-is-functor*:
  $:\!\!\rightarrow\colon I\ F\ \mathfrak{C} : :_{C}\ I \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *cf-discrete*) *cf-discrete-the-cf-discrete-is-functor'*:
  assumes $\mathfrak{A}' = :_{C}\ I$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $:\!\!\rightarrow\colon I\ F\ \mathfrak{C} : \mathfrak{A}' \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}'$
  ⟨*proof*⟩

lemmas [*cat-discrete-cs-intros*] =
  cf-discrete.cf-discrete-the-cf-discrete-is-functor'

11.3.6 Uniqueness of the discrete category

lemma (**in** *cat-discrete*) *cat-discrete-iso-the-cat-discrete*:
  assumes $I \subseteq_{\circ} \mathit{Vset}\ \alpha$ **and** $I \approx_{\circ} \mathfrak{C}(\!|\mathit{Obj}|\!)$
  obtains F **where** $:\!\!\rightarrow\colon I\ F\ \mathfrak{C} : :_{C}\ I \mapsto\!\!\mapsto_{C.iso\alpha} \mathfrak{C}$
⟨*proof*⟩

11.3.7 Opposite discrete functor

lemma (**in** *cf-discrete*) *cf-discrete-the-cf-discrete-op*[*cat-op-simps*]:

op-cf (:→: *I F* $\mathfrak{C}$) = :→: *I F* (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-simps*] = *cf-discrete.cf-discrete-the-cf-discrete-op*

lemma (**in** *cf-discrete*) *cf-discrete-op*[*cat-op-intros*]:
cf-discrete α *I F* (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cf-discrete.cf-discrete-op*

11.4 Tiny discrete category

11.4.1 Background

named-theorems *cat-small-discrete-cs-simps*
named-theorems *cat-small-discrete-cs-intros*

lemmas [*cat-small-discrete-cs-simps*] = *cat-discrete-cs-simps*
lemmas [*cat-small-discrete-cs-intros*] = *cat-discrete-cs-intros*

11.4.2 Definition and elementary properties

locale *tiny-cat-discrete* = *cat-discrete* α $\mathfrak{C}$ + *tiny-category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$

Rules.

lemma (**in** *tiny-cat-discrete*) *tiny-cat-discrete-axioms'*[*cat-discrete-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows *tiny-cat-discrete* α' $\mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *tiny-cat-discrete-def*
|*intro tiny-cat-discreteI*|
|*dest tiny-cat-discreteD*[*dest*]|
|*elim tiny-cat-discreteE*[*elim*]|

lemmas [*cat-small-discrete-cs-intros*] = *tiny-cat-discreteD*

lemma *tiny-cat-discreteI'*:
assumes *tiny-category* α $\mathfrak{C}$ **and** $\bigwedge f.\ f \in_\circ \mathfrak{C}(\!|Arr|\!) \Longrightarrow f \in_\circ \mathcal{R}_\circ\ (\mathfrak{C}(\!|CId|\!))$
shows *tiny-cat-discrete* α $\mathfrak{C}$
⟨*proof*⟩

11.4.3 The discrete category is a tiny category

lemma (**in** $\mathcal{Z}$) *tiny-cat-discrete-the-cat-discrete*[*cat-small-discrete-cs-intros*]:
assumes $I \in_\circ Vset\ \alpha$
shows *tiny-cat-discrete* α ($:_C$ *I*)
⟨*proof*⟩

lemmas [*cat-small-discrete-cs-intros*] = $\mathcal{Z}$.*cat-discrete-the-cat-discrete*

11.5 Discrete functor with tiny maps

11.5.1 Definition and elementary properties

locale *tm-cf-discrete* = *category* α $\mathfrak{C}$ **for** α *I F* $\mathfrak{C}$ +
assumes *tm-cf-discrete-selector-vrange*[*cat-small-discrete-cs-intros*]:
$i \in_\circ I \Longrightarrow F\ i \in_\circ \mathfrak{C}(\!|Obj|\!)$

and *tm-cf-discrete-ObjMap-in-Vset*: *VLambda I F* $\in_\circ$ *Vset* α
and *tm-cf-discrete-ArrMap-in-Vset*: $(\lambda i \in_\circ I.\ \mathfrak{C}(\!|CId|\!)(\!|F\ i|\!)) \in_\circ$ *Vset* α

Rules.

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-axioms'*[*cat-small-discrete-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $I' = I$ **and** $F' = F$
shows *tm-cf-discrete* α' I' F' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *tm-cf-discrete-def*[*unfolded tm-cf-discrete-axioms-def*]
|*intro tm-cf-discreteI*|
|*dest tm-cf-discreteD*[*dest*]|
|*elim tm-cf-discreteE*[*elim*]|

lemma *tm-cf-discreteI'*:
assumes *cf-discrete* α I F $\mathfrak{C}$
and $(\lambda i \in_\circ I.\ F\ i) \in_\circ$ *Vset* α
and $(\lambda i \in_\circ I.\ \mathfrak{C}(\!|CId|\!)(\!|F\ i|\!)) \in_\circ$ *Vset* α
shows *tm-cf-discrete* α I F $\mathfrak{C}$
⟨*proof*⟩

Elementary properties.

sublocale *tm-cf-discrete* ⊆ *cf-discrete*
⟨*proof*⟩

lemmas (**in** *tm-cf-discrete*) *tm-cf-discrete-is-cf-discrete-axioms* =
cf-discrete-axioms

lemmas [*cat-small-discrete-cs-intros*] =
tm-cf-discrete.tm-cf-discrete-is-cf-discrete-axioms

lemma (**in** *tm-cf-discrete*)
tm-cf-discrete-index-in-Vset[*cat-small-discrete-cs-intros*]:
$I \in_\circ$ *Vset* α
⟨*proof*⟩

11.5.2 Opposite discrete functor with tiny maps

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-op*[*cat-op-intros*]:
tm-cf-discrete α I F (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *tm-cf-discrete.tm-cf-discrete-op*

11.5.3 Discrete functor with tiny maps is a functor with tiny maps

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-the-cf-discrete-is-tm-functor*:
$:\!\!\rightarrow: I\ F\ \mathfrak{C} : :_C\ I \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *tm-cf-discrete*) *tm-cf-discrete-the-cf-discrete-is-tm-functor'*:
assumes $\mathfrak{A}' = :_C\ I$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $:\!\!\rightarrow: I\ F\ \mathfrak{C} : \mathfrak{A}' \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-discrete-cs-intros*] =
tm-cf-discrete.tm-cf-discrete-the-cf-discrete-is-tm-functor'

12 →•← and ←•→: cospan and span

12.1 Background

General information about →•← and ←•→ (also known as cospans and spans, respectively) can be found in in Chapters III-3 and III-4 in [7], as well as nLab [1][56].

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

named-theorems *cat-ss-elem-simps*

definition $\mathfrak{o}_{SS}$ **where** [*cat-ss-elem-simps*]: $\mathfrak{o}_{SS} = 0$
definition $\mathfrak{a}_{SS}$ **where** [*cat-ss-elem-simps*]: $\mathfrak{a}_{SS} = 1_{\mathbb{N}}$
definition $\mathfrak{b}_{SS}$ **where** [*cat-ss-elem-simps*]: $\mathfrak{b}_{SS} = 2_{\mathbb{N}}$
definition $\mathfrak{g}_{SS}$ **where** [*cat-ss-elem-simps*]: $\mathfrak{g}_{SS} = 3_{\mathbb{N}}$
definition $\mathfrak{f}_{SS}$ **where** [*cat-ss-elem-simps*]: $\mathfrak{f}_{SS} = 4_{\mathbb{N}}$

lemma *cat-ss-ineq*:
shows *cat-ss-*$\mathfrak{ab}$[*cat-ss-cs-intros*]: $\mathfrak{a}_{SS} \neq \mathfrak{b}_{SS}$
and *cat-ss-*$\mathfrak{ao}$[*cat-ss-cs-intros*]: $\mathfrak{a}_{SS} \neq \mathfrak{o}_{SS}$
and *cat-ss-*$\mathfrak{bo}$[*cat-ss-cs-intros*]: $\mathfrak{b}_{SS} \neq \mathfrak{o}_{SS}$
and *cat-ss-*$\mathfrak{gf}$[*cat-ss-cs-intros*]: $\mathfrak{g}_{SS} \neq \mathfrak{f}_{SS}$
and *cat-ss-*$\mathfrak{ga}$[*cat-ss-cs-intros*]: $\mathfrak{g}_{SS} \neq \mathfrak{a}_{SS}$
and *cat-ss-*$\mathfrak{gb}$[*cat-ss-cs-intros*]: $\mathfrak{g}_{SS} \neq \mathfrak{b}_{SS}$
and *cat-ss-*$\mathfrak{go}$[*cat-ss-cs-intros*]: $\mathfrak{g}_{SS} \neq \mathfrak{o}_{SS}$
and *cat-ss-*$\mathfrak{fa}$[*cat-ss-cs-intros*]: $\mathfrak{f}_{SS} \neq \mathfrak{a}_{SS}$
and *cat-ss-*$\mathfrak{fb}$[*cat-ss-cs-intros*]: $\mathfrak{f}_{SS} \neq \mathfrak{b}_{SS}$
and *cat-ss-*$\mathfrak{fo}$[*cat-ss-cs-intros*]: $\mathfrak{f}_{SS} \neq \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)**
shows *cat-ss-*$\mathfrak{a}$[*cat-ss-cs-intros*]: $\mathfrak{a}_{SS} \in_{\circ} \mathit{Vset}\ \alpha$
and *cat-ss-*$\mathfrak{b}$[*cat-ss-cs-intros*]: $\mathfrak{b}_{SS} \in_{\circ} \mathit{Vset}\ \alpha$
and *cat-ss-*$\mathfrak{o}$[*cat-ss-cs-intros*]: $\mathfrak{o}_{SS} \in_{\circ} \mathit{Vset}\ \alpha$
and *cat-ss-*$\mathfrak{g}$[*cat-ss-cs-intros*]: $\mathfrak{g}_{SS} \in_{\circ} \mathit{Vset}\ \alpha$
and *cat-ss-*$\mathfrak{f}$[*cat-ss-cs-intros*]: $\mathfrak{f}_{SS} \in_{\circ} \mathit{Vset}\ \alpha$
⟨*proof*⟩

12.2 Composable arrows in →•← and ←•→

abbreviation *cat-scospan-composable* :: *V*
where *cat-scospan-composable* ≡
$(\mathit{set}\ \{\mathfrak{o}_{SS}\} \times_{\bullet} \mathit{set}\ \{\mathfrak{o}_{SS}, \mathfrak{g}_{SS}, \mathfrak{f}_{SS}\}) \cup_{\circ}$
$(\mathit{set}\ \{\mathfrak{g}_{SS}, \mathfrak{a}_{SS}\} \times_{\bullet} \mathit{set}\ \{\mathfrak{a}_{SS}\}) \cup_{\circ}$
$(\mathit{set}\ \{\mathfrak{f}_{SS}, \mathfrak{b}_{SS}\} \times_{\bullet} \mathit{set}\ \{\mathfrak{b}_{SS}\})$

abbreviation *cat-sspan-composable* :: *V*
where *cat-sspan-composable* ≡ $(\mathit{cat\text{-}scospan\text{-}composable})^{-1_{\bullet}}$

Rules.

lemma *cat-scospan-composable-*$\mathfrak{oo}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $[g, f]_{\circ} \in_{\circ} \mathit{cat\text{-}scospan\text{-}composable}$
⟨*proof*⟩

[5] https://ncatlab.org/nlab/show/cospan
[6] https://ncatlab.org/nlab/show/span

lemma *cat-scospan-composable-*$\mathfrak{og}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composable-*$\mathfrak{of}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composable-*$\mathfrak{ga}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composable-*$\mathfrak{fb}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composable-*$\mathfrak{aa}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composable-*$\mathfrak{bb}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
⟨*proof*⟩

lemma *cat-scospan-composableE*:
assumes $[g, f]_\circ \in_\circ$ *cat-scospan-composable*
obtains $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
| $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
| $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
| $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
| $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
| $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
| $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{oo}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{go}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{fo}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{ag}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{g}_{SS}$

shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{bf}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{aa}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composable-*$\mathfrak{bb}$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
⟨*proof*⟩

lemma *cat-sspan-composableE*:
assumes $[g, f]_\circ \in_\circ$ *cat-sspan-composable*
obtains $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
| $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
| $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
| $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
| $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
| $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
| $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
⟨*proof*⟩

12.3 Categories →•← and ←•→

12.3.1 Definition and elementary properties

See Chapter III-3 and Chapter III-4 in [7].

definition *the-cat-scospan* :: V ($\langle \to\bullet\leftarrow_C \rangle$)
where $\to\bullet\leftarrow_C =$
[
set $\{\mathfrak{a}_{SS}, \mathfrak{b}_{SS}, \mathfrak{o}_{SS}\}$,
set $\{\mathfrak{a}_{SS}, \mathfrak{g}_{SS}, \mathfrak{o}_{SS}, \mathfrak{f}_{SS}, \mathfrak{b}_{SS}\}$,
(
$\lambda x \in_\circ$ *set* $\{\mathfrak{a}_{SS}, \mathfrak{g}_{SS}, \mathfrak{o}_{SS}, \mathfrak{f}_{SS}, \mathfrak{b}_{SS}\}$.
if $x = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}_{SS}$
| $x = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}_{SS}$
| $x = \mathfrak{g}_{SS} \Rightarrow \mathfrak{a}_{SS}$
| $x = \mathfrak{f}_{SS} \Rightarrow \mathfrak{b}_{SS}$
| *otherwise* $\Rightarrow \mathfrak{o}_{SS}$
),
(
$\lambda x \in_\circ$ *set* $\{\mathfrak{a}_{SS}, \mathfrak{g}_{SS}, \mathfrak{o}_{SS}, \mathfrak{f}_{SS}, \mathfrak{b}_{SS}\}$.
if $x = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}_{SS}$
| $x = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}_{SS}$
| *otherwise* $\Rightarrow \mathfrak{o}_{SS}$
),
(
$\lambda gf \in_\circ$ *cat-scospan-composable*.
if $gf = [\mathfrak{o}_{SS}, \mathfrak{g}_{SS}]_\circ \Rightarrow \mathfrak{g}_{SS}$
| $gf = [\mathfrak{o}_{SS}, \mathfrak{f}_{SS}]_\circ \Rightarrow \mathfrak{f}_{SS}$
| *otherwise* $\Rightarrow gf(\!|0|\!)$

```
    ),
    vid-on (set {𝔞_SS, 𝔟_SS, 𝔬_SS})
  ]∘

definition the-cat-sspan :: V (‹←•→_C›)
  where ←•→_C =
  [
    set {𝔞_SS, 𝔟_SS, 𝔬_SS},
    set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS},
    (
      λx∈∘set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS}.
       if x = 𝔞_SS ⇒ 𝔞_SS
        | x = 𝔟_SS ⇒ 𝔟_SS
        | otherwise ⇒ 𝔬_SS
    ),
    (
      λx∈∘set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS}.
       if x = 𝔞_SS ⇒ 𝔞_SS
        | x = 𝔟_SS ⇒ 𝔟_SS
        | x = 𝔤_SS ⇒ 𝔞_SS
        | x = 𝔣_SS ⇒ 𝔟_SS
        | otherwise ⇒ 𝔬_SS
    ),
    (
      λgf∈∘cat-sspan-composable.
       if gf = [𝔞_SS, 𝔤_SS]∘ ⇒ 𝔤_SS
        | gf = [𝔟_SS, 𝔣_SS]∘ ⇒ 𝔣_SS
        | otherwise ⇒ gf(|0|)
    ),
    vid-on (set {𝔞_SS, 𝔟_SS, 𝔬_SS})
  ]∘
```

Components.

```
lemma the-cat-scospan-components:
  shows →•←_C(|Obj|) = set {𝔞_SS, 𝔟_SS, 𝔬_SS}
    and →•←_C(|Arr|) = set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS}
    and →•←_C(|Dom|) =
      (
        λx∈∘set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS}.
         if x = 𝔞_SS ⇒ 𝔞_SS
          | x = 𝔟_SS ⇒ 𝔟_SS
          | x = 𝔤_SS ⇒ 𝔞_SS
          | x = 𝔣_SS ⇒ 𝔟_SS
          | otherwise ⇒ 𝔬_SS
      )
    and →•←_C(|Cod|) =
      (
        λx∈∘set {𝔞_SS, 𝔤_SS, 𝔬_SS, 𝔣_SS, 𝔟_SS}.
         if x = 𝔞_SS ⇒ 𝔞_SS
          | x = 𝔟_SS ⇒ 𝔟_SS
          | otherwise ⇒ 𝔬_SS
      )
    and →•←_C(|Comp|) =
      (
        λgf∈∘cat-scospan-composable.
         if gf = [𝔬_SS, 𝔤_SS]∘ ⇒ 𝔤_SS
          | gf = [𝔬_SS, 𝔣_SS]∘ ⇒ 𝔣_SS
          | otherwise ⇒ gf(|0|)
```

)
and $\rightarrow\!\bullet\!\leftarrow_C(\!|CId|\!) = vid\text{-}on\ (set\ \{\mathfrak{a}_{SS},\ \mathfrak{b}_{SS},\ \mathfrak{o}_{SS}\})$
⟨*proof*⟩

lemma *the-cat-sspan-components*:
shows $\leftarrow\!\bullet\!\rightarrow_C(\!|Obj|\!) = set\ \{\mathfrak{a}_{SS},\ \mathfrak{b}_{SS},\ \mathfrak{o}_{SS}\}$
and $\leftarrow\!\bullet\!\rightarrow_C(\!|Arr|\!) = set\ \{\mathfrak{a}_{SS},\ \mathfrak{g}_{SS},\ \mathfrak{o}_{SS},\ \mathfrak{f}_{SS},\ \mathfrak{b}_{SS}\}$
and $\leftarrow\!\bullet\!\rightarrow_C(\!|Dom|\!) =$
(
$\lambda x \in_\circ set\ \{\mathfrak{a}_{SS},\ \mathfrak{g}_{SS},\ \mathfrak{o}_{SS},\ \mathfrak{f}_{SS},\ \mathfrak{b}_{SS}\}.$
if $x = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}_{SS}$
$\mid x = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}_{SS}$
$\mid otherwise \Rightarrow \mathfrak{o}_{SS}$
)
and $\leftarrow\!\bullet\!\rightarrow_C(\!|Cod|\!) =$
(
$\lambda x \in_\circ set\ \{\mathfrak{a}_{SS},\ \mathfrak{g}_{SS},\ \mathfrak{o}_{SS},\ \mathfrak{f}_{SS},\ \mathfrak{b}_{SS}\}.$
if $x = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}_{SS}$
$\mid x = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}_{SS}$
$\mid x = \mathfrak{g}_{SS} \Rightarrow \mathfrak{a}_{SS}$
$\mid x = \mathfrak{f}_{SS} \Rightarrow \mathfrak{b}_{SS}$
$\mid otherwise \Rightarrow \mathfrak{o}_{SS}$
)
and $\leftarrow\!\bullet\!\rightarrow_C(\!|Comp|\!) =$
(
$\lambda gf \in_\circ cat\text{-}sspan\text{-}composable.$
if $gf = [\mathfrak{a}_{SS},\ \mathfrak{g}_{SS}]_\circ \Rightarrow \mathfrak{g}_{SS}$
$\mid gf = [\mathfrak{b}_{SS},\ \mathfrak{f}_{SS}]_\circ \Rightarrow \mathfrak{f}_{SS}$
$\mid otherwise \Rightarrow gf(\!|0|\!)$
)
and $\leftarrow\!\bullet\!\rightarrow_C(\!|CId|\!) = vid\text{-}on\ (set\ \{\mathfrak{a}_{SS},\ \mathfrak{b}_{SS},\ \mathfrak{o}_{SS}\})$
⟨*proof*⟩

Elementary properties.

lemma *the-cat-scospan-components-vsv*[*cat-ss-cs-intros*]: $vsv\ (\rightarrow\!\bullet\!\leftarrow_C)$
⟨*proof*⟩

lemma *the-cat-sspan-components-vsv*[*cat-ss-cs-intros*]: $vsv\ (\leftarrow\!\bullet\!\rightarrow_C)$
⟨*proof*⟩

12.3.2 Objects

lemma *the-cat-scospan-Obj-*$\mathfrak{o}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}_{SS}$
shows $a \in_\circ \rightarrow\!\bullet\!\leftarrow_C(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Obj-*$\mathfrak{a}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{a}_{SS}$
shows $a \in_\circ \rightarrow\!\bullet\!\leftarrow_C(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Obj-*$\mathfrak{b}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{b}_{SS}$
shows $a \in_\circ \rightarrow\!\bullet\!\leftarrow_C(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-ObjE*:

assumes $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Obj|\!)$
obtains $\langle a = \mathfrak{o}_{SS} \rangle \mid \langle a = \mathfrak{a}_{SS} \rangle \mid \langle a = \mathfrak{b}_{SS} \rangle$
⟨*proof*⟩

lemma *the-cat-sspan-Obj-*$\mathfrak{o}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}_{SS}$
shows $a \in_{\circ} \leftarrow\bullet\rightarrow_{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Obj-*$\mathfrak{a}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{a}_{SS}$
shows $a \in_{\circ} \leftarrow\bullet\rightarrow_{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Obj-*$\mathfrak{b}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{b}_{SS}$
shows $a \in_{\circ} \leftarrow\bullet\rightarrow_{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-ObjE*:
assumes $a \in_{\circ} \leftarrow\bullet\rightarrow_{C}(\!|Obj|\!)$
obtains $\langle a = \mathfrak{o}_{SS} \rangle \mid \langle a = \mathfrak{a}_{SS} \rangle \mid \langle a = \mathfrak{b}_{SS} \rangle$
⟨*proof*⟩

12.3.3 Arrows

lemma *the-cat-scospan-Arr-*$\mathfrak{a}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{a}_{SS}$
shows $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Arr-*$\mathfrak{b}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{b}_{SS}$
shows $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Arr-*$\mathfrak{o}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}_{SS}$
shows $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Arr-*$\mathfrak{g}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{g}_{SS}$
shows $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-Arr-*$\mathfrak{f}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{f}_{SS}$
shows $a \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-scospan-ArrE*:
assumes $f \in_{\circ} \to\bullet\leftarrow_{C}(\!|Arr|\!)$
obtains $\langle f = \mathfrak{a}_{SS} \rangle \mid \langle f = \mathfrak{b}_{SS} \rangle \mid \langle f = \mathfrak{o}_{SS} \rangle \mid \langle f = \mathfrak{g}_{SS} \rangle \mid \langle f = \mathfrak{f}_{SS} \rangle$
⟨*proof*⟩

lemma *the-cat-sspan-Arr-*$\mathfrak{a}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{a}_{SS}$

shows $a \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Arr-*$\mathfrak{b}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{b}_{SS}$
shows $a \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Arr-*$\mathfrak{o}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}_{SS}$
shows $a \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Arr-*$\mathfrak{g}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{g}_{SS}$
shows $a \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-Arr-*$\mathfrak{f}I$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{f}_{SS}$
shows $a \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-sspan-ArrE*:
assumes $f \in_{\circ} \leftarrow\!\!\bullet\!\!\rightarrow_{C}(\!|Arr|\!)$
obtains $\langle f = \mathfrak{a}_{SS}\rangle \mid \langle f = \mathfrak{b}_{SS}\rangle \mid \langle f = \mathfrak{o}_{SS}\rangle \mid \langle f = \mathfrak{g}_{SS}\rangle \mid \langle f = \mathfrak{f}_{SS}\rangle$
⟨*proof*⟩

12.3.4 Domain

mk-VLambda *the-cat-scospan-components*(*3*)
|*vsv the-cat-scospan-Dom-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-scospan-Dom-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-scospan-Dom-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\rightarrow\!\!\bullet\!\!\leftarrow_{C}(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Dom-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\rightarrow\!\!\bullet\!\!\leftarrow_{C}(\!|Dom|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Dom-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\rightarrow\!\!\bullet\!\!\leftarrow_{C}(\!|Dom|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Dom-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\rightarrow\!\!\bullet\!\!\leftarrow_{C}(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Dom-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$
shows $\rightarrow\!\!\bullet\!\!\leftarrow_{C}(\!|Dom|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

mk-VLambda *the-cat-sspan-components*(*3*)
|*vsv the-cat-sspan-Dom-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-sspan-Dom-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-sspan-Dom-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\leftarrow\!\cdot\!\rightarrow_C(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Dom-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\leftarrow\!\cdot\!\rightarrow_C(\!|Dom|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Dom-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\leftarrow\!\cdot\!\rightarrow_C(\!|Dom|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Dom-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\leftarrow\!\cdot\!\rightarrow_C(\!|Dom|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Dom-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$
shows $\leftarrow\!\cdot\!\rightarrow_C(\!|Dom|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

12.3.5 Codomain

mk-VLambda *the-cat-scospan-components*(*4*)
|*vsv the-cat-scospan-Cod-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-scospan-Cod-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-scospan-Cod-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\rightarrow\!\cdot\!\leftarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Cod-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\rightarrow\!\cdot\!\leftarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Cod-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\rightarrow\!\cdot\!\leftarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Cod-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\rightarrow\!\cdot\!\leftarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-scospan-Cod-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$

shows $\rightarrow\cdot\leftarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

mk-VLambda *the-cat-sspan-components*(4)
|*vsv the-cat-sspan-Cod-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-sspan-Cod-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-sspan-Cod-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\leftarrow\cdot\rightarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Cod-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\leftarrow\cdot\rightarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Cod-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\leftarrow\cdot\rightarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{o}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Cod-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\leftarrow\cdot\rightarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{a}_{SS}$
⟨*proof*⟩

lemma *the-cat-sspan-Cod-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$
shows $\leftarrow\cdot\rightarrow_C(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}_{SS}$
⟨*proof*⟩

12.3.6 Composition

mk-VLambda *the-cat-scospan-components*(5)
|*vsv the-cat-scospan-Comp-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-scospan-Comp-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-scospan-Comp-app-*$\mathfrak{a}\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $g \circ_{A\rightarrow\cdot\leftarrow_C} f = g$ $g \circ_{A\rightarrow\cdot\leftarrow_C} f = f$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{b}\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $g \circ_{A\rightarrow\cdot\leftarrow_C} f = g$ $g \circ_{A\rightarrow\cdot\leftarrow_C} f = f$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{o}\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $g \circ_{A\rightarrow\cdot\leftarrow_C} f = g$ $g \circ_{A\rightarrow\cdot\leftarrow_C} f = f$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{o}\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
shows $g \circ_{A\rightarrow\cdot\leftarrow_C} f = f$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{of}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
shows $g \circ_{A \to \bullet \leftarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{ga}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $g \circ_{A \to \bullet \leftarrow C} f = g$
⟨*proof*⟩

lemma *the-cat-scospan-Comp-app-*$\mathfrak{fb}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $g \circ_{A \to \bullet \leftarrow C} f = g$
⟨*proof*⟩

mk-VLambda *the-cat-sspan-components*(*5*)
|*vsv the-cat-sspan-Comp-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-sspan-Comp-vdomain*[*cat-ss-cs-simps*]|

lemma *the-cat-sspan-Comp-app-*$\mathfrak{aa}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = g$ $g \circ_{A \leftarrow \bullet \rightarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{bb}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = g$ $g \circ_{A \leftarrow \bullet \rightarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{oo}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = g$ $g \circ_{A \leftarrow \bullet \rightarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{ag}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{bf}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = f$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{go}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{g}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = g$
⟨*proof*⟩

lemma *the-cat-sspan-Comp-app-*$\mathfrak{fo}$[*cat-ss-cs-simps*]:
assumes $g = \mathfrak{f}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $g \circ_{A \leftarrow \bullet \rightarrow C} f = g$
⟨*proof*⟩

12.3.7 Identity

mk-VLambda *the-cat-scospan-components*(*6*)[*folded VLambda-vid-on*]
|*vsv the-cat-scospan-CId-vsv*[*cat-ss-cs-intros*]|

|*vdomain the-cat-scospan-CId-vdomain*[*cat-ss-cs-simps*]|
|*app the-cat-scospan-CId-app*[*cat-ss-cs-simps*]|

mk-VLambda *the-cat-sspan-components*(*6*)[*folded VLambda-vid-on*]
|*vsv the-cat-sspan-CId-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cat-sspan-CId-vdomain*[*cat-ss-cs-simps*]|
|*app the-cat-sspan-CId-app*[*cat-ss-cs-simps*]|

12.3.8 Arrow with a domain and a codomain

lemma *the-cat-scospan-is-arr-*$\mathfrak{aaa}$[*cat-ss-cs-intros*]:
assumes $a' = \mathfrak{a}_{SS}$ **and** $b' = \mathfrak{a}_{SS}$ **and** $f = \mathfrak{a}_{SS}$
shows $f : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
⟨*proof*⟩

lemma *the-cat-scospan-is-arr-*$\mathfrak{bbb}$[*cat-ss-cs-intros*]:
assumes $a' = \mathfrak{b}_{SS}$ **and** $b' = \mathfrak{b}_{SS}$ **and** $f = \mathfrak{b}_{SS}$
shows $f : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
⟨*proof*⟩

lemma *the-cat-scospan-is-arr-*$\mathfrak{ooo}$[*cat-ss-cs-intros*]:
assumes $a' = \mathfrak{o}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{o}_{SS}$
shows $f : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
⟨*proof*⟩

lemma *the-cat-scospan-is-arr-*$\mathfrak{aog}$[*cat-ss-cs-intros*]:
assumes $a' = \mathfrak{a}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{g}_{SS}$
shows $f : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
⟨*proof*⟩

lemma *the-cat-scospan-is-arr-*$\mathfrak{bof}$[*cat-ss-cs-intros*]:
assumes $a' = \mathfrak{b}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f = \mathfrak{f}_{SS}$
shows $f : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
⟨*proof*⟩

lemma *the-cat-scospan-is-arrE*:
assumes $f' : a' \mapsto_{\rightarrow\bullet\leftarrow_C} b'$
obtains $a' = \mathfrak{a}_{SS}$ **and** $b' = \mathfrak{a}_{SS}$ **and** $f' = \mathfrak{a}_{SS}$
| $a' = \mathfrak{b}_{SS}$ **and** $b' = \mathfrak{b}_{SS}$ **and** $f' = \mathfrak{b}_{SS}$
| $a' = \mathfrak{o}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f' = \mathfrak{o}_{SS}$
| $a' = \mathfrak{a}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f' = \mathfrak{g}_{SS}$
| $a' = \mathfrak{b}_{SS}$ **and** $b' = \mathfrak{o}_{SS}$ **and** $f' = \mathfrak{f}_{SS}$
⟨*proof*⟩

12.3.9 $\rightarrow\bullet\leftarrow$ is a finite category

lemma (**in** $\mathcal{Z}$) *finite-category-the-cat-scospan*[*cat-ss-cs-intros*]:
finite-category α $(\rightarrow\bullet\leftarrow_C)$
⟨*proof*⟩

lemmas [*cat-ss-cs-intros*] = $\mathcal{Z}$.*finite-category-the-cat-scospan*

12.3.10 Duality for $\rightarrow\bullet\leftarrow$ and $\leftarrow\bullet\rightarrow$

lemma *the-cat-scospan-op*[*cat-op-simps*]: *op-cat* $(\rightarrow\bullet\leftarrow_C) = \leftarrow\bullet\rightarrow_C$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *the-cat-sspan-op*[*cat-op-simps*]: *op-cat* $(\leftarrow\bullet\rightarrow_C) = \rightarrow\bullet\leftarrow_C$
⟨*proof*⟩

lemmas [*cat-op-simps*] = $\mathcal{Z}$.*the-cat-sspan-op*

12.3.11 $\leftarrow\bullet\rightarrow$ is a finite category

lemma (**in** $\mathcal{Z}$) *finite-category-the-cat-sspan*[*cat-ss-cs-intros*]:
finite-category α $(\leftarrow\bullet\rightarrow_C)$
⟨*proof*⟩

12.4 Local assumptions for functors from $\rightarrow\bullet\leftarrow$ and $\leftarrow\bullet\rightarrow$

The functors from $\rightarrow\bullet\leftarrow$ and $\leftarrow\bullet\rightarrow$ are introduced as convenient abstractions for the definition of the pullbacks and the pushouts (e.g., see Chapter III-3 and Chapter III-4 in [7]).

12.4.1 Definitions and elementary properties

locale *cf-scospan* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ $\mathfrak{C}$ +
assumes *cf-scospan-*$\mathfrak{g}$[*cat-ss-cs-intros*]: $\mathfrak{g} : \mathfrak{a} \mapsto_{\mathfrak{C}} \mathfrak{o}$
and *cf-scospan-*$\mathfrak{f}$[*cat-ss-cs-intros*]: $\mathfrak{f} : \mathfrak{b} \mapsto_{\mathfrak{C}} \mathfrak{o}$

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{g}'$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{a}$ **and** $b = \mathfrak{o}$
shows $\mathfrak{g} : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{g}''$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}$ **and** $b = \mathfrak{o}$
shows $g : \mathfrak{a} \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{g}'''$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}$ **and** $a = \mathfrak{a}$
shows $g : a \mapsto_{\mathfrak{C}} \mathfrak{o}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{f}'$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{b}$ **and** $b = \mathfrak{o}$
shows $\mathfrak{f} : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{f}''$[*cat-ss-cs-intros*]:
assumes $f = \mathfrak{f}$ **and** $b = \mathfrak{o}$
shows $f : \mathfrak{b} \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-*$\mathfrak{f}'''$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{f}$ **and** $b = \mathfrak{b}$
shows $g : b \mapsto_{\mathfrak{C}} \mathfrak{o}$
⟨*proof*⟩

locale *cf-sspan* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ **and** $\mathfrak{C}$ +
assumes *cf-sspan-*$\mathfrak{g}$[*cat-ss-cs-intros*]: $\mathfrak{g} : \mathfrak{o} \mapsto_{\mathfrak{C}} \mathfrak{a}$
and *cf-sspan-*$\mathfrak{f}$[*cat-ss-cs-intros*]: $\mathfrak{f} : \mathfrak{o} \mapsto_{\mathfrak{C}} \mathfrak{b}$

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{g}'$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}$ **and** $b = \mathfrak{a}$
shows $\mathfrak{g} : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{g}''$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}$ **and** $a = \mathfrak{a}$
shows $g : \mathfrak{o} \mapsto_{\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{g}'''$[*cat-ss-cs-intros*]:
assumes $g = \mathfrak{g}$ **and** $a = \mathfrak{o}$
shows $g : a \mapsto_{\mathfrak{C}} \mathfrak{a}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{f}'$[*cat-ss-cs-intros*]:
assumes $a = \mathfrak{o}$ **and** $b = \mathfrak{b}$
shows $\mathfrak{f} : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{f}''$[*cat-ss-cs-intros*]:
assumes $f = \mathfrak{f}$ **and** $b = \mathfrak{b}$
shows $f : \mathfrak{o} \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-*$\mathfrak{f}'''$[*cat-ss-cs-intros*]:
assumes $f = \mathfrak{f}$ **and** $b = \mathfrak{o}$
shows $f : b \mapsto_{\mathfrak{C}} \mathfrak{b}$
⟨*proof*⟩

Rules.

lemmas (**in** *cf-scospan*) [*cat-ss-cs-intros*] = *cf-scospan-axioms*

mk-ide rf *cf-scospan-def*[*unfolded cf-scospan-axioms-def*]
|*intro cf-scospanI*|
|*dest cf-scospanD*[*dest*]|
|*elim cf-scospanE*[*elim*]|

lemmas [*cat-ss-cs-intros*] = *cf-scospanD*(*1*)

lemmas (**in** *cf-sspan*) [*cat-ss-cs-intros*] = *cf-sspan-axioms*

mk-ide rf *cf-sspan-def*[*unfolded cf-sspan-axioms-def*]
|*intro cf-sspanI*|
|*dest cf-sspanD*[*dest*]|
|*elim cf-sspanE*[*elim*]|

Duality.

lemma (**in** *cf-scospan*) *cf-sspan-op*[*cat-op-intros*]:
cf-sspan α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cf-scospan.cf-sspan-op*

lemma (**in** *cf-sspan*) *cf-scospan-op*[*cat-op-intros*]:
cf-scospan α $\mathfrak{a}$ $\mathfrak{g}$ $\mathfrak{o}$ $\mathfrak{f}$ $\mathfrak{b}$ (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cf-sspan.cf-scospan-op*

12.5 Functors from →•← and ←•→

12.5.1 Definition and elementary properties

definition *the-cf-scospan* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
($\langle\langle$-→-→-←-←-$\rangle_{CF^1}\rangle$ [*51*, *51*, *51*, *51*, *51*] *999*)
where $\langle\mathfrak{a}{\to}\mathfrak{g}{\to}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b}\rangle_{CF}\mathfrak{C}$ =
[
(
$\lambda a{\in_\circ}{\to}{\bullet}{\leftarrow}_C(\!|Obj|\!)$.
if $a = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}$
| $a = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}$
| *otherwise* $\Rightarrow \mathfrak{o}$
),
(
$\lambda f{\in_\circ}{\to}{\bullet}{\leftarrow}_C(\!|Arr|\!)$.
if $f = \mathfrak{a}_{SS} \Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}|\!)$
| $f = \mathfrak{b}_{SS} \Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}|\!)$
| $f = \mathfrak{g}_{SS} \Rightarrow \mathfrak{g}$
| $f = \mathfrak{f}_{SS} \Rightarrow \mathfrak{f}$
| *otherwise* $\Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{o}|\!)$
),
${\to}{\bullet}{\leftarrow}_C$,
$\mathfrak{C}$
]$_\circ$

definition *the-cf-sspan* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
($\langle\langle$-←-←-→-→-$\rangle_{CF^1}\rangle$ [*51*, *51*, *51*, *51*, *51*] *999*)
where $\langle\mathfrak{a}{\leftarrow}\mathfrak{g}{\leftarrow}\mathfrak{o}{\to}\mathfrak{f}{\to}\mathfrak{b}\rangle_{CF}\mathfrak{C}$ =
[
(
$\lambda a{\in_\circ}{\leftarrow}{\bullet}{\to}_C(\!|Obj|\!)$.
if $a = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}$
| $a = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}$
| *otherwise* $\Rightarrow \mathfrak{o}$
),
(
$\lambda f{\in_\circ}{\leftarrow}{\bullet}{\to}_C(\!|Arr|\!)$.
if $f = \mathfrak{a}_{SS} \Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}|\!)$
| $f = \mathfrak{b}_{SS} \Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}|\!)$
| $f = \mathfrak{g}_{SS} \Rightarrow \mathfrak{g}$
| $f = \mathfrak{f}_{SS} \Rightarrow \mathfrak{f}$
| *otherwise* $\Rightarrow \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{o}|\!)$
),
${\leftarrow}{\bullet}{\to}_C$,
$\mathfrak{C}$
]$_\circ$

Components.

lemma *the-cf-scospan-components*:
shows $\langle\mathfrak{a}{\to}\mathfrak{g}{\to}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b}\rangle_{CF}\mathfrak{C}(\!|ObjMap|\!)$ =
(
$\lambda a{\in_\circ}{\to}{\bullet}{\leftarrow}_C(\!|Obj|\!)$.
if $a = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}$
| $a = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}$
| *otherwise* $\Rightarrow \mathfrak{o}$
)
and $\langle\mathfrak{a}{\to}\mathfrak{g}{\to}\mathfrak{o}{\leftarrow}\mathfrak{f}{\leftarrow}\mathfrak{b}\rangle_{CF}\mathfrak{C}(\!|ArrMap|\!)$ =
(

$\lambda f \in_\circ \to\bullet\leftarrow_C (\!| Arr |\!).$
$\quad$ *if* $f = \mathfrak{a}_{SS} \Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{a} |\!)$
$\quad$ | $f = \mathfrak{b}_{SS} \Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{b} |\!)$
$\quad$ | $f = \mathfrak{g}_{SS} \Rightarrow \mathfrak{g}$
$\quad$ | $f = \mathfrak{f}_{SS} \Rightarrow \mathfrak{f}$
$\quad$ | *otherwise* $\Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{o} |\!)$
)
and [*cat-ss-cs-simps*]: $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| HomDom |\!) = \to\bullet\leftarrow_C$
and [*cat-ss-cs-simps*]: $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| HomCod |\!) = \mathfrak{C}$
⟨*proof*⟩

lemma *the-cf-sspan-components*:
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| ObjMap |\!) =$
(
$\lambda a \in_\circ \leftarrow\bullet\to_C (\!| Obj |\!).$
$\quad$ *if* $a = \mathfrak{a}_{SS} \Rightarrow \mathfrak{a}$
$\quad$ | $a = \mathfrak{b}_{SS} \Rightarrow \mathfrak{b}$
$\quad$ | *otherwise* $\Rightarrow \mathfrak{o}$
)
and $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| ArrMap |\!) =$
(
$\lambda f \in_\circ \leftarrow\bullet\to_C (\!| Arr |\!).$
$\quad$ *if* $f = \mathfrak{a}_{SS} \Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{a} |\!)$
$\quad$ | $f = \mathfrak{b}_{SS} \Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{b} |\!)$
$\quad$ | $f = \mathfrak{g}_{SS} \Rightarrow \mathfrak{g}$
$\quad$ | $f = \mathfrak{f}_{SS} \Rightarrow \mathfrak{f}$
$\quad$ | *otherwise* $\Rightarrow \mathfrak{C}(\!| CId |\!)(\!| \mathfrak{o} |\!)$
)
and [*cat-ss-cs-simps*]: $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| HomDom |\!) = \leftarrow\bullet\to_C$
and [*cat-ss-cs-simps*]: $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| HomCod |\!) = \mathfrak{C}$
⟨*proof*⟩

Elementary properties.

lemma *the-cf-scospan-components-vsv*[*cat-ss-cs-intros*]: *vsv* $(\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}})$
⟨*proof*⟩

lemma *the-cf-sspan-components-vsv*[*cat-ss-cs-intros*]: *vsv* $(\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}})$
⟨*proof*⟩

12.5.2 Object map.

mk-VLambda *the-cf-scospan-components*(*1*)
|*vsv the-cf-scospan-ObjMap-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cf-scospan-ObjMap-vdomain*[*cat-ss-cs-simps*]|
|*app the-cf-scospan-ObjMap-app*|

lemma *the-cf-scospan-ObjMap-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{a}_{SS}$
shows $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| ObjMap |\!)(\!| x |\!) = \mathfrak{a}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ObjMap-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{b}_{SS}$
shows $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\!| ObjMap |\!)(\!| x |\!) = \mathfrak{b}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ObjMap-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{o}_{SS}$

shows $\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = \mathfrak{o}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ObjMap-vrange*:
$\mathcal{R}_{\circ}\ (\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$
⟨*proof*⟩

mk-VLambda *the-cf-sspan-components*(*1*)
|*vsv the-cf-sspan-ObjMap-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cf-sspan-ObjMap-vdomain*[*cat-ss-cs-simps*]|
|*app the-cf-sspan-ObjMap-app*|

lemma *the-cf-sspan-ObjMap-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{a}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \rightarrow \mathfrak{f} \rightarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = \mathfrak{a}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ObjMap-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{b}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \rightarrow \mathfrak{f} \rightarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = \mathfrak{b}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ObjMap-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $x = \mathfrak{o}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \rightarrow \mathfrak{f} \rightarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)(\lfloor x \rfloor) = \mathfrak{o}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ObjMap-vrange*:
$\mathcal{R}_{\circ}\ (\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \rightarrow \mathfrak{f} \rightarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$
⟨*proof*⟩

12.5.3 Arrow map.

mk-VLambda *the-cf-scospan-components*(*2*)
|*vsv the-cf-scospan-ArrMap-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cf-scospan-ArrMap-vdomain*[*cat-ss-cs-simps*]|
|*app the-cf-scospan-ArrMap-app*|

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{o} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{a} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{b} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\langle \mathfrak{a} \rightarrow \mathfrak{g} \rightarrow \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{g}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$
shows $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{f}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *the-cf-scospan-ArrMap-vrange*:
$\mathcal{R}_\circ\ (\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

mk-VLambda *the-cf-sspan-components*(*2*)
|*vsv the-cf-sspan-ArrMap-vsv*[*cat-ss-cs-intros*]|
|*vdomain the-cf-sspan-ArrMap-vdomain*[*cat-ss-cs-simps*]|
|*app the-cf-sspan-ArrMap-app*|

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-app-*$\mathfrak{o}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{o}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{o} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-app-*$\mathfrak{a}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{a}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{a} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-app-*$\mathfrak{b}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{b}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{C}(\lfloor CId \rfloor)(\lfloor \mathfrak{b} \rfloor)$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-app-*$\mathfrak{g}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{g}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{g}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-app-*$\mathfrak{f}$[*cat-ss-cs-simps*]:
assumes $f = \mathfrak{f}_{SS}$
shows $\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{f}$
⟨*proof*⟩

lemma (**in** *cf-sspan*) *the-cf-sspan-ArrMap-vrange*:
$\mathcal{R}_\circ\ (\langle \mathfrak{a} \leftarrow \mathfrak{g} \leftarrow \mathfrak{o} \to \mathfrak{f} \to \mathfrak{b} \rangle_{CF\mathfrak{C}}(\lfloor ArrMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

12.5.4 Functor from →·← is a functor

lemma (**in** *cf-scospan*) *cf-scospan-the-cf-scospan-is-tm-functor*:
$\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}} : \to\cdot\leftarrow_C \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *cf-scospan*) *cf-scospan-the-cf-scospan-is-tm-functor'*:
assumes $\mathfrak{A}' = \to\cdot\leftarrow_C$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\langle \mathfrak{a} \to \mathfrak{g} \to \mathfrak{o} \leftarrow \mathfrak{f} \leftarrow \mathfrak{b} \rangle_{CF\mathfrak{C}} : \mathfrak{A}' \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-ss-cs-intros*] = *cf-scospan.cf-scospan-the-cf-scospan-is-tm-functor*

12.5.5 Duality for the functors from →•← and ←•→

lemma *op-cf-cf-scospan*[*cat-op-simps*]:
  op-cf (⟨𝔞→𝔤→𝔬←𝔣←𝔟⟩$_{CF}$𝔈) = ⟨𝔞←𝔤←𝔬→𝔣→𝔟⟩$_{CF\,op\text{-}cat}$ 𝔈
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *op-cf-cf-scospan*[*cat-op-simps*]:
  op-cf (⟨𝔞←𝔤←𝔬→𝔣→𝔟⟩$_{CF}$𝔈) = ⟨𝔞→𝔤→𝔬←𝔣←𝔟⟩$_{CF\,op\text{-}cat}$ 𝔈
⟨*proof*⟩

lemmas [*cat-op-simps*] = $\mathcal{Z}$.*op-cf-cf-scospan*

12.5.6 Functor from ←•→ is a functor

lemma (**in** *cf-sspan*) *cf-sspan-the-cf-sspan-is-tm-functor*:
  ⟨𝔞←𝔤←𝔬→𝔣→𝔟⟩$_{CF}$𝔈 : ←•→$_{C}$ ↦↦$_{C.tm\alpha}$ 𝔈
⟨*proof*⟩

lemma (**in** *cf-sspan*) *cf-sspan-the-cf-sspan-is-tm-functor′*:
  assumes 𝔄′ = ←•→$_{C}$ **and** 𝔈′ = 𝔈
  shows ⟨𝔞←𝔤←𝔬→𝔣→𝔟⟩$_{CF}$𝔈 : 𝔄′ ↦↦$_{C.tm\alpha}$ 𝔈′
  ⟨*proof*⟩

lemmas [*cat-ss-cs-intros*] = *cf-sspan.cf-sspan-the-cf-sspan-is-tm-functor*

13 Categories with parallel arrows between two objects

13.1 Background: category with parallel arrows between two objects

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

definition $\mathfrak{a}_{PL} :: V \Rightarrow V$ **where** $\mathfrak{a}_{PL}\ F = set\ \{F, 0\}$
definition $\mathfrak{b}_{PL} :: V \Rightarrow V$ **where** $\mathfrak{b}_{PL}\ F = set\ \{F, 1_{\mathbb{N}}\}$

lemma *cat-PL-*$\mathfrak{a}$*-nin-F*[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL}\ F \notin_{\circ} F$
⟨*proof*⟩

lemma *cat-PL-*$\mathfrak{b}$*-nin-F*[*cat-parallel-cs-intros*]: $\mathfrak{b}_{PL}\ F \notin_{\circ} F$
⟨*proof*⟩

lemma *cat-PL-*$\mathfrak{ab}$[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL}\ F \neq \mathfrak{b}_{PL}\ F$
⟨*proof*⟩

lemmas *cat-PL-*$\mathfrak{ba}$[*cat-parallel-cs-intros*] = *cat-PL-*$\mathfrak{ab}$[*symmetric*]

13.2 Composable arrows for a category with parallel arrows between two objects

definition *cat-parallel-composable* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F \equiv$
$set\ \{[\mathfrak{a}, \mathfrak{a}]_{\circ}, [\mathfrak{b}, \mathfrak{b}]_{\circ}\} \cup_{\circ}$
$(F \times_{\bullet} set\ \{\mathfrak{a}\}) \cup_{\circ}$
$(set\ \{\mathfrak{b}\} \times_{\bullet} F)$

Rules.

lemma *cat-parallel-composable-*$\mathfrak{aa}$[*cat-parallel-cs-intros*]:
assumes $g = \mathfrak{a}$ **and** $f = \mathfrak{a}$
shows $[g, f]_{\circ} \in_{\circ}$ *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F$
⟨*proof*⟩

lemma *cat-parallel-composable-*$\mathfrak{bf}$[*cat-parallel-cs-intros*]:
assumes $g = \mathfrak{b}$ **and** $f \in_{\circ} F$
shows $[g, f]_{\circ} \in_{\circ}$ *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F$
⟨*proof*⟩

lemma *cat-parallel-composable-*$\mathfrak{fa}$[*cat-parallel-cs-intros*]:
assumes $g \in_{\circ} F$ **and** $f = \mathfrak{a}$
shows $[g, f]_{\circ} \in_{\circ}$ *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F$
⟨*proof*⟩

lemma *cat-parallel-composable-*$\mathfrak{bb}$[*cat-parallel-cs-intros*]:
assumes $g = \mathfrak{b}$ **and** $f = \mathfrak{b}$
shows $[g, f]_{\circ} \in_{\circ}$ *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F$
⟨*proof*⟩

lemma *cat-parallel-composableE*:
assumes $[g, f]_{\circ} \in_{\circ}$ *cat-parallel-composable* $\mathfrak{a}\ \mathfrak{b}\ F$
obtains $g = \mathfrak{b}$ **and** $f = \mathfrak{b}$
| $g = \mathfrak{b}$ **and** $f \in_{\circ} F$
| $g \in_{\circ} F$ **and** $f = \mathfrak{a}$
| $g = \mathfrak{a}$ **and** $f = \mathfrak{a}$
⟨*proof*⟩

Elementary properties.

lemma *cat-parallel-composable-fconverse*:
(*cat-parallel-composable* $\mathfrak{a}$ $\mathfrak{b}$ F)$^{-1}_{\bullet}$ = *cat-parallel-composable* $\mathfrak{b}$ $\mathfrak{a}$ F
⟨*proof*⟩

13.3 Local assumptions for a category with parallel arrows between two objects

locale *cat-parallel* = $\mathcal{Z}$ α **for** α +
fixes $\mathfrak{a}$ $\mathfrak{b}$ F
assumes *cat-parallel-*$\mathfrak{ab}$[*cat-parallel-cs-intros*]: $\mathfrak{a} \neq \mathfrak{b}$
and *cat-parallel-*$\mathfrak{a}F$[*cat-parallel-cs-intros*]: $\mathfrak{a} \notin_{\circ} F$
and *cat-parallel-*$\mathfrak{b}F$[*cat-parallel-cs-intros*]: $\mathfrak{b} \notin_{\circ} F$
and *cat-parallel-*$\mathfrak{a}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{a} \in_{\circ} Vset\ \alpha$
and *cat-parallel-*$\mathfrak{b}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{b} \in_{\circ} Vset\ \alpha$
and *cat-parallel-F-in-Vset*[*cat-parallel-cs-intros*]: $F \in_{\circ} Vset\ \alpha$

lemmas (**in** *cat-parallel*) *cat-parallel-ineq* =
cat-parallel-$\mathfrak{ab}$
cat-parallel-$\mathfrak{a}F$
cat-parallel-$\mathfrak{b}F$

Rules.

lemmas (**in** *cat-parallel*) [*cat-parallel-cs-intros*] = *cat-parallel-axioms*

mk-ide rf *cat-parallel-def*[*unfolded cat-parallel-axioms-def*]
|*intro cat-parallelI*|
|*dest cat-parallelD*[*dest*]|
|*elim cat-parallelE*[*elim*]|

Duality.

lemma (**in** *cat-parallel*) *cat-parallel-op*[*cat-op-intros*]:
cat-parallel α $\mathfrak{b}$ $\mathfrak{a}$ F
⟨*proof*⟩

Elementary properties.

lemma (**in** $\mathcal{Z}$) *cat-parallel-PL*:
assumes $F \in_{\circ} Vset\ \alpha$
shows *cat-parallel* α ($\mathfrak{a}_{PL}$ F) ($\mathfrak{b}_{PL}$ F) F
⟨*proof*⟩

13.4 $\Uparrow$: category with parallel arrows between two objects

13.4.1 Definition and elementary properties

See Chapter I-2 and Chapter III-3 in [7].

definition *the-cat-parallel* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ ($\langle\Uparrow_C\rangle$)
where $\Uparrow_C$ $\mathfrak{a}$ $\mathfrak{b}$ F =
[
set {$\mathfrak{a}$, $\mathfrak{b}$},
set {$\mathfrak{a}$, $\mathfrak{b}$} $\cup_{\circ}$ F,
($\lambda x \in_{\circ} set\ \{\mathfrak{a}, \mathfrak{b}\} \cup_{\circ} F.\ (x = \mathfrak{b}\ ?\ \mathfrak{b} : \mathfrak{a})$),
($\lambda x \in_{\circ} set\ \{\mathfrak{a}, \mathfrak{b}\} \cup_{\circ} F.\ (x = \mathfrak{a}\ ?\ \mathfrak{a} : \mathfrak{b})$),
(
$\lambda gf \in_{\circ} cat\text{-}parallel\text{-}composable\ \mathfrak{a}\ \mathfrak{b}\ F.$
if $gf = [\mathfrak{b}, \mathfrak{b}]_{\circ} \Rightarrow \mathfrak{b}$

```
      | ∃f. gf = [𝔟, f]∘ ⇒ gf(⟦1ℕ⟧)
      | ∃f. gf = [f, 𝔞]∘ ⇒ gf(⟦0⟧)
      | otherwise ⇒ 𝔞
    ),
    vid-on (set {𝔞, 𝔟})
  ]∘
```

Components.

lemma *the-cat-parallel-components*:
shows $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Obj \rfloor) = set\ \{\mathfrak{a}, \mathfrak{b}\}$
and $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Arr \rfloor) = set\ \{\mathfrak{a}, \mathfrak{b}\} \cup_\circ F$
and $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Dom \rfloor) = (\lambda x \in_\circ set\ \{\mathfrak{a}, \mathfrak{b}\} \cup_\circ F.\ (x = \mathfrak{b}\ ?\ \mathfrak{b} : \mathfrak{a}))$
and $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Cod \rfloor) = (\lambda x \in_\circ set\ \{\mathfrak{a}, \mathfrak{b}\} \cup_\circ F.\ (x = \mathfrak{a}\ ?\ \mathfrak{a} : \mathfrak{b}))$
and $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Comp \rfloor) =$

```
  (
    λgf∈∘ cat-parallel-composable 𝔞 𝔟 F.
     if gf = [𝔟, 𝔟]∘ ⇒ 𝔟
      | ∃f. gf = [𝔟, f]∘ ⇒ gf(⟦1ℕ⟧)
      | ∃f. gf = [f, 𝔞]∘ ⇒ gf(⟦0⟧)
      | otherwise ⇒ 𝔞
  )
```

and $\Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor CId \rfloor) = vid\text{-}on\ (set\ \{\mathfrak{a}, \mathfrak{b}\})$
⟨*proof*⟩

13.4.2 Objects

lemma *the-cat-parallel-Obj-𝔞I*[*cat-parallel-cs-intros*]:
assumes $a = \mathfrak{a}$
shows $a \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *the-cat-parallel-Obj-𝔟I*[*cat-parallel-cs-intros*]:
assumes $a = \mathfrak{b}$
shows $a \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *the-cat-parallel-ObjE*:
assumes $a \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Obj \rfloor)$
obtains $a = \mathfrak{a} \mid a = \mathfrak{b}$
⟨*proof*⟩

13.4.3 Arrows

lemma *the-cat-parallel-Arr-𝔞I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{a}$
shows $f \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *the-cat-parallel-Arr-𝔟I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{b}$
shows $f \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *the-cat-parallel-Arr-FI*[*cat-parallel-cs-intros*]:
assumes $f \in_\circ F$
shows $f \in_\circ \Uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *the-cat-parallel-ArrE*:
assumes $f \in_{\circ} \Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Arr|\!)$
obtains $f = \mathfrak{a} \mid f = \mathfrak{b} \mid f \in_{\circ} F$
⟨*proof*⟩

13.4.4 Domain

mk-VLambda *the-cat-parallel-components*(*3*)
|*vsv the-cat-parallel-Dom-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cat-parallel-Dom-vdomain*[*cat-parallel-cs-simps*]|

lemma (**in** *cat-parallel*) *the-cat-parallel-Dom-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Dom|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Dom-app-*$\mathfrak{b}$

lemma (**in** *cat-parallel*) *the-cat-parallel-Dom-app-F*[*cat-parallel-cs-simps*]:
assumes $f \in_{\circ} F$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Dom-app-F*

lemma (**in** *cat-parallel*) *the-cat-parallel-Dom-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Dom-app-*$\mathfrak{a}$

13.4.5 Codomain

mk-VLambda *the-cat-parallel-components*(*4*)
|*vsv the-cat-parallel-Cod-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cat-parallel-Cod-vdomain*[*cat-parallel-cs-simps*]|

lemma (**in** *cat-parallel*) *the-cat-parallel-Cod-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Cod-app-*$\mathfrak{b}$

lemma (**in** *cat-parallel*) *the-cat-parallel-Cod-app-F*[*cat-parallel-cs-simps*]:
assumes $f \in_{\circ} F$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Cod-app-F*

lemma (**in** *cat-parallel*) *the-cat-parallel-Cod-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F(\!|Cod|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel.the-cat-parallel-Cod-app-*$\mathfrak{a}$

13.4.6 Composition

mk-VLambda *the-cat-parallel-components*(*5*)
|*vsv the-cat-parallel-Comp-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cat-parallel-Comp-vdomain*[*cat-parallel-cs-simps*]|
|*app the-cat-parallel-Comp-app*[*cat-parallel-cs-simps*]|

lemma *the-cat-parallel-Comp-app-𝔟𝔟*[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{b}$ **and** $f = \mathfrak{b}$
shows $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = g$ $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = f$
⟨*proof*⟩

lemma *the-cat-parallel-Comp-app-𝔞𝔞*[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{a}$ **and** $f = \mathfrak{a}$
shows $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = g$ $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = f$
⟨*proof*⟩

lemma *the-cat-parallel-Comp-app-𝔟F*[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{b}$ **and** $f \in_\circ F$
shows $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = f$
⟨*proof*⟩

lemma (**in** *cat-parallel*) *the-cat-parallel-Comp-app-F𝔞*[*cat-parallel-cs-simps*]:
assumes $g \in_\circ F$ **and** $f = \mathfrak{a}$
shows $g \circ_{A\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} f = g$
⟨*proof*⟩

13.4.7 Identity

mk-VLambda *the-cat-parallel-components*(*6*)[*unfolded VLambda-vid-on*[*symmetric*]]
|*vsv the-cat-parallel-CId-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cat-parallel-CId-vdomain*[*cat-parallel-cs-simps*]|
|*app the-cat-parallel-CId-app*|

lemma *the-cat-parallel-CId-app-𝔞*[*cat-parallel-cs-simps*]:
assumes $a = \mathfrak{a}$
shows $\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor CId \rfloor)(\lfloor a \rfloor) = \mathfrak{a}$
⟨*proof*⟩

lemma *the-cat-parallel-CId-app-𝔟*[*cat-parallel-cs-simps*]:
assumes $a = \mathfrak{b}$
shows $\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F(\lfloor CId \rfloor)(\lfloor a \rfloor) = \mathfrak{b}$
⟨*proof*⟩

13.4.8 Arrow with a domain and a codomain

lemma (**in** *cat-parallel*) *the-cat-parallel-is-arr-𝔞𝔞𝔞*[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{a}$ **and** $b' = \mathfrak{a}$ **and** $f = \mathfrak{a}$
shows $f : a' \mapsto_{\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel*) *the-cat-parallel-is-arr-𝔟𝔟𝔟*[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{b}$ **and** $b' = \mathfrak{b}$ **and** $f = \mathfrak{b}$
shows $f : a' \mapsto_{\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel*) *the-cat-parallel-is-arr-𝔞𝔟F*[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f \in_\circ F$

shows $f : a' \mapsto_{\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel*) *the-cat-parallel-is-arrE*:
assumes $f' : a' \mapsto_{\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F} b'$
obtains $a' = \mathfrak{a}$ **and** $b' = \mathfrak{a}$ **and** $f' = \mathfrak{a}$
| $a' = \mathfrak{b}$ **and** $b' = \mathfrak{b}$ **and** $f' = \mathfrak{b}$
| $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f' \in_\circ F$
⟨*proof*⟩

13.4.9 $\Vert$ is a category

lemma (**in** *cat-parallel*) *tiny-category-the-cat-parallel*[*cat-parallel-cs-intros*]:
tiny-category α $(\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F)$
⟨*proof*⟩

lemmas [*cat-parallel-cs-intros*] = *cat-parallel.tiny-category-the-cat-parallel*

13.4.10 Opposite parallel category

lemma (**in** *cat-parallel*) *op-cat-the-cat-parallel*[*cat-op-simps*]:
op-cat $(\Vert_C\ \mathfrak{a}\ \mathfrak{b}\ F) = \Vert_C\ \mathfrak{b}\ \mathfrak{a}\ F$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *cat-parallel.op-cat-the-cat-parallel*

13.5 Parallel functor

13.5.1 Background

See Chapter III-3 and Chapter III-4 in [7]).

13.5.2 Local assumptions for the parallel functor

locale *cf-parallel* = *cat-parallel* α $\mathfrak{a}$ $\mathfrak{b}$ F + *category* α $\mathfrak{C}$ + F': *vsv* F'
for α $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F' $\mathfrak{C}$ :: V +
assumes *cf-parallel-F'-vdomain*[*cat-parallel-cs-simps*]: $\mathcal{D}_\circ\ F' = F$
and *cf-parallel-F'*[*cat-parallel-cs-intros*]: $\mathfrak{f} \in_\circ F \Longrightarrow F'(\!|\mathfrak{f}|\!) : \mathfrak{a}' \mapsto_{\mathfrak{C}} \mathfrak{b}'$
and *cf-parallel-*$\mathfrak{a}'$[*cat-parallel-cs-intros*]: $\mathfrak{a}' \in_\circ \mathfrak{C}(\!|Obj|\!)$
and *cf-parallel-*$\mathfrak{b}'$[*cat-parallel-cs-intros*]: $\mathfrak{b}' \in_\circ \mathfrak{C}(\!|Obj|\!)$

lemmas (**in** *cf-parallel*) [*cat-parallel-cs-intros*] = *F'.vsv-axioms*

lemma (**in** *cf-parallel*) *cf-parallel-F''*[*cat-parallel-cs-intros*]:
assumes $\mathfrak{f} \in_\circ F$ **and** $a = \mathfrak{a}'$ **and** $b = \mathfrak{b}'$
shows $F'(\!|\mathfrak{f}|\!) : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-parallel*) *cf-parallel-F'''*[*cat-parallel-cs-intros*]:
assumes $\mathfrak{f} \in_\circ F$ **and** $f = F'(\!|\mathfrak{f}|\!)$ **and** $b = \mathfrak{b}'$
shows $f : \mathfrak{a}' \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cf-parallel*) *cf-parallel-F''''*[*cat-parallel-cs-intros*]:
assumes $\mathfrak{f} \in_\circ F$ **and** $f = F'(\!|\mathfrak{f}|\!)$ **and** $a = \mathfrak{a}'$
shows $f : a \mapsto_{\mathfrak{C}} \mathfrak{b}'$
⟨*proof*⟩

Rules.

lemma (in *cf-parallel*) *cf-parallel-axioms'*[*cat-parallel-cs-intros*]:
assumes $\alpha' = \alpha$
and $a'' = \mathfrak{a}$
and $b'' = \mathfrak{b}$
and $F'' = F$
and $a''' = \mathfrak{a}'$
and $b''' = \mathfrak{b}'$
and $F''' = F'$
shows *cf-parallel* α' a'' b'' F'' a''' b''' F''' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cf-parallel-def*[*unfolded cf-parallel-axioms-def*]
|*intro cf-parallelI*|
|*dest cf-parallelD*[*dest*]|
|*elim cf-parallelE*[*elim*]|

lemmas [*cat-parallel-cs-intros*] = *cf-parallelD*(*1*,*2*)

Duality.

lemma (in *cf-parallel*) *cf-parallel-op*[*cat-op-intros*]:
cf-parallel α $\mathfrak{b}$ $\mathfrak{a}$ F $\mathfrak{b}'$ $\mathfrak{a}'$ F' (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cf-parallel.cf-parallel-op*

13.5.3 Definition and elementary properties

definition *the-cf-parallel* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
(‹⇑→⇑$_{CF}$›)
where ⇑→⇑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F' =
[
(λa∈$_\circ$⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ F(|*Obj*|). (a = $\mathfrak{a}$? $\mathfrak{a}'$: $\mathfrak{b}'$)),
(
λf∈$_\circ$⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ F(|*Arr*|).
(
if f = $\mathfrak{a}$ ⇒ $\mathfrak{C}$(|*CId*|)(|$\mathfrak{a}'$|)
| f = $\mathfrak{b}$ ⇒ $\mathfrak{C}$(|*CId*|)(|$\mathfrak{b}'$|)
| *otherwise* ⇒ F'(|f|)
)
),
⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ F,
$\mathfrak{C}$
]$_\circ$

Components.

lemma *the-cf-parallel-components*:
shows ⇑→⇑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F'(|*ObjMap*|) =
(λa∈$_\circ$⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ F(|*Obj*|). (a = $\mathfrak{a}$? $\mathfrak{a}'$: $\mathfrak{b}'$))
and ⇑→⇑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F'(|*ArrMap*|) =
(
λf∈$_\circ$⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ F(|*Arr*|).
(
if f = $\mathfrak{a}$ ⇒ $\mathfrak{C}$(|*CId*|)(|$\mathfrak{a}'$|)
| f = $\mathfrak{b}$ ⇒ $\mathfrak{C}$(|*CId*|)(|$\mathfrak{b}'$|)
| *otherwise* ⇒ F'(|f|)
)

)
and [*cat-parallel-cs-simps*]: $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|HomDom|\!) = \Uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ F$
and [*cat-parallel-cs-simps*]: $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|HomCod|\!) = \mathfrak{C}$
⟨*proof*⟩

13.5.4 Object map

mk-VLambda *the-cf-parallel-components*(*1*)
|*vsv the-cf-parallel-ObjMap-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cf-parallel-ObjMap-vdomain*[*cat-parallel-cs-simps*]|
|*app the-cf-parallel-ObjMap-app*|

lemma (**in** *cf-parallel*) *the-cf-parallel-ObjMap-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $x = \mathfrak{a}$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ObjMap|\!)(\!|x|\!) = \mathfrak{a}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel.the-cf-parallel-ObjMap-app-*$\mathfrak{a}$

lemma (**in** *cf-parallel*) *the-cf-parallel-ObjMap-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $x = \mathfrak{b}$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ObjMap|\!)(\!|x|\!) = \mathfrak{b}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel.the-cf-parallel-ObjMap-app-*$\mathfrak{b}$

lemma (**in** *cf-parallel*) *the-cf-parallel-ObjMap-vrange*:
$\mathcal{R}_{\circ}\ (\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ObjMap|\!)) = set\ \{\mathfrak{a}',\ \mathfrak{b}'\}$
⟨*proof*⟩

lemma (**in** *cf-parallel*) *the-cf-parallel-ObjMap-vrange-vsubset-Obj*:
$\mathcal{R}_{\circ}\ (\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ObjMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

13.5.5 Arrow map

mk-VLambda *the-cf-parallel-components*(*2*)
|*vsv the-cf-parallel-ArrMap-vsv*[*cat-parallel-cs-intros*]|
|*vdomain the-cf-parallel-ArrMap-vdomain*[*cat-parallel-cs-simps*]|
|*app the-cf-parallel-ArrMap-app*|

lemma (**in** *cf-parallel*) *the-cf-parallel-ArrMap-app-F*[*cat-parallel-cs-simps*]:
assumes $f \in_{\circ} F$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ArrMap|\!)(\!|f|\!) = F'(\!|f|\!)$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel.the-cf-parallel-ArrMap-app-F*

lemma (**in** *cf-parallel*) *the-cf-parallel-ArrMap-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}'|\!)$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel.the-cf-parallel-ArrMap-app-*$\mathfrak{a}$

lemma (**in** *cf-parallel*) *the-cf-parallel-ArrMap-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ F\ \mathfrak{a}'\ \mathfrak{b}'\ F'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}'|\!)$

⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel.the-cf-parallel-ArrMap-app-*$\mathfrak{b}$

lemma (**in** *cf-parallel*) *the-cf-parallel-ArrMap-vrange*:
$\mathcal{R}_\circ$ ($\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ $F'(\!|ArrMap|\!)$) = (F' ‘$_\circ$ F) $\cup_\circ$ *set* {$\mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}'|\!)$, $\mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}'|\!)$}
(**is** ‹$\mathcal{R}_\circ$ ($\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ $F'(\!|ArrMap|\!)$) = *?FF* $\cup_\circ$ *?CID*›)
⟨*proof*⟩

lemma (**in** *cf-parallel*) *the-cf-parallel-ArrMap-vrange-vsubset-Arr*:
$\mathcal{R}_\circ$ ($\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ $F'(\!|ArrMap|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

13.5.6 Parallel functor is a functor

lemma (**in** *cf-parallel*) *cf-parallel-the-cf-parallel-is-tm-functor*:
$\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F' : $\Uparrow_C$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mapsto\!\!\mapsto_{C.tm\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *cf-parallel*) *cf-parallel-the-cf-parallel-is-tm-functor′*:
assumes $\mathfrak{A}'$ = $\Uparrow_C$ $\mathfrak{a}$ $\mathfrak{b}$ F **and** $\mathfrak{C}'$ = $\mathfrak{C}$
shows $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F' : $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{C.tm\alpha}$ $\mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-intros*] =
cf-parallel.cf-parallel-the-cf-parallel-is-tm-functor′

13.5.7 Opposite parallel functor

lemma (**in** *cf-parallel*) *cf-parallel-the-cf-parallel-op*[*cat-op-simps*]:
op-cf ($\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ F $\mathfrak{a}'$ $\mathfrak{b}'$ F') = $\Uparrow\!\!\rightarrow\!\!\Uparrow_{CF}$ (*op-cat* $\mathfrak{C}$) $\mathfrak{b}$ $\mathfrak{a}$ F $\mathfrak{b}'$ $\mathfrak{a}'$ F'
⟨*proof*⟩

lemmas [*cat-op-simps*] = *cf-parallel.cf-parallel-the-cf-parallel-op*

13.6 Background for the definition of a category with two parallel arrows between two objects

The case of two parallel arrows between two objects is treated explicitly because it is prevalent in applications.

definition $\mathfrak{g}_{PL}$ :: V **where** $\mathfrak{g}_{PL}$ = *0*
definition $\mathfrak{f}_{PL}$ :: V **where** $\mathfrak{f}_{PL}$ = $1_{\mathbb{N}}$

definition $\mathfrak{a}_{PL2}$ :: V **where** $\mathfrak{a}_{PL2}$ = $\mathfrak{a}_{PL}$ (*set* {$\mathfrak{g}_{PL}$, $\mathfrak{f}_{PL}$})
definition $\mathfrak{b}_{PL2}$ :: V **where** $\mathfrak{b}_{PL2}$ = $\mathfrak{b}_{PL}$ (*set* {$\mathfrak{g}_{PL}$, $\mathfrak{f}_{PL}$})

lemma *cat-PL2-ineq*:
shows *cat-PL2-*$\mathfrak{ab}$[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL2}$ ≠ $\mathfrak{b}_{PL2}$
and *cat-PL2-*$\mathfrak{ag}$[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL2}$ ≠ $\mathfrak{g}_{PL}$
and *cat-PL2-*$\mathfrak{af}$[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL2}$ ≠ $\mathfrak{f}_{PL}$
and *cat-PL2-*$\mathfrak{bg}$[*cat-parallel-cs-intros*]: $\mathfrak{b}_{PL2}$ ≠ $\mathfrak{g}_{PL}$
and *cat-PL2-*$\mathfrak{bf}$[*cat-parallel-cs-intros*]: $\mathfrak{b}_{PL2}$ ≠ $\mathfrak{f}_{PL}$
and *cat-PL2-*$\mathfrak{gf}$[*cat-parallel-cs-intros*]: $\mathfrak{g}_{PL}$ ≠ $\mathfrak{f}_{PL}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$)
shows *cat-PL2-*$\mathfrak{a}$[*cat-parallel-cs-intros*]: $\mathfrak{a}_{PL2}$ $\in_\circ$ *Vset* α

and *cat-PL2-*$\mathfrak{b}$[*cat-parallel-cs-intros*]: $\mathfrak{b}_{PL2} \in_{\circ} Vset\ \alpha$
and *cat-PL2-*$\mathfrak{g}$[*cat-parallel-cs-intros*]: $\mathfrak{g}_{PL} \in_{\circ} Vset\ \alpha$
and *cat-PL2-*$\mathfrak{f}$[*cat-parallel-cs-intros*]: $\mathfrak{f}_{PL} \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

13.7 Local assumptions for a category with two parallel arrows between two objects

locale *cat-parallel-2* = $\mathcal{Z}\ \alpha$ **for** α +
fixes $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$
assumes *cat-parallel-2-*$\mathfrak{ab}$[*cat-parallel-cs-intros*]: $\mathfrak{a} \neq \mathfrak{b}$
and *cat-parallel-2-*$\mathfrak{ag}$[*cat-parallel-cs-intros*]: $\mathfrak{a} \neq \mathfrak{g}$
and *cat-parallel-2-*$\mathfrak{af}$[*cat-parallel-cs-intros*]: $\mathfrak{a} \neq \mathfrak{f}$
and *cat-parallel-2-*$\mathfrak{bg}$[*cat-parallel-cs-intros*]: $\mathfrak{b} \neq \mathfrak{g}$
and *cat-parallel-2-*$\mathfrak{bf}$[*cat-parallel-cs-intros*]: $\mathfrak{b} \neq \mathfrak{f}$
and *cat-parallel-2-*$\mathfrak{gf}$[*cat-parallel-cs-intros*]: $\mathfrak{g} \neq \mathfrak{f}$
and *cat-parallel-2-*$\mathfrak{a}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{a} \in_{\circ} Vset\ \alpha$
and *cat-parallel-2-*$\mathfrak{b}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{b} \in_{\circ} Vset\ \alpha$
and *cat-parallel-2-*$\mathfrak{g}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{g} \in_{\circ} Vset\ \alpha$
and *cat-parallel-2-*$\mathfrak{f}$*-in-Vset*[*cat-parallel-cs-intros*]: $\mathfrak{f} \in_{\circ} Vset\ \alpha$

lemmas (**in** *cat-parallel-2*) *cat-parallel-ineq* =
cat-parallel-2-$\mathfrak{ab}$
cat-parallel-2-$\mathfrak{ag}$
cat-parallel-2-$\mathfrak{af}$
cat-parallel-2-$\mathfrak{bg}$
cat-parallel-2-$\mathfrak{bf}$
cat-parallel-2-$\mathfrak{gf}$

Rules.

lemmas (**in** *cat-parallel-2*) [*cat-parallel-cs-intros*] = *cat-parallel-2-axioms*

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

sublocale *cat-parallel-2* ⊆ *cat-parallel* α $\mathfrak{a}$ $\mathfrak{b}$ ‹*set* {$\mathfrak{g}$, $\mathfrak{f}$}›
⟨*proof*⟩

Duality.

lemma (**in** *cat-parallel-2*) *cat-parallel-op*[*cat-op-intros*]:
cat-parallel-2 α $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$
⟨*proof*⟩

13.8 ↑↑: category with two parallel arrows between two objects

13.8.1 Definition and elementary properties

See Chapter I-2 and Chapter III-3 in [7].

definition *the-cat-parallel-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹↑↑$_C$›)
where ↑↑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ = ⇑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ (*set* {$\mathfrak{g}$, $\mathfrak{f}$})

Components.

lemma *the-cat-parallel-2-components*:
shows ↑↑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$(|*Obj*|) = *set* {$\mathfrak{a}$, $\mathfrak{b}$}
and ↑↑$_C$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$(|*Arr*|) = *set* {$\mathfrak{a}$, $\mathfrak{b}$, $\mathfrak{g}$, $\mathfrak{f}$}

⟨*proof*⟩

Elementary properties.

lemma *the-cat-parallel-2-commute*: $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f} = \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{f}\ \mathfrak{g}$
⟨*proof*⟩

lemma *cat-parallel-is-cat-parallel-2*:
assumes *cat-parallel* $\alpha\ \mathfrak{a}\ \mathfrak{b}$ (*set* $\{\mathfrak{g}, \mathfrak{f}\}$) **and** $\mathfrak{g} \neq \mathfrak{f}$
shows *cat-parallel-2* $\alpha\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$
⟨*proof*⟩

13.8.2 Objects

lemma *the-cat-parallel-2-Obj-*$\mathfrak{a}$*I*[*cat-parallel-cs-intros*]:
assumes $a = \mathfrak{a}$
shows $a \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-Obj-*$\mathfrak{b}$*I*[*cat-parallel-cs-intros*]:
assumes $a = \mathfrak{b}$
shows $a \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Obj|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-ObjE*:
assumes $a \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Obj|\!)$
obtains $a = \mathfrak{a} \mid a = \mathfrak{b}$
⟨*proof*⟩

13.8.3 Arrows

lemma *the-cat-parallel-2-Arr-*$\mathfrak{a}$*I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{a}$
shows $f \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-Arr-*$\mathfrak{b}$*I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{b}$
shows $f \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-Arr-*$\mathfrak{g}$*I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{g}$
shows $f \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-Arr-*$\mathfrak{f}$*I*[*cat-parallel-cs-intros*]:
assumes $f = \mathfrak{f}$
shows $f \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Arr|\!)$
⟨*proof*⟩

lemma *the-cat-parallel-2-ArrE*:
assumes $f \in_\circ \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Arr|\!)$
obtains $f = \mathfrak{a} \mid f = \mathfrak{b} \mid f = \mathfrak{g} \mid f = \mathfrak{f}$
⟨*proof*⟩

13.8.4 Domain

lemma *the-cat-parallel-2-Dom-vsv*[*cat-parallel-cs-intros*]: *vsv* $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!))$

⟨*proof*⟩

lemma *the-cat-parallel-2-Dom-vdomain*[*cat-parallel-cs-simps*]:
$\mathcal{D}_\circ\ (\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!)) = set\ \{\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}\}$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Dom-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Dom-app-*$\mathfrak{b}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Dom-app-*$\mathfrak{g}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{g}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Dom-app-*$\mathfrak{g}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Dom-app-*$\mathfrak{f}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{f}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Dom-app-*$\mathfrak{f}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Dom-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Dom|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Dom-app-*$\mathfrak{a}$

13.8.5 Codomain

lemma *the-cat-parallel-2-Cod-vsv*[*cat-parallel-cs-intros*]: *vsv* $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!))$
⟨*proof*⟩

lemma *the-cat-parallel-2-Cod-vdomain*[*cat-parallel-cs-simps*]:
$\mathcal{D}_\circ\ (\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!)) = set\ \{\mathfrak{a}, \mathfrak{b}, \mathfrak{g}, \mathfrak{f}\}$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Cod-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Cod-app-*$\mathfrak{b}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Cod-app-*$\mathfrak{g}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{g}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Cod-app-*$\mathfrak{g}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Cod-app-*$\mathfrak{f}$[*cat-parallel-cs-simps*]:

assumes $f = \mathfrak{f}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!)(\!|f|\!) = \mathfrak{b}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Cod-app-*$\mathfrak{f}$

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Cod-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Cod|\!)(\!|f|\!) = \mathfrak{a}$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cat-parallel-2.the-cat-parallel-2-Cod-app-*$\mathfrak{a}$

13.8.6 Composition

lemma *the-cat-parallel-2-Comp-vsv*[*cat-parallel-cs-intros*]:
vsv $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|Comp|\!))$
⟨*proof*⟩

lemma *the-cat-parallel-2-Comp-app-*$\mathfrak{b}\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{b}$ **and** $f = \mathfrak{b}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = g$ $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = f$
⟨*proof*⟩

lemma *the-cat-parallel-2-Comp-app-*$\mathfrak{a}\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{a}$ **and** $f = \mathfrak{a}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = g$ $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = f$
⟨*proof*⟩

lemma *the-cat-parallel-2-Comp-app-*$\mathfrak{b}\mathfrak{g}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{b}$ **and** $f = \mathfrak{g}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = f$
⟨*proof*⟩

lemma *the-cat-parallel-2-Comp-app-*$\mathfrak{b}\mathfrak{f}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{b}$ **and** $f = \mathfrak{f}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = f$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Comp-app-*$\mathfrak{g}\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{g}$ **and** $f = \mathfrak{a}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = g$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-Comp-app-*$\mathfrak{f}\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $g = \mathfrak{f}$ **and** $f = \mathfrak{a}$
shows $g \circ_{A\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} f = g$
⟨*proof*⟩

13.8.7 Identity

lemma *the-cat-parallel-2-CId-vsv*[*cat-parallel-cs-intros*]: *vsv* $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|CId|\!))$
⟨*proof*⟩

lemma *the-cat-parallel-2-CId-vdomain*[*cat-parallel-cs-simps*]:
$\mathcal{D}_\circ\ (\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|CId|\!)) = set\ \{\mathfrak{a}, \mathfrak{b}\}$
⟨*proof*⟩

lemma *the-cat-parallel-2-CId-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $a = \mathfrak{a}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|CId|\!)(\!|a|\!) = \mathfrak{a}$
⟨*proof*⟩

lemma *the-cat-parallel-2-CId-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $a = \mathfrak{b}$
shows $\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}(\!|CId|\!)(\!|a|\!) = \mathfrak{b}$
⟨*proof*⟩

13.8.8 Arrow with a domain and a codomain

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-is-arr-*$\mathfrak{aaa}$[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{a}$ **and** $b' = \mathfrak{a}$ **and** $f = \mathfrak{a}$
shows $f : a' \mapsto_{\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-is-arr-*$\mathfrak{bbb}$[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{b}$ **and** $b' = \mathfrak{b}$ **and** $f = \mathfrak{b}$
shows $f : a' \mapsto_{\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-is-arr-*$\mathfrak{abg}$[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f = \mathfrak{g}$
shows $f : a' \mapsto_{\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-is-arr-*$\mathfrak{abf}$[*cat-parallel-cs-intros*]:
assumes $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f = \mathfrak{f}$
shows $f : a' \mapsto_{\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} b'$
⟨*proof*⟩

lemma (**in** *cat-parallel-2*) *the-cat-parallel-2-is-arrE*:
assumes $f' : a' \mapsto_{\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}} b'$
obtains $a' = \mathfrak{a}$ **and** $b' = \mathfrak{a}$ **and** $f' = \mathfrak{a}$
| $a' = \mathfrak{b}$ **and** $b' = \mathfrak{b}$ **and** $f' = \mathfrak{b}$
| $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f' = \mathfrak{g}$
| $a' = \mathfrak{a}$ **and** $b' = \mathfrak{b}$ **and** $f' = \mathfrak{f}$
⟨*proof*⟩

13.8.9 $\uparrow\uparrow$ is a category

lemma (**in** *cat-parallel-2*)
finite-category-the-cat-parallel-2[*cat-parallel-cs-intros*]:
finite-category α $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f})$
⟨*proof*⟩

lemmas [*cat-parallel-cs-intros*] =
cat-parallel-2.finite-category-the-cat-parallel-2

13.8.10 Opposite parallel category

lemma (**in** *cat-parallel-2*) *op-cat-the-cat-parallel-2*[*cat-op-simps*]:
op-cat $(\uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}) = \uparrow\uparrow_C\ \mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *cat-parallel-2.op-cat-the-cat-parallel-2*

13.9 Parallel functor for a category with two parallel arrows between two objects

locale *cf-parallel-2* = *cat-parallel-2* α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ + *category* α $\mathfrak{C}$
  for α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$ $\mathfrak{C}$ :: V +
  assumes *cf-parallel-*$\mathfrak{g}'$[*cat-parallel-cs-intros*]: $\mathfrak{g}' : \mathfrak{a}' \mapsto_{\mathfrak{C}} \mathfrak{b}'$
    and *cf-parallel-*$\mathfrak{f}'$[*cat-parallel-cs-intros*]: $\mathfrak{f}' : \mathfrak{a}' \mapsto_{\mathfrak{C}} \mathfrak{b}'$

sublocale *cf-parallel-2* $\subseteq$
  cf-parallel α $\mathfrak{a}$ $\mathfrak{b}$ ‹*set* {$\mathfrak{g}$, $\mathfrak{f}$}› $\mathfrak{a}'$ $\mathfrak{b}'$ ‹$\lambda f \in_{\circ}$ *set* {$\mathfrak{g}$, $\mathfrak{f}$}. (f = $\mathfrak{f}$ *?* $\mathfrak{f}'$: $\mathfrak{g}'$)› $\mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{g}''$[*cat-parallel-cs-intros*]:
  assumes a = $\mathfrak{a}'$ **and** b = $\mathfrak{b}'$
  shows $\mathfrak{g}' : a \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{g}'''$[*cat-parallel-cs-intros*]:
  assumes g = $\mathfrak{g}'$ **and** b = $\mathfrak{b}'$
  shows $g : \mathfrak{a}' \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{g}''''$[*cat-parallel-cs-intros*]:
  assumes g = $\mathfrak{g}'$ **and** a = $\mathfrak{a}'$
  shows $g : a \mapsto_{\mathfrak{C}} \mathfrak{b}'$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{f}''$[*cat-parallel-cs-intros*]:
  assumes a = $\mathfrak{a}'$ **and** b = $\mathfrak{b}'$
  shows $\mathfrak{f}' : a \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{f}'''$[*cat-parallel-cs-intros*]:
  assumes f = $\mathfrak{f}'$ **and** b = $\mathfrak{b}'$
  shows $f : \mathfrak{a}' \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *cf-parallel-2-*$\mathfrak{f}''''$[*cat-parallel-cs-intros*]:
  assumes f = $\mathfrak{f}'$ **and** a = $\mathfrak{a}'$
  shows $f : a \mapsto_{\mathfrak{C}} \mathfrak{b}'$
  ⟨*proof*⟩

Rules.

lemma (**in** *cf-parallel-2*) *cf-parallel-axioms'*[*cat-parallel-cs-intros*]:
  assumes α' = α
    and a = $\mathfrak{a}$
    and b = $\mathfrak{b}$
    and g = $\mathfrak{g}$
    and f = $\mathfrak{f}$
    and a' = $\mathfrak{a}'$
    and b' = $\mathfrak{b}'$
    and g' = $\mathfrak{g}'$
    and f' = $\mathfrak{f}'$
  shows *cf-parallel-2* α' a b g f a' b' g' f' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *cf-parallel-2-def*[*unfolded cf-parallel-2-axioms-def*]
  |*intro cf-parallel-2I*|

|*dest cf-parallel-2D*[*dest*]|
|*elim cf-parallel-2E*[*elim*]|

lemmas [*cat-parallel-cs-intros*] = *cf-parallelD*(*1*,*2*)

Duality.

lemma (in *cf-parallel-2*) *cf-parallel-2-op*[*cat-op-intros*]:
cf-parallel-2 α $\mathfrak{b}$ $\mathfrak{a}$ $\mathfrak{f}$ $\mathfrak{g}$ $\mathfrak{b}'$ $\mathfrak{a}'$ $\mathfrak{f}'$ $\mathfrak{g}'$ (*op-cat* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*cat-op-intros*] = *cf-parallel.cf-parallel-op*

13.9.1 Definition and elementary properties

definition *the-cf-parallel-2* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
(‹↑↑→↑↑$_{CF}$›)
where ↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$ =
⇑→⇑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ (*set* {$\mathfrak{g}$, $\mathfrak{f}$}) $\mathfrak{a}'$ $\mathfrak{b}'$ ($\lambda f \in_{\circ} set$ {$\mathfrak{g}$, $\mathfrak{f}$}. (f = $\mathfrak{f}$? $\mathfrak{f}'$: $\mathfrak{g}'$))

Components.

lemma *the-cf-parallel-2-components*:
shows [*cat-parallel-cs-simps*]:
↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$(|*HomDom*|) = ↑↑$_{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$
and [*cat-parallel-cs-simps*]:
↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$(|*HomCod*|) = $\mathfrak{C}$
⟨*proof*⟩

Elementary properties.

lemma (in *cf-parallel-2*) *cf-parallel-2-the-cf-parallel-2-commute*:
↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$ = ↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{f}$ $\mathfrak{g}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{f}'$ $\mathfrak{g}'$
⟨*proof*⟩

lemma *cf-parallel-is-cf-parallel-2*:
assumes
cf-parallel α $\mathfrak{a}$ $\mathfrak{b}$ (*set* {$\mathfrak{g}$, $\mathfrak{f}$}) $\mathfrak{a}'$ $\mathfrak{b}'$ ($\lambda f \in_{\circ} set$ {$\mathfrak{g}$, $\mathfrak{f}$}. (f = $\mathfrak{f}$? $\mathfrak{f}'$: $\mathfrak{g}'$)) $\mathfrak{C}$
and $\mathfrak{g} \neq \mathfrak{f}$
shows *cf-parallel-2* α $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$ $\mathfrak{C}$
⟨*proof*⟩

13.9.2 Object map

lemma *the-cf-parallel-2-ObjMap-vsv*[*cat-parallel-cs-intros*]:
vsv (↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$(|*ObjMap*|))
⟨*proof*⟩

lemma *the-cf-parallel-2-ObjMap-vdomain*[*cat-parallel-cs-simps*]:
$\mathcal{D}_{\circ}$ (↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$(|*ObjMap*|)) = ↑↑$_{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$(|*Obj*|)
⟨*proof*⟩

lemma (in *cf-parallel-2*) *the-cf-parallel-2-ObjMap-app-*$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes x = $\mathfrak{a}$
shows ↑↑→↑↑$_{CF}$ $\mathfrak{C}$ $\mathfrak{a}$ $\mathfrak{b}$ $\mathfrak{g}$ $\mathfrak{f}$ $\mathfrak{a}'$ $\mathfrak{b}'$ $\mathfrak{g}'$ $\mathfrak{f}'$(|*ObjMap*|)(|x|) = $\mathfrak{a}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ObjMap-app-*$\mathfrak{a}$

lemma (in *cf-parallel-2*) *the-cf-parallel-2-ObjMap-app-*$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes x = $\mathfrak{b}$

shows $\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ObjMap|\!)(\!|x|\!) = \mathfrak{b}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ObjMap-app*-$\mathfrak{b}$

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ObjMap-vrange*:
$\mathcal{R}_{\circ}\ (\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ObjMap|\!)) = set\ \{\mathfrak{a}', \mathfrak{b}'\}$
⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ObjMap-vrange-vsubset-Obj*:
$\mathcal{R}_{\circ}\ (\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ObjMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

13.9.3 Arrow map

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-app*-$\mathfrak{g}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{g}$
shows $\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{g}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ArrMap-app*-$\mathfrak{g}$

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-app*-$\mathfrak{f}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{f}$
shows $\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{f}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ArrMap-app*-$\mathfrak{f}$

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-app*-$\mathfrak{a}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{a}$
shows $\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}'|\!)$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ArrMap-app*-$\mathfrak{a}$

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-app*-$\mathfrak{b}$[*cat-parallel-cs-simps*]:
assumes $f = \mathfrak{b}$
shows $\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}'|\!)$
⟨*proof*⟩

lemmas [*cat-parallel-cs-simps*] = *cf-parallel-2.the-cf-parallel-2-ArrMap-app*-$\mathfrak{b}$

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-vrange*:
$\mathcal{R}_{\circ}\ (\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)) = set\ \{\mathfrak{C}(\!|CId|\!)(\!|\mathfrak{a}'|\!), \mathfrak{C}(\!|CId|\!)(\!|\mathfrak{b}'|\!), \mathfrak{f}', \mathfrak{g}'\}$
⟨*proof*⟩

lemma (**in** *cf-parallel-2*) *the-cf-parallel-2-ArrMap-vrange-vsubset-Arr*:
$\mathcal{R}_{\circ}\ (\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}'(\!|ArrMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

13.9.4 Parallel functor for a category with two parallel arrows between two objects is a functor

lemma (**in** *cf-parallel-2*) *cf-parallel-2-the-cf-parallel-2-is-tm-functor*:
$\uparrow\uparrow\!\!\rightarrow\!\!\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}' : \uparrow\uparrow_{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f} \mapsto\!\!\mapsto_{C.tm\alpha}\ \mathfrak{C}$
⟨*proof*⟩

lemma (in *cf-parallel-2*) *cf-parallel-2-the-cf-parallel-2-is-tm-functor′*:
assumes $\mathfrak{A}' = \uparrow\uparrow_C\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\uparrow\uparrow{\rightarrow}\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}' : \mathfrak{A}' \mapsto\mapsto_{C.tm\alpha}\ \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-parallel-cs-intros*] =
cf-parallel-2.cf-parallel-2-the-cf-parallel-2-is-tm-functor′

13.9.5 Opposite parallel functor for a category with two parallel arrows between two objects

lemma (in *cf-parallel-2*) *cf-parallel-2-the-cf-parallel-2-op*[*cat-op-simps*]:
op-cf $(\uparrow\uparrow{\rightarrow}\uparrow\uparrow_{CF}\ \mathfrak{C}\ \mathfrak{a}\ \mathfrak{b}\ \mathfrak{g}\ \mathfrak{f}\ \mathfrak{a}'\ \mathfrak{b}'\ \mathfrak{g}'\ \mathfrak{f}') =$
$\uparrow\uparrow{\rightarrow}\uparrow\uparrow_{CF}$ (*op-cat* $\mathfrak{C}$) $\mathfrak{b}\ \mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}\ \mathfrak{b}'\ \mathfrak{a}'\ \mathfrak{f}'\ \mathfrak{g}'$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *cf-parallel-2.cf-parallel-2-the-cf-parallel-2-op*

14 Comma categories

14.1 Background

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

14.2 Comma category

14.2.1 Definition and elementary properties

See Exercise 1.3.vi in [12] or Chapter II-6 in [7].

definition *cat-comma-Obj* :: $V \Rightarrow V \Rightarrow V$
  where *cat-comma-Obj* $\mathfrak{G}\ \mathfrak{H} \equiv$ *set*
    {
      $[a,\ b,\ f]_\circ \mid a\ b\ f.$
        $a \in_\circ \mathfrak{G}(\lfloor HomDom \rfloor)(\lfloor Obj \rfloor) \wedge$
        $b \in_\circ \mathfrak{H}(\lfloor HomDom \rfloor)(\lfloor Obj \rfloor) \wedge$
        $f : \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{G}(\lfloor HomCod \rfloor)} \mathfrak{H}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
    }

lemma *small-cat-comma-Obj*[*simp*]:
  small
    {
      $[a,\ b,\ f]_\circ \mid a\ b\ f.$
        $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor) \wedge b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor) \wedge f : \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{C}} \mathfrak{H}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$
    }
  (**is** ‹*small ?abfs*›)
⟨*proof*⟩

definition *cat-comma-Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cat-comma-Hom* $\mathfrak{G}\ \mathfrak{H}\ A\ B \equiv$ *set*
    {
      $[A,\ B,\ [g,\ h]_\circ]_\circ \mid g\ h.$
        $A \in_\circ$ *cat-comma-Obj* $\mathfrak{G}\ \mathfrak{H} \wedge$
        $B \in_\circ$ *cat-comma-Obj* $\mathfrak{G}\ \mathfrak{H} \wedge$
        $g : A(\lfloor 0 \rfloor) \mapsto_{\mathfrak{G}(\lfloor HomDom \rfloor)} B(\lfloor 0 \rfloor) \wedge$
        $h : A(\lfloor 1_{\mathbb{N}} \rfloor) \mapsto_{\mathfrak{H}(\lfloor HomDom \rfloor)} B(\lfloor 1_{\mathbb{N}} \rfloor) \wedge$
        $B(\lfloor 2_{\mathbb{N}} \rfloor) \circ_{A\mathfrak{G}(\lfloor HomCod \rfloor)} \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) =$
        $\mathfrak{H}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor) \circ_{A\mathfrak{G}(\lfloor HomCod \rfloor)} A(\lfloor 2_{\mathbb{N}} \rfloor)$
    }

lemma *small-cat-comma-Hom*[*simp*]: *small*
  {
    $[A,\ B,\ [g,\ h]_\circ]_\circ \mid g\ h.$
      $A \in_\circ$ *cat-comma-Obj* $\mathfrak{G}\ \mathfrak{H} \wedge$
      $B \in_\circ$ *cat-comma-Obj* $\mathfrak{G}\ \mathfrak{H} \wedge$
      $g : A(\lfloor 0 \rfloor) \mapsto_{\mathfrak{A}} B(\lfloor 0 \rfloor) \wedge$
      $h : A(\lfloor 1_{\mathbb{N}} \rfloor) \mapsto_{\mathfrak{B}} B(\lfloor 1_{\mathbb{N}} \rfloor) \wedge$
      $B(\lfloor 2_{\mathbb{N}} \rfloor) \circ_{A\mathfrak{C}} \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) = \mathfrak{H}(\lfloor ArrMap \rfloor)(\lfloor h \rfloor) \circ_{A\mathfrak{C}} A(\lfloor 2_{\mathbb{N}} \rfloor)$
  }
  (**is** ‹*small ?abf-a'b'f'-gh*›)
⟨*proof*⟩

definition *cat-comma-Arr* :: $V \Rightarrow V \Rightarrow V$
  where *cat-comma-Arr* $\mathfrak{G}\ \mathfrak{H} \equiv$
    (

$\bigcup_\circ A \in_\circ$ *cat-comma-Obj* $\mathfrak{G}$ $\mathfrak{H}$. $\bigcup_\circ B \in_\circ$ *cat-comma-Obj* $\mathfrak{G}$ $\mathfrak{H}$.
cat-comma-Hom $\mathfrak{G}$ $\mathfrak{H}$ A B
)

definition *cat-comma-composable* :: $V \Rightarrow V \Rightarrow V$
where *cat-comma-composable* $\mathfrak{G}$ $\mathfrak{H}$ $\equiv$ *set*
{
$[[B,\ C,\ G]_\circ,\ [A,\ B,\ F]_\circ]_\circ \mid A\ B\ C\ G\ F.$
$[B,\ C,\ G]_\circ \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$ $\wedge$ $[A,\ B,\ F]_\circ \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$
}

lemma *small-cat-comma-composable*[*simp*]:
shows *small*
{
$[[B,\ C,\ G]_\circ,\ [A,\ B,\ F]_\circ]_\circ \mid A\ B\ C\ G\ F.$
$[B,\ C,\ G]_\circ \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$ $\wedge$ $[A,\ B,\ F]_\circ \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$
}
(**is** ‹*small ?S*›)
⟨*proof*⟩

definition *cat-comma* :: $V \Rightarrow V \Rightarrow V$ (‹(- $_{CF}\!\downarrow_{CF}$ -)› [*1000*, *1000*] *999*)
where $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}$ =
[
cat-comma-Obj $\mathfrak{G}$ $\mathfrak{H}$,
cat-comma-Arr $\mathfrak{G}$ $\mathfrak{H}$,
($\lambda F \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$. $F(\!|0|\!)$),
($\lambda F \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$. $F(\!|1_{\mathbb{N}}|\!)$),
(
$\lambda GF \in_\circ$ *cat-comma-composable* $\mathfrak{G}$ $\mathfrak{H}$.
[
$GF(\!|1_{\mathbb{N}}|\!)(\!|0|\!)$,
$GF(\!|0|\!)(\!|1_{\mathbb{N}}|\!)$,
[
$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!) \circ_{A\,\mathfrak{G}(\!|HomDom|\!)} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!)$,
$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!) \circ_{A\,\mathfrak{H}(\!|HomDom|\!)} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!)$
]$_\circ$
]$_\circ$
),
(
$\lambda A \in_\circ$ *cat-comma-Obj* $\mathfrak{G}$ $\mathfrak{H}$.
$[A,\ A,\ [\mathfrak{G}(\!|HomDom|\!)(\!|CId|\!)(\!|A(\!|0|\!)|\!),\ \mathfrak{H}(\!|HomDom|\!)(\!|CId|\!)(\!|A(\!|1_{\mathbb{N}}|\!)|\!)]_\circ]_\circ$
)
]$_\circ$

Components.

lemma *cat-comma-components*:
shows $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}(\!|Obj|\!)$ = *cat-comma-Obj* $\mathfrak{G}$ $\mathfrak{H}$
and $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}(\!|Arr|\!)$ = *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$
and $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}(\!|Dom|\!)$ = ($\lambda F \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$. $F(\!|0|\!)$)
and $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}(\!|Cod|\!)$ = ($\lambda F \in_\circ$ *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$. $F(\!|1_{\mathbb{N}}|\!)$)
and $\mathfrak{G}$ $_{CF}\!\downarrow_{CF}$ $\mathfrak{H}(\!|Comp|\!)$ =
(
$\lambda GF \in_\circ$ *cat-comma-composable* $\mathfrak{G}$ $\mathfrak{H}$.
[
$GF(\!|1_{\mathbb{N}}|\!)(\!|0|\!)$,
$GF(\!|0|\!)(\!|1_{\mathbb{N}}|\!)$,
[
$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!) \circ_{A\,\mathfrak{G}(\!|HomDom|\!)} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!)$,

$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!) \circ_{A\mathfrak{H}(\!|HomDom|\!)} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!)$
]$_\circ$
]$_\circ$
)
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|CId|\!)$ =
(
$\lambda A \in_\circ cat\text{-}comma\text{-}Obj\ \mathfrak{G}\ \mathfrak{H}.$
$[A,\ A,\ [\mathfrak{G}(\!|HomDom|\!)(\!|CId|\!)(\!|A(\!|0|\!)|\!),\ \mathfrak{H}(\!|HomDom|\!)(\!|CId|\!)(\!|A(\!|1_{\mathbb{N}}|\!)|\!)]_\circ]_\circ$
)
⟨*proof*⟩

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

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

lemma *cat-comma-Obj-def′*:
cat-comma-Obj $\mathfrak{G}$ $\mathfrak{H}$ ≡ *set*
{
$[a,\ b,\ f]_\circ \mid a\ b\ f.$
$a \in_\circ \mathfrak{A}(\!|Obj|\!) \wedge b \in_\circ \mathfrak{B}(\!|Obj|\!) \wedge f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
}
⟨*proof*⟩

lemma *cat-comma-Hom-def′*:
cat-comma-Hom $\mathfrak{G}$ $\mathfrak{H}$ A B ≡ *set*
{
$[A,\ B,\ [g,\ h]_\circ]_\circ \mid g\ h.$
$A \in_\circ cat\text{-}comma\text{-}Obj\ \mathfrak{G}\ \mathfrak{H} \wedge$
$B \in_\circ cat\text{-}comma\text{-}Obj\ \mathfrak{G}\ \mathfrak{H} \wedge$
$g : A(\!|0|\!) \mapsto_{\mathfrak{A}} B(\!|0|\!) \wedge$
$h : A(\!|1_{\mathbb{N}}|\!) \mapsto_{\mathfrak{B}} B(\!|1_{\mathbb{N}}|\!) \wedge$
$B(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} A(\!|2_{\mathbb{N}}|\!)$
}
⟨*proof*⟩

lemma *cat-comma-components′*:
shows $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|Obj|\!)$ = *cat-comma-Obj* $\mathfrak{G}$ $\mathfrak{H}$
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|Arr|\!)$ = *cat-comma-Arr* $\mathfrak{G}$ $\mathfrak{H}$
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|Dom|\!)$ = $(\lambda F \in_\circ cat\text{-}comma\text{-}Arr\ \mathfrak{G}\ \mathfrak{H}.\ F(\!|0|\!))$
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|Cod|\!)$ = $(\lambda F \in_\circ cat\text{-}comma\text{-}Arr\ \mathfrak{G}\ \mathfrak{H}.\ F(\!|1_{\mathbb{N}}|\!))$
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|Comp|\!)$ =
(
$\lambda GF \in_\circ cat\text{-}comma\text{-}composable\ \mathfrak{G}\ \mathfrak{H}.$
[
$GF(\!|1_{\mathbb{N}}|\!)(\!|0|\!),$
$GF(\!|0|\!)(\!|1_{\mathbb{N}}|\!),$
[
$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!) \circ_{A\mathfrak{A}} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|0|\!),$
$GF(\!|0|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!) \circ_{A\mathfrak{B}} GF(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!)$
]$_\circ$
]$_\circ$
)
and $\mathfrak{G}$ $_{CF\downarrow CF}$ $\mathfrak{H}(\!|CId|\!)$ =

$(\lambda A \in_\circ \mathit{cat\text{-}comma\text{-}Obj}\ \mathfrak{G}\ \mathfrak{H}.\ [A,\ A,\ [\mathfrak{A}(\!|\mathit{CId}|\!)(\!|A(\!|0|\!)|\!),\ \mathfrak{B}(\!|\mathit{CId}|\!)(\!|A(\!|1_{\mathbb{N}}|\!)|\!)]_\circ]_\circ)$
⟨*proof*⟩

end

14.2.2 Objects

lemma *cat-comma-ObjI*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $A = [a,\ b,\ f]_\circ$
and $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$
and $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
and $f : \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|\mathit{ObjMap}|\!)(\!|b|\!)$
shows $A \in_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemma *cat-comma-ObjD*[*dest*]:
assumes $[a,\ b,\ f]_\circ \in_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|\mathit{Obj}|\!)$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$
and $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
and $f : \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|\mathit{ObjMap}|\!)(\!|b|\!)$
⟨*proof*⟩

lemma *cat-comma-ObjE*[*elim*]:
assumes $A \in_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|\mathit{Obj}|\!)$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $a\ b\ f$ **where** $A = [a,\ b,\ f]_\circ$
and $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$
and $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
and $f : \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|\mathit{ObjMap}|\!)(\!|b|\!)$
⟨*proof*⟩

14.2.3 Arrows

lemma *cat-comma-HomI*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $F = [A,\ B,\ [g,\ h]_\circ]_\circ$
and $A = [a,\ b,\ f]_\circ$
and $B = [a',\ b',\ f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|\mathit{ObjMap}|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|\mathit{ObjMap}|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|\mathit{ArrMap}|\!)(\!|g|\!) = \mathfrak{H}(\!|\mathit{ArrMap}|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
shows $F \in_\circ \mathit{cat\text{-}comma\text{-}Hom}\ \mathfrak{G}\ \mathfrak{H}\ A\ B$
⟨*proof*⟩

lemma *cat-comma-HomE*[*elim*]:
assumes $F \in_\circ \mathit{cat\text{-}comma\text{-}Hom}\ \mathfrak{G}\ \mathfrak{H}\ A\ B$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $a\ b\ f\ a'\ b'\ f'\ g\ h$
where $F = [A,\ B,\ [g,\ h]_\circ]_\circ$

and $A = [a,\ b,\ f]_\circ$
and $B = [a',\ b',\ f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-comma-HomD*[*dest*]:
assumes $[[a,\ b,\ f]_\circ,\ [a',\ b',\ f']_\circ,\ [g,\ h]_\circ]_\circ \in_\circ$ *cat-comma-Hom* $\mathfrak{G}\ \mathfrak{H}\ A\ B$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-comma-ArrI*[*cat-comma-cs-intros*]:
assumes $F \in_\circ$ *cat-comma-Hom* $\mathfrak{G}\ \mathfrak{H}\ A\ B$
and $A \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
and $B \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
shows $F \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-comma-ArrE*[*elim*]:
assumes $F \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Arr|\!)$
obtains $A\ B$
where $F \in_\circ$ *cat-comma-Hom* $\mathfrak{G}\ \mathfrak{H}\ A\ B$
and $A \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
and $B \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-comma-ArrD*[*dest*]:
assumes $[A,\ B,\ F]_\circ \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Arr|\!)$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $[A,\ B,\ F]_\circ \in_\circ$ *cat-comma-Hom* $\mathfrak{G}\ \mathfrak{H}\ A\ B$
and $A \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
and $B \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Obj|\!)$
⟨*proof*⟩

14.2.4 Domain

lemma *cat-comma-Dom-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Dom|\!))$
⟨*proof*⟩

lemma *cat-comma-Dom-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ\ (\mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Dom|\!)) = \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-comma-Dom-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A,\ B,\ F]_\circ$ **and** $ABF \in_\circ \mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Arr|\!)$
shows $\mathfrak{G}\ _{CF\downarrow_{CF}}\ \mathfrak{H}(\!|Dom|\!)(\!|ABF|\!) = A$
⟨*proof*⟩

lemma *cat-comma-Dom-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Dom|\!)) \subseteq_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!)$
⟨*proof*⟩

14.2.5 Codomain

lemma *cat-comma-Cod-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Cod|\!))$
⟨*proof*⟩

lemma *cat-comma-Cod-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ\ (\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Cod|\!)) = \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-comma-Cod-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A,\ B,\ F]_\circ$ **and** $ABF \in_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!)$
shows $\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Cod|\!)(\!|ABF|\!) = B$
⟨*proof*⟩

lemma *cat-comma-Cod-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Cod|\!)) \subseteq_\circ \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!)$
⟨*proof*⟩

14.2.6 Arrow with a domain and a codomain

lemma *cat-comma-is-arrI*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $ABF = [A,\ B,\ F]_\circ$
and $A = [a,\ b,\ f]_\circ$
and $B = [a',\ b',\ f']_\circ$
and $F = [g,\ h]_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
shows $ABF : A \mapsto_{\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}} B$
⟨*proof*⟩

lemma *cat-comma-is-arrD*[*dest*]:
assumes $[[a,\ b,\ f]_\circ,\ [a',\ b',\ f']_\circ,\ [g,\ h]_\circ]_\circ :$
$[a,\ b,\ f]_\circ \mapsto_{\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}} [a',\ b',\ f']_\circ$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma *cat-comma-is-arrE*[*elim*]:
assumes $ABF : A \mapsto_{\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}} B$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains $a\ b\ f\ a'\ b'\ f'\ g\ h$

where $ABF = [[a,\ b,\ f]_\circ,\ [a',\ b',\ f']_\circ,\ [g,\ h]_\circ]_\circ$
and $A = [a,\ b,\ f]_\circ$
and $B = [a',\ b',\ f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $h : b \mapsto_{\mathfrak{B}} b'$
and $f : \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b|\!)$
and $f' : \mathfrak{G}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{C}} \mathfrak{H}(\!|ObjMap|\!)(\!|b'|\!)$
and $f' \circ_{A\mathfrak{C}} \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{H}(\!|ArrMap|\!)(\!|h|\!) \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

14.2.7 Composition

lemma *cat-comma-composableI*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $ABCGF = [BCG,\ ABF]_\circ$
and $BCG : B \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} C$
and $ABF : A \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} B$
shows $ABCGF \in_\circ$ *cat-comma-composable* $\mathfrak{G}\ \mathfrak{H}$
⟨*proof*⟩

lemma *cat-comma-composableE*[*elim*]:
assumes $ABCGF \in_\circ$ *cat-comma-composable* $\mathfrak{G}\ \mathfrak{H}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
obtains *BCG ABF A B C*
where $ABCGF = [BCG,\ ABF]_\circ$
and $BCG : B \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} C$
and $ABF : A \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} B$
⟨*proof*⟩

lemma *cat-comma-Comp-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{G}\ _{CF\downarrow CF}\ \mathfrak{H}(\!|Comp|\!))$
⟨*proof*⟩

lemma *cat-comma-Comp-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ\ (\mathfrak{G}\ _{CF\downarrow CF}\ \mathfrak{H}(\!|Comp|\!)) =$ *cat-comma-composable* $\mathfrak{G}\ \mathfrak{H}$
⟨*proof*⟩

lemma *cat-comma-Comp-app*[*cat-comma-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $G = [B,\ C,\ [g',\ h']_\circ]_\circ$
and $F = [A,\ B,\ [g,\ h]_\circ]_\circ$
and $G : B \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} C$
and $F : A \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} B$
shows $G \circ_{A\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} F = [A,\ C,\ [g' \circ_{A\mathfrak{A}} g,\ h' \circ_{A\mathfrak{B}} h]_\circ]_\circ$
⟨*proof*⟩

lemma *cat-comma-Comp-is-arr*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $BCG : B \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} C$
and $ABF : A \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} B$
shows $BCG \circ_{A\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} ABF : A \mapsto_{\mathfrak{G}\ CF\downarrow CF\ \mathfrak{H}} C$
⟨*proof*⟩

14.2.8 Identity

lemma *cat-comma-CId-vsv*[*cat-comma-cs-intros*]: *vsv* ($\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|CId|\!)$)
⟨*proof*⟩

lemma *cat-comma-CId-vdomain*[*cat-comma-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ ($\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|CId|\!)$) = $\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-comma-CId-app*[*cat-comma-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $A = [a,\ b\ ,f]_\circ$
and $A \in_\circ$ $\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|Obj|\!)$
shows $\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|CId|\!)(\!|A|\!) = [A,\ A,\ [\mathfrak{A}(\!|CId|\!)(\!|a|\!),\ \mathfrak{B}(\!|CId|\!)(\!|b|\!)]_\circ]_\circ$
⟨*proof*⟩

14.2.9 *Hom*-set

lemma *cat-comma-Hom*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $A \in_\circ$ $\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|Obj|\!)$
and $B \in_\circ$ $\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}(\!|Obj|\!)$
shows *Hom* ($\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}$) A B = *cat-comma-Hom* $\mathfrak{G}$ $\mathfrak{H}$ A B
⟨*proof*⟩

14.2.10 Comma category is a category

lemma *category-cat-comma*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
shows *category* α ($\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}$)
⟨*proof*⟩

14.2.11 Tiny comma category

lemma *tiny-category-cat-comma*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{C}$
shows *tiny-category* α ($\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H}$)
⟨*proof*⟩

14.3 Opposite comma category functor

14.3.1 Background

See [2][7] for background information.

14.3.2 Object flip

definition *op-cf-commma-obj-flip* :: $V \Rightarrow V \Rightarrow V$
where *op-cf-commma-obj-flip* $\mathfrak{G}$ $\mathfrak{H}$ =
$(\lambda A \in_\circ (\mathfrak{G}$ $_{CF\downarrow_{CF}}$ $\mathfrak{H})(\!|Obj|\!).\ [A(\!|1_{\mathbb{N}}|\!),\ A(\!|0|\!),\ A(\!|2_{\mathbb{N}}|\!)]_\circ)$

Elementary properties.

mk-VLambda *op-cf-commma-obj-flip-def*
|*vsv op-cf-commma-obj-flip-vsv*[*cat-comma-cs-intros*]|

[7] https://en.wikipedia.org/wiki/Opposite_category

|*vdomain op-cf-commma-obj-flip-vdomain*[*cat-comma-cs-simps*]|
|*app op-cf-commma-obj-flip-app'*|

lemma *op-cf-commma-obj-flip-app*[*cat-comma-cs-simps*]:
assumes $A = [a, b, f]_\circ$ **and** $A \in_\circ (\mathfrak{G}\ _{CF}{\downarrow_{CF}}\ \mathfrak{H})(\!|Obj|\!)$
shows *op-cf-commma-obj-flip* $\mathfrak{G}\ \mathfrak{H}(\!|A|\!) = [b, a, f]_\circ$
⟨*proof*⟩

lemma *op-cf-commma-obj-flip-v11*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *v11* (*op-cf-commma-obj-flip* $\mathfrak{G}\ \mathfrak{H}$)
⟨*proof*⟩

lemma *op-cf-commma-obj-flip-vrange*[*cat-comma-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ$ (*op-cf-commma-obj-flip* $\mathfrak{G}\ \mathfrak{H}$) = (*op-cf* $\mathfrak{H}$) $_{CF}{\downarrow_{CF}}$ (*op-cf* $\mathfrak{G}$)(|*Obj*|)
⟨*proof*⟩

14.3.3 Definition and elementary properties

definition *op-cf-comma* :: $V \Rightarrow V \Rightarrow V$
where *op-cf-comma* $\mathfrak{G}\ \mathfrak{H}$ =
[
op-cf-commma-obj-flip $\mathfrak{G}\ \mathfrak{H}$,
(
$\lambda ABF \in_\circ (\mathfrak{G}\ _{CF}{\downarrow_{CF}}\ \mathfrak{H})(\!|Arr|\!)$.
[
op-cf-commma-obj-flip $\mathfrak{G}\ \mathfrak{H}(\!|ABF(\!|1_{\mathbb{N}}|\!)|\!)$,
op-cf-commma-obj-flip $\mathfrak{G}\ \mathfrak{H}(\!|ABF(\!|0|\!)|\!)$,
$[ABF(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!),\ ABF(\!|2_{\mathbb{N}}|\!)(\!|0_{\mathbb{N}}|\!)]_\circ$
]$_\circ$
),
op-cat $(\mathfrak{G}\ _{CF}{\downarrow_{CF}}\ \mathfrak{H})$,
(*op-cf* $\mathfrak{H}$) $_{CF}{\downarrow_{CF}}$ (*op-cf* $\mathfrak{G}$)
]$_\circ$

Components.

lemma *op-cf-comma-components*:
shows [*cat-comma-cs-simps*]:
op-cf-comma $\mathfrak{G}\ \mathfrak{H}(\!|ObjMap|\!)$ = *op-cf-commma-obj-flip* $\mathfrak{G}\ \mathfrak{H}$
and *op-cf-comma* $\mathfrak{G}\ \mathfrak{H}(\!|ArrMap|\!)$ =
(
$\lambda ABF \in_\circ (\mathfrak{G}\ _{CF}{\downarrow_{CF}}\ \mathfrak{H})(\!|Arr|\!)$.
[
op-cf-commma-obj-flip $\mathfrak{G}\ \mathfrak{H}(\!|ABF(\!|1_{\mathbb{N}}|\!)|\!)$,
op-cf-commma-obj-flip $\mathfrak{G}\ \mathfrak{H}(\!|ABF(\!|0|\!)|\!)$,
$[ABF(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!),\ ABF(\!|2_{\mathbb{N}}|\!)(\!|0_{\mathbb{N}}|\!)]_\circ$
]$_\circ$
)
and [*cat-comma-cs-simps*]:
op-cf-comma $\mathfrak{G}\ \mathfrak{H}(\!|HomDom|\!)$ = *op-cat* $(\mathfrak{G}\ _{CF}{\downarrow_{CF}}\ \mathfrak{H})$
and [*cat-comma-cs-simps*]:
op-cf-comma $\mathfrak{G}\ \mathfrak{H}(\!|HomCod|\!)$ = (*op-cf* $\mathfrak{H}$) $_{CF}{\downarrow_{CF}}$ (*op-cf* $\mathfrak{G}$)
⟨*proof*⟩

14.3.4 Arrow map

mk-VLambda *op-cf-comma-components*(*2*)

|*vsv op-cf-comma-ArrMap-vsv*[*cat-comma-cs-intros*]|
|*vdomain op-cf-comma-ArrMap-vdomain*[*cat-comma-cs-simps*]|
|*app op-cf-comma-ArrMap-app'*|

lemma *op-cf-comma-ArrMap-app*[*cat-comma-cs-simps*]:
assumes $ABF = [[a, b, f]_\circ, [a', b', f']_\circ, [g, h]_\circ]_\circ$
and $ABF \in_\circ \mathfrak{G}_{CF\downarrow CF} \mathfrak{H}(\!|Arr|\!)$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}(\!|ArrMap|\!)(\!|ABF|\!) =$
$[$
op-cf-commma-obj-flip $\mathfrak{G}$ $\mathfrak{H}(\!|a', b', f'|\!)_\bullet$,
op-cf-commma-obj-flip $\mathfrak{G}$ $\mathfrak{H}(\!|a, b, f|\!)_\bullet$,
$[h, g]_\circ$
$]_\circ$
⟨*proof*⟩

lemma *op-cf-comma-ArrMap-v11*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *v11* (*op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *op-cf-comma-ArrMap-is-arr*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $ABF : A \mapsto_{\mathfrak{G}_{CF\downarrow CF} \mathfrak{H}} B$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}(\!|ArrMap|\!)(\!|ABF|\!)$:
op-cf-commma-obj-flip $\mathfrak{G}$ $\mathfrak{H}(\!|B|\!) \mapsto_{(op\text{-}cf\ \mathfrak{H})_{CF\downarrow CF} (op\text{-}cf\ \mathfrak{G})}$
op-cf-commma-obj-flip $\mathfrak{G}$ $\mathfrak{H}(\!|A|\!)$
⟨*proof*⟩

lemma *op-cf-comma-ArrMap-is-arr'*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $ABF : A \mapsto_{\mathfrak{G}_{CF\downarrow CF} \mathfrak{H}} B$
and $A' =$ *op-cf-commma-obj-flip* $\mathfrak{G}$ $\mathfrak{H}(\!|B|\!)$
and $B' =$ *op-cf-commma-obj-flip* $\mathfrak{G}$ $\mathfrak{H}(\!|A|\!)$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}(\!|ArrMap|\!)(\!|ABF|\!) : A' \mapsto_{(op\text{-}cf\ \mathfrak{H})_{CF\downarrow CF} (op\text{-}cf\ \mathfrak{G})} B'$
⟨*proof*⟩

lemma *op-cf-comma-ArrMap-vrange*[*cat-comma-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ$ (*op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}(\!|ArrMap|\!)$) = $(op\text{-}cf\ \mathfrak{H})_{CF\downarrow CF} (op\text{-}cf\ \mathfrak{G})(\!|Arr|\!)$
⟨*proof*⟩

14.3.5 Opposite comma category functor is an isomorphism of categories

lemma *op-cf-comma-is-iso-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}$:
op-cat $(\mathfrak{G}_{CF\downarrow CF} \mathfrak{H}) \mapsto\mapsto_{C.iso\alpha} (op\text{-}cf\ \mathfrak{H})_{CF\downarrow CF} (op\text{-}cf\ \mathfrak{G})$
⟨*proof*⟩

lemma *op-cf-comma-is-iso-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' =$ *op-cat* $(\mathfrak{G}_{CF\downarrow CF} \mathfrak{H})$
and $\mathfrak{B}' = (op\text{-}cf\ \mathfrak{H})_{CF\downarrow CF} (op\text{-}cf\ \mathfrak{G})$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H} : \mathfrak{A}' \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemma *op-cf-comma-is-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H}$:
$op\text{-}cat\ (\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}) \mapsto\mapsto_{C\alpha} (op\text{-}cf\ \mathfrak{H})\ _{CF}{\downarrow}_{CF}\ (op\text{-}cf\ \mathfrak{G})$
⟨*proof*⟩

lemma *op-cf-comma-is-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = op\text{-}cat\ (\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H})$
and $\mathfrak{B}' = (op\text{-}cf\ \mathfrak{H})\ _{CF}{\downarrow}_{CF}\ (op\text{-}cf\ \mathfrak{G})$
shows *op-cf-comma* $\mathfrak{G}$ $\mathfrak{H} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

14.4 Projections for a comma category

14.4.1 Definitions and elementary properties

See Chapter II-6 in [7].

definition *cf-comma-proj-left* :: $V \Rightarrow V \Rightarrow V$ (‹(- $_{CF}\sqcap$ -)› [*1000*, *1000*] *999*)
where $\mathfrak{G}\ _{CF}\sqcap\ \mathfrak{H} =$
[
$(\lambda a{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!).\ a(\!|0|\!))$,
$(\lambda f{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!).\ f(\!|2_{\mathbb{N}}|\!)(\!|0|\!))$,
$\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}$,
$\mathfrak{G}(\!|HomDom|\!)$
]$_\circ$

definition *cf-comma-proj-right* :: $V \Rightarrow V \Rightarrow V$ (‹(- $\sqcap_{CF}$ -)› [*1000*, *1000*] *999*)
where $\mathfrak{G} \sqcap_{CF} \mathfrak{H} =$
[
$(\lambda a{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!).\ a(\!|1_{\mathbb{N}}|\!))$,
$(\lambda f{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!).\ f(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!))$,
$\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}$,
$\mathfrak{H}(\!|HomDom|\!)$
]$_\circ$

Components.

lemma *cf-comma-proj-left-components*:
shows $\mathfrak{G}\ _{CF}\sqcap\ \mathfrak{H}(\!|ObjMap|\!) = (\lambda a{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!).\ a(\!|0|\!))$
and $\mathfrak{G}\ _{CF}\sqcap\ \mathfrak{H}(\!|ArrMap|\!) = (\lambda f{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!).\ f(\!|2_{\mathbb{N}}|\!)(\!|0|\!))$
and $\mathfrak{G}\ _{CF}\sqcap\ \mathfrak{H}(\!|HomDom|\!) = \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}$
and $\mathfrak{G}\ _{CF}\sqcap\ \mathfrak{H}(\!|HomCod|\!) = \mathfrak{G}(\!|HomDom|\!)$
⟨*proof*⟩

lemma *cf-comma-proj-right-components*:
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|ObjMap|\!) = (\lambda a{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Obj|\!).\ a(\!|1_{\mathbb{N}}|\!))$
and $\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|ArrMap|\!) = (\lambda f{\in_\circ}\mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}(\!|Arr|\!).\ f(\!|2_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!))$
and $\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|HomDom|\!) = \mathfrak{G}\ _{CF}{\downarrow}_{CF}\ \mathfrak{H}$
and $\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|HomCod|\!) = \mathfrak{H}(\!|HomDom|\!)$
⟨*proof*⟩

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

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

lemmas *cf-comma-proj-left-components′* =
cf-comma-proj-left-components[*of* $\mathfrak{G}$ $\mathfrak{H}$, *unfolded* $\mathfrak{G}$.*cf-HomDom*]

lemmas *cf-comma-proj-right-components′* =
cf-comma-proj-right-components[*of* $\mathfrak{G}$ $\mathfrak{H}$, *unfolded* $\mathfrak{H}$.*cf-HomDom*]

lemmas [*cat-comma-cs-simps*] =
cf-comma-proj-left-components′(*3*,*4*)
cf-comma-proj-right-components′(*3*,*4*)

end

14.4.2 Object map

mk-VLambda *cf-comma-proj-left-components*(*1*)
|*vsv cf-comma-proj-left-ObjMap-vsv*[*cat-comma-cs-intros*]|
|*vdomain cf-comma-proj-left-ObjMap-vdomain*[*cat-comma-cs-simps*]|

mk-VLambda *cf-comma-proj-right-components*(*1*)
|*vsv cf-comma-proj-right-ObjMap-vsv*[*cat-comma-cs-intros*]|
|*vdomain cf-comma-proj-right-ObjMap-vdomain*[*cat-comma-cs-simps*]|

lemma *cf-comma-proj-left-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a, b, f]_\circ$ **and** $[a, b, f]_\circ \in_\circ \mathfrak{G}_{\ CF{\downarrow}CF}\ \mathfrak{H}(\!|Obj|\!)$
shows $\mathfrak{G}\ {}_{CF}\prod\ \mathfrak{H}(\!|ObjMap|\!)(\!|A|\!) = a$
⟨*proof*⟩

lemma *cf-comma-proj-right-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a, b, f]_\circ$ **and** $[a, b, f]_\circ \in_\circ \mathfrak{G}_{\ CF{\downarrow}CF}\ \mathfrak{H}(\!|Obj|\!)$
shows $\mathfrak{G} \prod_{CF}\ \mathfrak{H}(\!|ObjMap|\!)(\!|A|\!) = b$
⟨*proof*⟩

lemma *cf-comma-proj-left-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G}\ {}_{CF}\prod\ \mathfrak{H}(\!|ObjMap|\!)) \subseteq_\circ \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-comma-proj-right-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G} \prod_{CF}\ \mathfrak{H}(\!|ObjMap|\!)) \subseteq_\circ \mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

14.4.3 Arrow map

mk-VLambda *cf-comma-proj-left-components*(*2*)
|*vsv cf-comma-proj-left-ArrMap-vsv*[*cat-comma-cs-intros*]|
|*vdomain cf-comma-proj-left-ArrMap-vdomain*[*cat-comma-cs-simps*]|

mk-VLambda *cf-comma-proj-right-components*(*2*)
|*vsv cf-comma-proj-right-ArrMap-vsv*[*cat-comma-cs-intros*]|
|*vdomain cf-comma-proj-right-ArrMap-vdomain*[*cat-comma-cs-simps*]|

lemma *cf-comma-proj-left-ArrMap-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, [g, h]_\circ]_\circ$ **and** $[A, B, [g, h]_\circ]_\circ \in_\circ \mathfrak{G}_{\ CF{\downarrow}CF}\ \mathfrak{H}(\!|Arr|\!)$

shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H}(\!|ArrMap|\!)(\!|ABF|\!) = g$
⟨*proof*⟩

lemma *cf-comma-proj-right-ArrMap-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, [g, h]_\circ]_\circ$
and $[A, B, [g, h]_\circ]_\circ \in_\circ \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}(\!|Arr|\!)$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|ArrMap|\!)(\!|ABF|\!) = h$
⟨*proof*⟩

lemma *cf-comma-proj-left-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H}(\!|ArrMap|\!)) \subseteq_\circ \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-comma-proj-right-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\mathfrak{G} \sqcap_{CF} \mathfrak{H}(\!|ArrMap|\!)) \subseteq_\circ \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

14.4.4 Projections for a comma category are functors

lemma *cf-comma-proj-left-is-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H} : \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H} \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comma-proj-left-is-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comma-proj-right-is-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H} : \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cf-comma-proj-right-is-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

14.4.5 Opposite projections for a comma category

lemma *op-cf-comma-proj-left*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *op-cf* $(\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H}) = (\textit{op-cf}\ \mathfrak{H}) \sqcap_{CF} (\textit{op-cf}\ \mathfrak{G}) \circ_{CF} \textit{op-cf-comma}\ \mathfrak{G}\ \mathfrak{H}$
⟨*proof*⟩

lemma *op-cf-comma-proj-right*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *op-cf* $(\mathfrak{G} \sqcap_{CF} \mathfrak{H}) = (\textit{op-cf}\ \mathfrak{H})\ {}_{CF}\sqcap\ (\textit{op-cf}\ \mathfrak{G}) \circ_{CF} \textit{op-cf-comma}\ \mathfrak{G}\ \mathfrak{H}$
⟨*proof*⟩

14.4.6 Projections for a tiny comma category

lemma *cf-comma-proj-left-is-tm-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H} : \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H} \mapsto\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comma-proj-left-is-tm-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{GH} = \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H} : \mathfrak{GH} \mapsto\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comma-proj-right-is-tm-functor*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H} : \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cf-comma-proj-right-is-tm-functor'*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $\mathfrak{GH} = \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H} : \mathfrak{GH} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cf-comp-cf-comma-proj-left-is-tm-functor*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G}\ {}_{CF}\sqcap\ \mathfrak{H} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comp-cf-comma-proj-right-is-tm-functor*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{G}\ {}_{CF}\downarrow_{CF}\ \mathfrak{H}$
shows $\mathfrak{G} \sqcap_{CF} \mathfrak{H} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

14.5 Comma categories constructed from a functor and an object

14.5.1 Definitions and elementary properties

See Chapter II-6 in [7].

definition *cat-cf-obj-comma* :: $V \Rightarrow V \Rightarrow V$ (‹(- ${}_{CF}\downarrow$ -)› [*1000*, *1000*] *999*)
where $\mathfrak{F}\ {}_{CF}\downarrow\ b \equiv \mathfrak{F}\ {}_{CF}\downarrow_{CF}\ (cf\text{-}const\ (cat\text{-}1\ 0\ 0)\ (\mathfrak{F}(\lfloor HomCod \rfloor))\ b)$

definition *cat-obj-cf-comma* :: $V \Rightarrow V \Rightarrow V$ (‹(- $\downarrow_{CF}$ -)› [*1000*, *1000*] *999*)
where $b \downarrow_{CF} \mathfrak{F} \equiv (cf\text{-}const\ (cat\text{-}1\ 0\ 0)\ (\mathfrak{F}(\lfloor HomCod \rfloor))\ b)\ {}_{CF}\downarrow_{CF}\ \mathfrak{F}$

Alternative forms of the definitions.

lemma (**in** *is-functor*) *cat-cf-obj-comma-def*:
$\mathfrak{F}\ {}_{CF}\downarrow\ b = \mathfrak{F}\ {}_{CF}\downarrow_{CF}\ (cf\text{-}const\ (cat\text{-}1\ 0\ 0)\ \mathfrak{B}\ b)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-def*:

$b \downarrow_{CF} \mathfrak{F} = (\mathit{cf\text{-}const}\ (\mathit{cat\text{-}1}\ 0\ 0)\ \mathfrak{B}\ b)\ {}_{CF}{\downarrow_{CF}}\ \mathfrak{F}$
⟨*proof*⟩

Size.

lemma *small-cat-cf-obj-comma-Obj*[*simp*]:
$\mathit{small}\ \{[a,\ 0,\ f]_\circ \mid a\ f.\ a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!) \wedge f : x \mapsto_{\mathfrak{C}} \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|a|\!)\}$
(**is** ‹*small ?afs*›)
⟨*proof*⟩

lemma *small-cat-obj-cf-comma-Obj*[*simp*]:
$\mathit{small}\ \{[0,\ b,\ f]_\circ \mid b\ f.\ b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!) \wedge f : x \mapsto_{\mathfrak{C}} \mathfrak{G}(\!|\mathit{ObjMap}|\!)(\!|b|\!)\}$
(**is** ‹*small ?bfs*›)
⟨*proof*⟩

14.5.2 Objects

lemma (**in** *is-functor*) *cat-cf-obj-comma-ObjI*[*cat-comma-cs-intros*]:
assumes $A = [a,\ 0,\ f]_\circ$ **and** $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$ **and** $f : \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
shows $A \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-cf-obj-comma-ObjI*

lemma (**in** *is-functor*) *cat-obj-cf-comma-ObjI*[*cat-comma-cs-intros*]:
assumes $A = [0,\ a,\ f]_\circ$ **and** $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$ **and** $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!)$
shows $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-obj-cf-comma-ObjI*

lemma (**in** *is-functor*) *cat-cf-obj-comma-ObjD*[*dest*]:
assumes $[a,\ b',\ f]_\circ \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|\mathit{Obj}|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
shows $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$ **and** $b' = 0$ **and** $f : \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
⟨*proof*⟩

lemmas [*dest*] = *is-functor.cat-cf-obj-comma-ObjD*[*rotated 1*]

lemma (**in** *is-functor*) *cat-obj-cf-comma-ObjD*[*dest*]:
assumes $[b',\ a,\ f]_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|\mathit{Obj}|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
shows $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$ **and** $b' = 0$ **and** $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!)$
⟨*proof*⟩

lemmas [*dest*] = *is-functor.cat-obj-cf-comma-ObjD*[*rotated 1*]

lemma (**in** *is-functor*) *cat-cf-obj-comma-ObjE*[*elim*]:
assumes $A \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|\mathit{Obj}|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
obtains $a\ f$
where $A = [a,\ 0,\ f]_\circ$
and $a \in_\circ \mathfrak{A}(\!|\mathit{Obj}|\!)$
and $f : \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-cf-obj-comma-ObjE*[*rotated 1*]

lemma (**in** *is-functor*) *cat-obj-cf-comma-ObjE*[*elim*]:
assumes $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|\mathit{Obj}|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|\mathit{Obj}|\!)$
obtains $a\ f$
where $A = [0,\ a,\ f]_\circ$

and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-obj-cf-comma-ObjE*[*rotated 1*]

14.5.3 Arrows

lemma (**in** *is-functor*) *cat-cf-obj-comma-ArrI*[*cat-comma-cs-intros*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $F = [A, B, [g, 0]_\circ]_\circ$
and $A = [a, 0, f]_\circ$
and $B = [a', 0, f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = f$
shows $F \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-cf-obj-comma-ArrI*

lemma (**in** *is-functor*) *cat-obj-cf-comma-ArrI*[*cat-comma-cs-intros*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $F = [A, B, [0, g]_\circ]_\circ$
and $A = [0, a, f]_\circ$
and $B = [0, a', f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!)$
and $\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} f = f'$
shows $F \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-obj-cf-comma-ArrI*

lemma (**in** *is-functor*) *cat-cf-obj-comma-ArrE*[*elim*]:
assumes $F \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|Arr|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
obtains $A\ B\ a\ f\ a'\ f'\ g$
where $F = [A, B, [g, 0]_\circ]_\circ$
and $A = [a, 0, f]_\circ$
and $B = [a', 0, f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = f$
and $A \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|Obj|\!)$
and $B \in_\circ \mathfrak{F}\ {}_{CF}{\downarrow}\ b(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-cf-obj-comma-ArrE*[*rotated 1*]

lemma (**in** *is-functor*) *cat-obj-cf-comma-ArrE*[*elim*]:
assumes $F \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
obtains $A\ B\ a\ f\ a'\ f'\ g$
where $F = [A, B, [0, g]_\circ]_\circ$
and $A = [0, a, f]_\circ$
and $B = [0, a', f']_\circ$

and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a' \rfloor)$
and $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} f = f'$
and $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$
and $B \in_\circ b \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-obj-cf-comma-ArrE*

lemma (**in** *is-functor*) *cat-cf-obj-comma-ArrD*[*dest*]:
assumes $[[a, b', f]_\circ, [a', b'', f']_\circ, [g, h]_\circ]_\circ \in_\circ \mathfrak{F}\ _{CF}\downarrow b(\lfloor Arr \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows $b' = 0$
and $b'' = 0$
and $h = 0$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a' \rfloor) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) = f$
and $[a, b', f]_\circ \in_\circ \mathfrak{F}\ _{CF}\downarrow b(\lfloor Obj \rfloor)$
and $[a', b'', f']_\circ \in_\circ \mathfrak{F}\ _{CF}\downarrow b(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas [*dest*] = *is-functor.cat-cf-obj-comma-ArrD*[*rotated 1*]

lemma (**in** *is-functor*) *cat-obj-cf-comma-ArrD*[*dest*]:
assumes $[[b', a, f]_\circ, [b'', a', f']_\circ, [h, g]_\circ]_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\lfloor Arr \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows $b' = 0$
and $b'' = 0$
and $h = 0$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a' \rfloor)$
and $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} f = f'$
and $[b', a, f]_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$
and $[b'', a', f']_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas [*dest*] = *is-functor.cat-obj-cf-comma-ArrD*

14.5.4 Domain

lemma *cat-cf-obj-comma-Dom-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{F}\ _{CF}\downarrow b(\lfloor Dom \rfloor))$
⟨*proof*⟩

lemma *cat-cf-obj-comma-Dom-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ (\mathfrak{F}\ _{CF}\downarrow b(\lfloor Dom \rfloor)) = \mathfrak{F}\ _{CF}\downarrow b(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *cat-cf-obj-comma-Dom-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, F]_\circ$ **and** $ABF \in_\circ \mathfrak{F}\ _{CF}\downarrow b(\lfloor Arr \rfloor)$
shows $\mathfrak{F}\ _{CF}\downarrow b(\lfloor Dom \rfloor)(\lfloor ABF \rfloor) = A$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-cf-obj-comma-Dom-vrange*:
assumes $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$

shows $\mathcal{R}_{\circ}\ (\mathfrak{F}_{\ CF\downarrow}\ b(\!|Dom|\!)) \subseteq_{\circ} \mathfrak{F}_{\ CF\downarrow}\ b(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Dom-vsv*[*cat-comma-cs-intros*]: *vsv* $(b \downarrow_{CF} \mathfrak{F}(\!|Dom|\!))$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Dom-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}\ (b \downarrow_{CF} \mathfrak{F}(\!|Dom|\!)) = b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Dom-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, F]_{\circ}$ **and** $ABF \in_{\circ} b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
shows $b \downarrow_{CF} \mathfrak{F}(\!|Dom|\!)(\!|ABF|\!) = A$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-Dom-vrange*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{R}_{\circ}\ (b \downarrow_{CF} \mathfrak{F}(\!|Dom|\!)) \subseteq_{\circ} b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

14.5.5 Codomain

lemma *cat-cf-obj-comma-Cod-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{F}_{\ CF\downarrow}\ b(\!|Cod|\!))$
⟨*proof*⟩

lemma *cat-cf-obj-comma-Cod-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}\ (\mathfrak{F}_{\ CF\downarrow}\ b(\!|Cod|\!)) = \mathfrak{F}_{\ CF\downarrow}\ b(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-cf-obj-comma-Cod-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, F]_{\circ}$ **and** $ABF \in_{\circ} \mathfrak{F}_{\ CF\downarrow}\ b(\!|Arr|\!)$
shows $\mathfrak{F}_{\ CF\downarrow}\ b(\!|Cod|\!)(\!|ABF|\!) = B$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-cf-obj-comma-Cod-vrange*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{R}_{\circ}\ (\mathfrak{F}_{\ CF\downarrow}\ b(\!|Cod|\!)) \subseteq_{\circ} \mathfrak{F}_{\ CF\downarrow}\ b(\!|Obj|\!)$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Cod-vsv*[*cat-comma-cs-intros*]: *vsv* $(b \downarrow_{CF} \mathfrak{F}(\!|Cod|\!))$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Cod-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}\ (b \downarrow_{CF} \mathfrak{F}(\!|Cod|\!)) = b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Cod-app*[*cat-comma-cs-simps*]:
assumes $ABF = [A, B, F]_{\circ}$ **and** $ABF \in_{\circ} b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
shows $b \downarrow_{CF} \mathfrak{F}(\!|Cod|\!)(\!|ABF|\!) = B$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-Cod-vrange*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{R}_{\circ}\ (b \downarrow_{CF} \mathfrak{F}(\!|Dom|\!)) \subseteq_{\circ} b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

14.5.6 Arrow with a domain and a codomain

lemma (**in** *is-functor*) *cat-cf-obj-comma-is-arrI*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $ABF = [A, B, F]_{\circ}$
and $A = [a, 0, f]_{\circ}$
and $B = [a', 0, f']_{\circ}$
and $F = [g, 0]_{\circ}$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = f$
shows $ABF : A \mapsto_{\mathfrak{F}\ CF\downarrow\ b} B$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-cf-obj-comma-is-arrI*

lemma (**in** *is-functor*) *cat-obj-cf-comma-is-arrI*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $ABF = [A, B, F]_{\circ}$
and $A = [0, a, f]_{\circ}$
and $B = [0, a', f']_{\circ}$
and $F = [0, g]_{\circ}$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!)$
and $\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} f = f'$
shows $ABF : A \mapsto_{b\ \downarrow_{CF}\ \mathfrak{F}} B$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cat-obj-cf-comma-is-arrI*

lemma (**in** *is-functor*) *cat-cf-obj-comma-is-arrD*[*dest*]:
assumes $[[a, b', f]_{\circ}, [a', b'', f']_{\circ}, [g, h]_{\circ}]_{\circ} :$
$[a, b', f]_{\circ} \mapsto_{\mathfrak{F}\ CF\downarrow\ b} [a', b'', f']_{\circ}$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $b' = 0$
and $b'' = 0$
and $h = 0$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = f$
and $[a, b', f]_{\circ} \in_{\circ} \mathfrak{F}\ CF\downarrow\ b(\!|Obj|\!)$
and $[a', b'', f']_{\circ} \in_{\circ} \mathfrak{F}\ CF\downarrow\ b(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-is-arrD*[*dest*]:
assumes $[[b', a, f]_{\circ}, [b'', a', f']_{\circ}, [h, g]_{\circ}]_{\circ} :$
$[b', a, f]_{\circ} \mapsto_{b\ \downarrow_{CF}\ \mathfrak{F}} [b'', a', f']_{\circ}$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $b' = 0$
and $b'' = 0$
and $h = 0$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!)$
and $\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} f = f'$

and $[b', a, f]_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
and $[b'', a', f']_\circ \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*dest*] = *is-functor.cat-obj-cf-comma-is-arrD*

lemma (in *is-functor*) *cat-cf-obj-comma-is-arrE*[*elim*]:
assumes $ABF : A \mapsto_{\mathfrak{F}\ CF\downarrow\ b} B$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
obtains $a\ f\ a'\ f'\ g$
where $ABF = [[a, 0, f]_\circ, [a', 0, f']_\circ, [g, 0]_\circ]_\circ$
and $A = [a, 0, f]_\circ$
and $B = [a', 0, f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} b$
and $f' : \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!) \mapsto_{\mathfrak{B}} b$
and $f' \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = f$
and $A \in_\circ \mathfrak{F}\ _{CF\downarrow}\ b(\!|Obj|\!)$
and $B \in_\circ \mathfrak{F}\ _{CF\downarrow}\ b(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-cf-obj-comma-is-arrE*

lemma (in *is-functor*) *cat-obj-cf-comma-is-arrE*[*elim*]:
assumes $ABF : A \mapsto_{b \downarrow_{CF}\ \mathfrak{F}} B$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
obtains $a\ f\ a'\ f'\ g$
where $ABF = [[0, a, f]_\circ, [0, a', f']_\circ, [0, g]_\circ]_\circ$
and $A = [0, a, f]_\circ$
and $B = [0, a', f']_\circ$
and $g : a \mapsto_{\mathfrak{A}} a'$
and $f : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
and $f' : b \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|a'|\!)$
and $\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} f = f'$
and $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
and $B \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*elim*] = *is-functor.cat-obj-cf-comma-is-arrE*

14.5.7 Composition

lemma *cat-cf-obj-comma-Comp-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{F}\ _{CF\downarrow}\ b(\!|Comp|\!))$
⟨*proof*⟩

lemma *cat-obj-cf-comma-Comp-vsv*[*cat-comma-cs-intros*]: *vsv* $(b \downarrow_{CF} \mathfrak{F}(\!|Comp|\!))$
⟨*proof*⟩

lemma (in *is-functor*) *cat-cf-obj-comma-Comp-app*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $BCG = [B, C, [g', h']_\circ]_\circ$
and $ABF = [A, B, [g, h]_\circ]_\circ$
and $BCG : B \mapsto_{\mathfrak{F}\ CF\downarrow\ b} C$
and $ABF : A \mapsto_{\mathfrak{F}\ CF\downarrow\ b} B$
shows $BCG \circ_{A\mathfrak{F}\ CF\downarrow\ b} ABF = [A, C, [g' \circ_{A\mathfrak{A}} g, 0]_\circ]_\circ$
⟨*proof*⟩

lemma (in *is-functor*) *cat-obj-cf-comma-Comp-app*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $BCG = [B, C, [h', g']_\circ]_\circ$

and $ABF = [A,\ B,\ [h,\ g]_\circ]_\circ$
and $BCG : B \mapsto_{b \downarrow_{CF} \mathfrak{F}} C$
and $ABF : A \mapsto_{b \downarrow_{CF} \mathfrak{F}} B$
shows $BCG \circ_{A\, b \downarrow_{CF} \mathfrak{F}} ABF = [A,\ C,\ [0,\ g' \circ_{A\mathfrak{A}} g]_\circ]_\circ$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-cf-obj-comma-Comp-is-arr*[*cat-comma-cs-intros*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $BCG : B \mapsto_{\mathfrak{F}\ CF\downarrow\ b} C$
and $ABF : A \mapsto_{\mathfrak{F}\ CF\downarrow\ b} B$
shows $BCG \circ_{A\mathfrak{F}\ CF\downarrow\ b} ABF : A \mapsto_{\mathfrak{F}\ CF\downarrow\ b} C$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-Comp-is-arr*[*cat-comma-cs-intros*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $BCG : B \mapsto_{b \downarrow_{CF} \mathfrak{F}} C$
and $ABF : A \mapsto_{b \downarrow_{CF} \mathfrak{F}} B$
shows $BCG \circ_{A\, b \downarrow_{CF} \mathfrak{F}} ABF : A \mapsto_{b \downarrow_{CF} \mathfrak{F}} C$
⟨*proof*⟩

14.5.8 Identity

lemma *cat-cf-obj-comma-CId-vsv*[*cat-comma-cs-intros*]: *vsv* $(\mathfrak{F}\ _{CF\downarrow}\ b(\!|CId|\!))$
⟨*proof*⟩

lemma *cat-obj-cf-comma-CId-vsv*[*cat-comma-cs-intros*]: *vsv* $(b \downarrow_{CF} \mathfrak{F}(\!|CId|\!))$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-cf-obj-comma-CId-vdomain*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{D}_\circ\ (\mathfrak{F}\ _{CF\downarrow}\ b(\!|CId|\!)) = \mathfrak{F}\ _{CF\downarrow}\ b(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-CId-vdomain*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{D}_\circ\ (b \downarrow_{CF} \mathfrak{F}(\!|CId|\!)) = b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-cf-obj-comma-CId-app*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $A = [a,\ b',\ f]_\circ$ **and** $A \in_\circ \mathfrak{F}\ _{CF\downarrow}\ b(\!|Obj|\!)$
shows $\mathfrak{F}\ _{CF\downarrow}\ b(\!|CId|\!)(\!|A|\!) = [A,\ A,\ [\mathfrak{A}(\!|CId|\!)(\!|a|\!),\ 0]_\circ]_\circ$
⟨*proof*⟩

lemma (**in** *is-functor*) *cat-obj-cf-comma-CId-app*[*cat-comma-cs-simps*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $A = [b',\ a,\ f]_\circ$ **and** $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
shows $b \downarrow_{CF} \mathfrak{F}(\!|CId|\!)(\!|A|\!) = [A,\ A,\ [0,\ \mathfrak{A}(\!|CId|\!)(\!|a|\!)]_\circ]_\circ$
⟨*proof*⟩

14.5.9 Comma categories constructed from a functor and an object are categories

lemma (**in** *is-functor*) *category-cat-cf-obj-comma*[*cat-comma-cs-intros*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *category* α $(\mathfrak{F}\ _{CF\downarrow}\ b)$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.category-cat-cf-obj-comma*

lemma (**in** *is-functor*) *category-cat-obj-cf-comma*[*cat-comma-cs-intros*]:

assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *category* α $(b \downarrow_{CF} \mathfrak{F})$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.category-cat-obj-cf-comma*

14.5.10 Tiny comma categories constructed from a functor and an object

lemma (**in** *is-tm-functor*) *tiny-category-cat-cf-obj-comma*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *tiny-category* α $(\mathfrak{F}\ {}_{CF}\!\downarrow b)$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tiny-category-cat-obj-cf-comma*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *tiny-category* α $(b \downarrow_{CF} \mathfrak{F})$
⟨*proof*⟩

14.6 Opposite comma category functors for the comma categories constructed from a functor and an object

14.6.1 Definitions and elementary properties

definition *op-cf-obj-comma* :: $V \Rightarrow V \Rightarrow V$
where *op-cf-obj-comma* $\mathfrak{F}$ b =
op-cf-comma $\mathfrak{F}$ (*cf-const* (*cat-1 0 0*) ($\mathfrak{F}(\!|HomCod|\!)$) b)

definition *op-obj-cf-comma* :: $V \Rightarrow V \Rightarrow V$
where *op-obj-cf-comma* b $\mathfrak{F}$ =
op-cf-comma (*cf-const* (*cat-1 0 0*) ($\mathfrak{F}(\!|HomCod|\!)$) b) $\mathfrak{F}$

Alternative forms of the definitions.

lemma (**in** *is-functor*) *op-cf-obj-comma-def*:
op-cf-obj-comma $\mathfrak{F}$ b = *op-cf-comma* $\mathfrak{F}$ (*cf-const* (*cat-1 0 0*) $\mathfrak{B}$ b)
⟨*proof*⟩

lemma (**in** *is-functor*) *op-obj-cf-comma-def*:
op-obj-cf-comma b $\mathfrak{F}$ = *op-cf-comma* (*cf-const* (*cat-1 0 0*) $\mathfrak{B}$ b) $\mathfrak{F}$
⟨*proof*⟩

14.6.2 Object map

lemma *op-cf-obj-comma-ObjMap-vsv*[*cat-comma-cs-intros*]:
vsv (*op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *op-obj-cf-comma-ObjMap-vsv*[*cat-comma-cs-intros*]:
vsv (*op-obj-cf-comma* b $\mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma (**in** *is-functor*) *op-cf-obj-comma-ObjMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}$ (*op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ObjMap|\!)$) = $\mathfrak{F}\ {}_{CF}\!\downarrow b(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *is-functor*) *op-obj-cf-comma-ObjMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}$ (*op-obj-cf-comma* b $\mathfrak{F}(\!|ObjMap|\!)$) = $b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
⟨*proof*⟩

lemma (in *is-functor*) *op-cf-obj-comma-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a, 0, f]_\circ$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $A \in_\circ \mathfrak{F}\ _{CF}{\downarrow}\ b(\!|Obj|\!)$
shows *op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ObjMap|\!)(\!|A|\!) = [0, a, f]_\circ$
⟨*proof*⟩

lemma (in *is-functor*) *op-obj-cf-comma-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [0, a, f]_\circ$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $A \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Obj|\!)$
shows *op-obj-cf-comma* b $\mathfrak{F}$ $(\!|ObjMap|\!)(\!|A|\!) = [a, 0, f]_\circ$
⟨*proof*⟩

14.6.3 Arrow map

lemma *op-cf-obj-comma-ArrMap-vsv*[*cat-comma-cs-intros*]:
vsv (*op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *op-obj-cf-comma-ArrMap-vsv*[*cat-comma-cs-intros*]:
vsv (*op-obj-cf-comma* b $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma (in *is-functor*) *op-cf-obj-comma-ArrMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ$ (*op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ArrMap|\!)$) = $\mathfrak{F}\ _{CF}{\downarrow}\ b(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.op-cf-obj-comma-ArrMap-vdomain*

lemma (in *is-functor*) *op-obj-cf-comma-ArrMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ$ (*op-obj-cf-comma* b $\mathfrak{F}(\!|ArrMap|\!)$) = $b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.op-obj-cf-comma-ArrMap-vdomain*

lemma (in *is-functor*) *op-cf-obj-comma-ArrMap-app*[*cat-comma-cs-simps*]:
assumes $ABF = [[a, 0, f]_\circ, [a', 0, f']_\circ, [g, 0]_\circ]_\circ$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $ABF \in_\circ \mathfrak{F}\ _{CF}{\downarrow}\ b(\!|Arr|\!)$
shows *op-cf-obj-comma* $\mathfrak{F}$ $b(\!|ArrMap|\!)(\!|ABF|\!) = [[0, a', f']_\circ, [0, a, f]_\circ, [0, g]_\circ]_\circ$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.op-cf-obj-comma-ArrMap-app*

lemma (in *is-functor*) *op-obj-cf-comma-ArrMap-app*[*cat-comma-cs-simps*]:
assumes $ABF = [[0, a, f]_\circ, [0, a', f']_\circ, [0, h]_\circ]_\circ$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $ABF \in_\circ b \downarrow_{CF} \mathfrak{F}(\!|Arr|\!)$
shows *op-obj-cf-comma* b $\mathfrak{F}(\!|ArrMap|\!)(\!|ABF|\!) = [[a', 0, f']_\circ, [a, 0, f]_\circ, [h, 0]_\circ]_\circ$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.op-obj-cf-comma-ArrMap-app*

14.6.4 Opposite comma category functors for the comma categories constructed from a functor and an object are isomorphisms of categories

lemma (in *is-functor*) *op-cf-obj-comma-is-iso-functor*:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows *op-cf-obj-comma* $\mathfrak{F}$ b : *op-cat* $(\mathfrak{F}\ _{CF}{\downarrow}\ b) \mapsto\mapsto_{C.iso\alpha} b \downarrow_{CF} (op\text{-}cf\ \mathfrak{F})$
⟨*proof*⟩

lemma (in *is-functor*) *op-cf-obj-comma-is-iso-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}' = op\text{-}cat\ (\mathfrak{F}\ _{CF}\!\downarrow\ b)$
and $\mathfrak{B}' = b \downarrow_{CF} (op\text{-}cf\ \mathfrak{F})$
shows $op\text{-}cf\text{-}obj\text{-}comma\ \mathfrak{F}\ b : \mathfrak{A}' \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.op-cf-obj-comma-is-iso-functor'*

lemma (in *is-functor*) *op-cf-obj-comma-is-functor*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $op\text{-}cf\text{-}obj\text{-}comma\ \mathfrak{F}\ b : op\text{-}cat\ (\mathfrak{F}\ _{CF}\!\downarrow\ b) \mapsto\mapsto_{C\alpha} b \downarrow_{CF} (op\text{-}cf\ \mathfrak{F})$
⟨*proof*⟩

lemma (in *is-functor*) *op-cf-obj-comma-is-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}' = op\text{-}cat\ (\mathfrak{F}\ _{CF}\!\downarrow\ b)$
and $\mathfrak{B}' = b \downarrow_{CF} (op\text{-}cf\ \mathfrak{F})$
shows $op\text{-}cf\text{-}obj\text{-}comma\ \mathfrak{F}\ b : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.op-cf-obj-comma-is-functor'*

lemma (in *is-functor*) *op-obj-cf-comma-is-iso-functor*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $op\text{-}obj\text{-}cf\text{-}comma\ b\ \mathfrak{F} : op\text{-}cat\ (b \downarrow_{CF} \mathfrak{F}) \mapsto\mapsto_{C.iso\alpha} (op\text{-}cf\ \mathfrak{F})\ _{CF}\!\downarrow\ b$
⟨*proof*⟩

lemma (in *is-functor*) *op-obj-cf-comma-is-iso-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}' = op\text{-}cat\ (b \downarrow_{CF} \mathfrak{F})$
and $\mathfrak{B}' = (op\text{-}cf\ \mathfrak{F})\ _{CF}\!\downarrow\ b$
shows $op\text{-}obj\text{-}cf\text{-}comma\ b\ \mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C.iso\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.op-obj-cf-comma-is-iso-functor'*

lemma (in *is-functor*) *op-obj-cf-comma-is-functor*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $op\text{-}obj\text{-}cf\text{-}comma\ b\ \mathfrak{F} : op\text{-}cat\ (b \downarrow_{CF} \mathfrak{F}) \mapsto\mapsto_{C\alpha} (op\text{-}cf\ \mathfrak{F})\ _{CF}\!\downarrow\ b$
⟨*proof*⟩

lemma (in *is-functor*) *op-obj-cf-comma-is-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}' = op\text{-}cat\ (b \downarrow_{CF} \mathfrak{F})$
and $\mathfrak{B}' = (op\text{-}cf\ \mathfrak{F})\ _{CF}\!\downarrow\ b$
shows $op\text{-}obj\text{-}cf\text{-}comma\ b\ \mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

14.7 Projections for comma categories constructed from a functor and an object

14.7.1 Definitions and elementary properties

definition *cf-cf-obj-comma-proj* :: $V \Rightarrow V \Rightarrow V$ (‹(- $_{CF}\!\sqcap_{O}$ -)› [*1000*, *1000*] *999*)
where $\mathfrak{F}\ _{CF}\!\sqcap_{O}\ b \equiv \mathfrak{F}\ _{CF}\!\sqcap\ (cf\text{-}const\ (cat\text{-}1\ 0\ 0)\ (\mathfrak{F}(\!|HomCod|\!))\ b)$

definition *cf-obj-cf-comma-proj* :: $V \Rightarrow V \Rightarrow V$ (‹(- $_{O}\!\sqcap_{CF}$ -)› [*1000*, *1000*] *999*)

where $b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F} \equiv (\mathit{cf\text{-}const}\ (\mathit{cat\text{-}1\ 0\ 0})\ (\mathfrak{F}(\!|\mathit{HomCod}|\!))\ b) \sqcap_{CF} \mathfrak{F}$

Alternative forms of the definitions.

lemma (in *is-functor*) *cf-cf-obj-comma-proj-def*:
$\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b = \mathfrak{F}\ {}_{CF}\!\sqcap\ (\mathit{cf\text{-}const}\ (\mathit{cat\text{-}1\ 0\ 0})\ \mathfrak{B}\ b)$
⟨*proof*⟩

lemma (in *is-functor*) *cf-obj-cf-comma-proj-def*:
$b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F} = (\mathit{cf\text{-}const}\ (\mathit{cat\text{-}1\ 0\ 0})\ \mathfrak{B}\ b) \sqcap_{CF} \mathfrak{F}$
⟨*proof*⟩

Components.

lemma (in *is-functor*) *cf-cf-obj-comma-proj-components*[*cat-comma-cs-simps*]:
shows $\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b(\!|\mathit{HomDom}|\!) = \mathfrak{F}\ {}_{CF}\!\downarrow\ b$
and $\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b(\!|\mathit{HomCod}|\!) = \mathfrak{A}$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-obj-comma-proj-components*

lemma (in *is-functor*) *cf-obj-cf-comma-proj-components*[*cat-comma-cs-simps*]:
shows $b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F}(\!|\mathit{HomDom}|\!) = b \downarrow_{CF} \mathfrak{F}$
and $b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F}(\!|\mathit{HomCod}|\!) = \mathfrak{A}$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-obj-cf-comma-proj-components*

14.7.2 Object map

lemma *cf-cf-obj-comma-proj-ObjMap-vsv*[*cat-comma-cs-intros*]:
vsv $(\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b(\!|\mathit{ObjMap}|\!))$
⟨*proof*⟩

lemma *cf-obj-cf-comma-proj-ObjMap-vsv*[*cat-comma-cs-intros*]:
vsv $(b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!))$
⟨*proof*⟩

lemma (in *is-functor*) *cf-cf-obj-comma-proj-ObjMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}\ (\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b(\!|\mathit{ObjMap}|\!)) = \mathfrak{F}\ {}_{CF}\!\downarrow\ b(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-obj-comma-proj-ObjMap-vdomain*

lemma (in *is-functor*) *cf-obj-cf-comma-proj-ObjMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_{\circ}\ (b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!)) = b \downarrow_{CF} \mathfrak{F}(\!|\mathit{Obj}|\!)$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-obj-cf-comma-proj-ObjMap-vdomain*

lemma (in *is-functor*) *cf-cf-obj-comma-proj-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a, b', f]_{\circ}$ **and** $[a, b', f]_{\circ} \in_{\circ} \mathfrak{F}\ {}_{CF}\!\downarrow\ b(\!|\mathit{Obj}|\!)$
shows $\mathfrak{F}\ {}_{CF}\!\sqcap_{O}\ b(\!|\mathit{ObjMap}|\!)(\!|A|\!) = a$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-obj-comma-proj-ObjMap-app*

lemma (in *is-functor*) *cf-obj-cf-comma-proj-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [b', a, f]_{\circ}$ **and** $[b', a, f]_{\circ} \in_{\circ} b \downarrow_{CF} \mathfrak{F}(\!|\mathit{Obj}|\!)$
shows $b\ {}_{O}\!\sqcap_{CF}\ \mathfrak{F}(\!|\mathit{ObjMap}|\!)(\!|A|\!) = a$

⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-obj-cf-comma-proj-ObjMap-app*

14.7.3 Arrow map

lemma *cf-cf-obj-comma-proj-ArrMap-vsv*[*cat-comma-cs-intros*]:
vsv (ℱ ‹*CF*⨝*O* *b*(|*ArrMap*|))
⟨*proof*⟩

lemma *cf-obj-cf-comma-proj-ArrMap-vsv*[*cat-comma-cs-intros*]:
vsv (*b* ‹*O*⨝*CF* ℱ(|*ArrMap*|))
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-cf-obj-comma-proj-ArrMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ$ (ℱ ‹*CF*⨝*O* *b*(|*ArrMap*|)) = ℱ ‹*CF*↓ *b*(|*Arr*|)
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-obj-comma-proj-ObjMap-vdomain*

lemma (**in** *is-functor*) *cf-obj-cf-comma-proj-ArrMap-vdomain*[*cat-comma-cs-simps*]:
$\mathcal{D}_\circ$ (*b* ‹*O*⨝*CF* ℱ(|*ArrMap*|)) = *b* ↓‹*CF* ℱ(|*Arr*|)
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-obj-cf-comma-proj-ArrMap-vdomain*

lemma (**in** *is-functor*) *cf-cf-obj-comma-proj-ArrMap-app*[*cat-comma-cs-simps*]:
assumes *ABF* = [*A*, *B*, [*g*, *h*]$_\circ$]$_\circ$
and [*A*, *B*, [*g*, *h*]$_\circ$]$_\circ$ ∈$_\circ$ ℱ ‹*CF*↓ *b*(|*Arr*|)
shows ℱ ‹*CF*⨝*O* *b*(|*ArrMap*|)(|*ABF*|) = *g*
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-obj-comma-proj-ArrMap-app*

lemma (**in** *is-functor*) *cf-obj-cf-comma-proj-ArrMap-app*[*cat-comma-cs-simps*]:
assumes *ABF* = [*A*, *B*, [*g*, *h*]$_\circ$]$_\circ$
and [*A*, *B*, [*g*, *h*]$_\circ$]$_\circ$ ∈$_\circ$ *b* ↓‹*CF* ℱ(|*Arr*|)
shows *b* ‹*O*⨝*CF* ℱ(|*ArrMap*|)(|*ABF*|) = *h*
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-obj-cf-comma-proj-ArrMap-app*

14.7.4 Projections for a comma category are functors

lemma (**in** *is-functor*) *cf-cf-obj-comma-proj-is-functor*:
assumes *b* ∈$_\circ$ 𝔅(|*Obj*|)
shows ℱ ‹*CF*⨝*O* *b* : ℱ ‹*CF*↓ *b* ↦↦‹*Cα* 𝔄
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-cf-obj-comma-proj-is-functor′*[*cat-comma-cs-intros*]:
assumes *b* ∈$_\circ$ 𝔅(|*Obj*|) **and** 𝔄′ = ℱ ‹*CF*↓ *b*
shows ℱ ‹*CF*⨝*O* *b* : 𝔄′ ↦↦‹*Cα* 𝔄
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cf-cf-obj-comma-proj-is-functor′*

lemma (**in** *is-functor*) *cf-obj-cf-comma-proj-is-functor*:
assumes *b* ∈$_\circ$ 𝔅(|*Obj*|)

shows $b\ {}_{O}\!\bigsqcap_{CF}\ \mathfrak{F} : b \downarrow_{CF} \mathfrak{F} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-obj-cf-comma-proj-is-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$ **and** $\mathfrak{A}' = b \downarrow_{CF} \mathfrak{F}$
shows $b\ {}_{O}\!\bigsqcap_{CF}\ \mathfrak{F} : \mathfrak{A}' \mapsto\!\!\mapsto_{C\alpha} \mathfrak{A}$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-functor.cf-obj-cf-comma-proj-is-functor'*

14.7.5 Opposite projections for comma categories constructed from a functor and an object

lemma (**in** *is-functor*) *op-cf-cf-obj-comma-proj*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *op-cf* $(\mathfrak{F}\ {}_{CF}\!\bigsqcap_{O}\ b) = b\ {}_{O}\!\bigsqcap_{CF}\ (\textit{op-cf}\ \mathfrak{F}) \circ_{CF} \textit{op-cf-obj-comma}\ \mathfrak{F}\ b$
⟨*proof*⟩

lemma (**in** *is-functor*) *op-cf-obj-cf-comma-proj*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *op-cf* $(b\ {}_{O}\!\bigsqcap_{CF}\ \mathfrak{F}) = (\textit{op-cf}\ \mathfrak{F})\ {}_{CF}\!\bigsqcap_{O}\ b \circ_{CF} \textit{op-obj-cf-comma}\ b\ \mathfrak{F}$
⟨*proof*⟩

14.7.6 Projections for a tiny comma category

lemma (**in** *is-tm-functor*) *cf-cf-obj-comma-proj-is-tm-functor*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{F}\ {}_{CF}\!\bigsqcap_{O}\ b : \mathfrak{F}\ {}_{CF}\!\downarrow\ b \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *cf-cf-obj-comma-proj-is-tm-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$ **and** $\mathfrak{F}b = \mathfrak{F}\ {}_{CF}\!\downarrow\ b$
shows $\mathfrak{F}\ {}_{CF}\!\bigsqcap_{O}\ b : \mathfrak{F}b \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-tm-functor.cf-cf-obj-comma-proj-is-tm-functor'*

lemma (**in** *is-tm-functor*) *cf-obj-cf-comma-proj-is-tm-functor*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $b\ {}_{O}\!\bigsqcap_{CF}\ \mathfrak{F} : b \downarrow_{CF} \mathfrak{F} \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *cf-obj-cf-comma-proj-is-tm-functor'*[*cat-comma-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$ **and** $\mathfrak{A}' = b \downarrow_{CF} \mathfrak{F}$
shows $b\ {}_{O}\!\bigsqcap_{CF}\ \mathfrak{F} : \mathfrak{A}' \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemmas [*cat-comma-cs-intros*] = *is-tm-functor.cf-obj-cf-comma-proj-is-tm-functor'*

lemma *cf-comp-cf-cf-obj-comma-proj-is-tm-functor*[*cat-comma-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{J} \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{G}\ {}_{CF}\!\downarrow\ c$
and $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathfrak{G}\ {}_{CF}\!\bigsqcap_{O}\ c \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\!\!\mapsto_{C.tm\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comp-cf-obj-cf-comma-proj-is-tm-functor*[*cat-comma-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{C}$

and $\mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} c \downarrow_{CF} \mathfrak{H}$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $c\ {}_O\bigcap_{CF} \mathfrak{H} \circ_{CF} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

14.8 Comma functors

14.8.1 Definition and elementary properties

See Theorem 1 in Chapter X-3 in [7].

definition *cf-arr-cf-comma* :: $V \Rightarrow V \Rightarrow V$
$(\langle(\text{-}\ {}_A\downarrow_{CF}\ \text{-})\rangle\ [1000, 1000]\ 999)$
where $g\ {}_A\downarrow_{CF}\ \mathfrak{F} =$
$[$
$(\lambda A \in_\circ (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F}(\!|Obj|\!).\ [0, A(\!|1_{\mathbb{N}}|\!), A(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ),$
$($
$\lambda F \in_\circ (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F}(\!|Arr|\!).$
$[$
$[0, F(\!|0|\!)(\!|1_{\mathbb{N}}|\!), F(\!|0|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ,$
$[0, F(\!|1_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!), F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ,$
$F(\!|2_{\mathbb{N}}|\!)$
$]_\circ$
$),$
$(\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F},$
$(\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F}$
$]_\circ$

definition *cf-cf-arr-comma* :: $V \Rightarrow V \Rightarrow V$
$(\langle(\text{-}\ {}_{CF}\downarrow_A\ \text{-})\rangle\ [1000, 1000]\ 999)$
where $\mathfrak{F}\ {}_{CF}\downarrow_A\ g =$
$[$
$(\lambda A \in_\circ \mathfrak{F}\ {}_{CF}\downarrow (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|Obj|\!).\ [A(\!|0|\!), 0, g \circ_{A\mathfrak{F}(\!|HomCod|\!)} A(\!|2_{\mathbb{N}}|\!)]_\circ),$
$($
$\lambda F \in_\circ \mathfrak{F}\ {}_{CF}\downarrow (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|Arr|\!).$
$[$
$[F(\!|0|\!)(\!|0|\!), 0, g \circ_{A\mathfrak{F}(\!|HomCod|\!)} F(\!|0|\!)(\!|2_{\mathbb{N}}|\!)]_\circ,$
$[F(\!|1_{\mathbb{N}}|\!)(\!|0|\!), 0, g \circ_{A\mathfrak{F}(\!|HomCod|\!)} F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)]_\circ,$
$F(\!|2_{\mathbb{N}}|\!)$
$]_\circ$
$),$
$\mathfrak{F}\ {}_{CF}\downarrow (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!)),$
$\mathfrak{F}\ {}_{CF}\downarrow (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))$
$]_\circ$

Components.

lemma *cf-arr-cf-comma-components*:
shows $g\ {}_A\downarrow_{CF}\ \mathfrak{F}(\!|ObjMap|\!) =$
$(\lambda A \in_\circ (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F}(\!|Obj|\!).\ [0, A(\!|1_{\mathbb{N}}|\!), A(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ)$
and $g\ {}_A\downarrow_{CF}\ \mathfrak{F}(\!|ArrMap|\!) =$
$($
$\lambda F \in_\circ (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!)) \downarrow_{CF} \mathfrak{F}(\!|Arr|\!).$
$[$
$[0, F(\!|0|\!)(\!|1_{\mathbb{N}}|\!), F(\!|0|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ,$
$[0, F(\!|1_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!), F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{F}(\!|HomCod|\!)} g]_\circ,$
$F(\!|2_{\mathbb{N}}|\!)$
$]_\circ$

)
and $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|HomDom|\!) = (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))\ {\downarrow}_{CF}\ \mathfrak{F}$
and $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|HomCod|\!) = (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))\ {\downarrow}_{CF}\ \mathfrak{F}$
⟨*proof*⟩

lemma *cf-cf-arr-comma-components*:
shows $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|ObjMap|\!) =$
$(\lambda A \in_{\circ} \mathfrak{F}\ {}_{CF}{\downarrow}\ (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|Obj|\!).\ [A(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{F}(\!|HomCod|\!)}\ A(\!|2_{\mathbb{N}}|\!)]_{\circ})$
and $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|ArrMap|\!) =$
(
$\lambda F \in_{\circ} \mathfrak{F}\ {}_{CF}{\downarrow}\ (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))(\!|Arr|\!).$
[
$[F(\!|0|\!)(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{F}(\!|HomCod|\!)}\ F(\!|0|\!)(\!|2_{\mathbb{N}}|\!)]_{\circ},$
$[F(\!|1_{\mathbb{N}}|\!)(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{F}(\!|HomCod|\!)}\ F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)]_{\circ},$
$F(\!|2_{\mathbb{N}}|\!)$
]$_{\circ}$
)
and $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|HomDom|\!) = \mathfrak{F}\ {}_{CF}{\downarrow}\ (\mathfrak{F}(\!|HomCod|\!)(\!|Dom|\!)(\!|g|\!))$
and $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|HomCod|\!) = \mathfrak{F}\ {}_{CF}{\downarrow}\ (\mathfrak{F}(\!|HomCod|\!)(\!|Cod|\!)(\!|g|\!))$
⟨*proof*⟩

context *is-functor*
begin

lemma *cf-arr-cf-comma-components′*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|ObjMap|\!) = (\lambda A \in_{\circ} c'\ {\downarrow}_{CF}\ \mathfrak{F}(\!|Obj|\!).\ [0,\ A(\!|1_{\mathbb{N}}|\!),\ A(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{B}}\ g]_{\circ})$
and $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|ArrMap|\!) =$
(
$\lambda F \in_{\circ} c'\ {\downarrow}_{CF}\ \mathfrak{F}(\!|Arr|\!).$
[
$[0,\ F(\!|0|\!)(\!|1_{\mathbb{N}}|\!),\ F(\!|0|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{B}}\ g]_{\circ},$
$[0,\ F(\!|1_{\mathbb{N}}|\!)(\!|1_{\mathbb{N}}|\!),\ F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!) \circ_{A\mathfrak{B}}\ g]_{\circ},$
$F(\!|2_{\mathbb{N}}|\!)$
]$_{\circ}$
)
and [*cat-comma-cs-simps*]: $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|HomDom|\!) = c'\ {\downarrow}_{CF}\ \mathfrak{F}$
and [*cat-comma-cs-simps*]: $g\ {}_{A}{\downarrow}_{CF}\ \mathfrak{F}(\!|HomCod|\!) = c\ {\downarrow}_{CF}\ \mathfrak{F}$
⟨*proof*⟩

lemma *cf-cf-arr-comma-components′*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|ObjMap|\!) = (\lambda A \in_{\circ} \mathfrak{F}\ {}_{CF}{\downarrow}\ c(\!|Obj|\!).\ [A(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{B}}\ A(\!|2_{\mathbb{N}}|\!)]_{\circ})$
and $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|ArrMap|\!) =$
(
$\lambda F \in_{\circ} \mathfrak{F}\ {}_{CF}{\downarrow}\ c(\!|Arr|\!).$
[
$[F(\!|0|\!)(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{B}}\ F(\!|0|\!)(\!|2_{\mathbb{N}}|\!)]_{\circ},$
$[F(\!|1_{\mathbb{N}}|\!)(\!|0|\!),\ 0,\ g \circ_{A\mathfrak{B}}\ F(\!|1_{\mathbb{N}}|\!)(\!|2_{\mathbb{N}}|\!)]_{\circ},$
$F(\!|2_{\mathbb{N}}|\!)$
]$_{\circ}$
)
and [*cat-comma-cs-simps*]: $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|HomDom|\!) = \mathfrak{F}\ {}_{CF}{\downarrow}\ c$
and [*cat-comma-cs-simps*]: $\mathfrak{F}\ {}_{CF}{\downarrow}_{A}\ g(\!|HomCod|\!) = \mathfrak{F}\ {}_{CF}{\downarrow}\ c'$
⟨*proof*⟩

end

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-arr-cf-comma-components'*(*3*,*4*)
lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-arr-comma-components'*(*3*,*4*)

14.8.2 Object map

mk-VLambda *cf-arr-cf-comma-components*(*1*)[*unfolded VLambda-vid-on*[*symmetric*]]
|*vsv cf-arr-cf-comma-ObjMap-vsv*[*cat-comma-cs-intros*]|

mk-VLambda *cf-cf-arr-comma-components*(*1*)[*unfolded VLambda-vid-on*[*symmetric*]]
|*vsv cf-cf-arr-comma-ObjMap-vsv*[*cat-comma-cs-intros*]|

context *is-functor*
begin

context
fixes g c c'
assumes *g*: $g : c \mapsto_{\mathfrak{B}} c'$
begin

mk-VLambda
cf-arr-cf-comma-components'(*1*)[*OF g*, *unfolded VLambda-vid-on*[*symmetric*]]
|*vdomain cf-arr-cf-comma-ObjMap-vdomain*[*cat-comma-cs-simps*]|

mk-VLambda
cf-cf-arr-comma-components'(*1*)[*OF g*, *unfolded VLambda-vid-on*[*symmetric*]]
|*vdomain cf-cf-arr-comma-ObjMap-vdomain*[*cat-comma-cs-simps*]|

end

end

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-arr-cf-comma-ObjMap-vdomain*
lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-arr-comma-ObjMap-vdomain*

lemma (**in** *is-functor*) *cf-arr-cf-comma-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a', b', f']_{\circ}$ **and** $A \in_{\circ} c' \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$ **and** $g : c \mapsto_{\mathfrak{B}} c'$
shows $g\ {}_{A}\!\downarrow_{CF} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor A \rfloor) = [a', b', f' \circ_{A\mathfrak{B}} g]_{\circ}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-cf-arr-comma-ObjMap-app*[*cat-comma-cs-simps*]:
assumes $A = [a', b', f']_{\circ}$ **and** $A \in_{\circ} \mathfrak{F}\ {}_{CF}\!\downarrow c(\lfloor Obj \rfloor)$ **and** $g : c \mapsto_{\mathfrak{B}} c'$
shows $\mathfrak{F}\ {}_{CF}\!\downarrow_{A} g(\lfloor ObjMap \rfloor)(\lfloor A \rfloor) = [a', b', g \circ_{A\mathfrak{B}} f']_{\circ}$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-arr-cf-comma-ObjMap-app*
lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-arr-comma-ObjMap-app*

lemma (**in** *is-functor*) *cf-arr-cf-comma-ObjMap-vrange*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows $\mathcal{R}_{\circ}\ (g\ {}_{A}\!\downarrow_{CF} \mathfrak{F}(\lfloor ObjMap \rfloor)) \subseteq_{\circ} c \downarrow_{CF} \mathfrak{F}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-cf-arr-comma-ObjMap-vrange*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows $\mathcal{R}_{\circ}\ (\mathfrak{F}\ {}_{CF}\!\downarrow_{A} g(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{F}\ {}_{CF}\!\downarrow c'(\lfloor Obj \rfloor)$
⟨*proof*⟩

14.8.3 Arrow map

mk-VLambda *cf-arr-cf-comma-components*(*2*)
  |*vsv cf-arr-cf-comma-ArrMap-vsv*[*cat-comma-cs-intros*]|

mk-VLambda *cf-cf-arr-comma-components*(*2*)
  |*vsv cf-cf-arr-comma-ArrMap-vsv*[*cat-comma-cs-intros*]|

context *is-functor*
begin

context
  fixes g c c'
  assumes *g*: $g : c \mapsto_{\mathfrak{B}} c'$
begin

mk-VLambda
  cf-arr-cf-comma-components'(*2*)[*OF g*, *unfolded VLambda-vid-on*[*symmetric*]]
  |*vdomain cf-arr-cf-comma-ArrMap-vdomain*[*cat-comma-cs-simps*]|

mk-VLambda
  cf-cf-arr-comma-components'(*2*)[*OF g*, *unfolded VLambda-vid-on*[*symmetric*]]
  |*vdomain cf-cf-arr-comma-ArrMap-vdomain*[*cat-comma-cs-simps*]|

end

end

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-arr-cf-comma-ArrMap-vdomain*
lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-arr-comma-ArrMap-vdomain*

lemma (**in** *is-functor*) *cf-arr-cf-comma-ArrMap-app*[*cat-comma-cs-simps*]:
  assumes $A = [[a, b, f]_\circ, [a', b', f']_\circ, [h, k]_\circ]_\circ$
    and $[[a, b, f]_\circ, [a', b', f']_\circ, [h, k]_\circ]_\circ :$
    $[a, b, f]_\circ \mapsto_{c' \downarrow_{CF}} \mathfrak{F}\ [a', b', f']_\circ$
    and $g : c \mapsto_{\mathfrak{B}} c'$
  shows $g\ {}_{A\downarrow_{CF}}\ \mathfrak{F}(\!|ArrMap|\!)(\!|A|\!) =$
    $[[a, b, f \circ_{A\mathfrak{B}} g]_\circ, [a', b', f' \circ_{A\mathfrak{B}} g]_\circ, [h, k]_\circ]_\circ$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-arr-cf-comma-ArrMap-app*

lemma (**in** *is-functor*) *cf-cf-arr-comma-ArrMap-app*[*cat-comma-cs-simps*]:
  assumes $A = [[a, b, f]_\circ, [a', b', f']_\circ, [h, k]_\circ]_\circ$
    and $[[a, b, f]_\circ, [a', b', f']_\circ, [h, k]_\circ]_\circ :$
      $[a, b, f]_\circ \mapsto_{\mathfrak{F}\ {}_{CF}\downarrow\ c} [a', b', f']_\circ$
    and $g : c \mapsto_{\mathfrak{B}} c'$
  shows $\mathfrak{F}\ {}_{CF\downarrow_A}\ g(\!|ArrMap|\!)(\!|A|\!) =$
    $[[a, b, g \circ_{A\mathfrak{B}} f]_\circ, [a', b', g \circ_{A\mathfrak{B}} f']_\circ, [h, k]_\circ]_\circ$
⟨*proof*⟩

lemmas [*cat-comma-cs-simps*] = *is-functor.cf-cf-arr-comma-ArrMap-app*

14.8.4 Comma functors are functors

lemma (**in** *is-functor*) *cf-arr-cf-comma-is-functor*:
  assumes $g : c \mapsto_{\mathfrak{B}} c'$
  shows $g\ {}_{A\downarrow_{CF}}\ \mathfrak{F} : c' \downarrow_{CF} \mathfrak{F} \mapsto\mapsto_{C\alpha} c \downarrow_{CF} \mathfrak{F}$
⟨*proof*⟩

lemma (in *is-functor*) *cf-cf-arr-comma-is-functor*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows $\mathfrak{F}_{CF\downarrow_A}\ g : \mathfrak{F}_{CF\downarrow}\ c \mapsto\mapsto_{C\alpha} \mathfrak{F}_{CF\downarrow}\ c'$
⟨*proof*⟩

lemma (in *is-functor*) *cf-arr-cf-comma-is-functor'*[*cat-comma-cs-intros*]:
assumes $g : c \mapsto_{\mathfrak{B}} c'$ **and** $\mathfrak{A}' = c' \downarrow_{CF} \mathfrak{F}$ **and** $\mathfrak{B}' = c \downarrow_{CF} \mathfrak{F}$
shows $g_{A\downarrow CF}\ \mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

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

lemma (in *is-functor*) *cf-cf-arr-comma-is-functor'*[*cat-comma-cs-intros*]:
assumes $g : c \mapsto_{\mathfrak{B}} c'$ **and** $\mathfrak{A}' = \mathfrak{F}_{CF\downarrow}\ c$ **and** $\mathfrak{B}' = \mathfrak{F}_{CF\downarrow}\ c'$
shows $\mathfrak{F}_{CF\downarrow_A}\ g : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

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

lemma (in *is-functor*) *cf-arr-cf-comma-CId*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $(\mathfrak{B}(\!|CId|\!)(\!|b|\!))_{A\downarrow CF}\ \mathfrak{F} = cf\text{-}id\ (b \downarrow_{CF} \mathfrak{F})$
⟨*proof*⟩

lemma (in *is-functor*) *cf-cf-arr-comma-CId*:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathfrak{F}_{CF\downarrow_A}\ (\mathfrak{B}(\!|CId|\!)(\!|b|\!)) = cf\text{-}id\ (\mathfrak{F}_{CF\downarrow}\ b)$
⟨*proof*⟩

14.8.5 Comma functors and projections

lemma (in *is-functor*)
cf-cf-comp-cf-obj-cf-comma-proj-cf-arr-cf-comma[*cat-comma-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows $a_{O\sqcap_{CF}\ \mathfrak{F}} \circ_{CF} f_{A\downarrow CF}\ \mathfrak{F} = b_{O\sqcap_{CF}\ \mathfrak{F}}$
⟨*proof*⟩

lemma (in *is-functor*)
cf-cf-comp-cf-cf-obj-comma-proj-cf-cf-arr-comma[*cat-comma-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{F}_{CF\sqcap_O\ b} \circ_{CF} \mathfrak{F}_{CF\downarrow_A}\ f = \mathfrak{F}_{CF\sqcap_O\ a}$
⟨*proof*⟩

14.8.6 Opposite comma functors

lemma (in *is-functor*) *cf-op-cf-obj-comma-cf-arr-cf-comma*:
assumes $g : c \mapsto_{\mathfrak{B}} c'$
shows
op-cf-obj-comma $\mathfrak{F}\ c' \circ_{CF}$ *op-cf* $(\mathfrak{F}_{CF\downarrow_A}\ g) =$
$g_{A\downarrow CF}$ (*op-cf* $\mathfrak{F}$) $\circ_{CF}$ *op-cf-obj-comma* $\mathfrak{F}\ c$
⟨*proof*⟩

15 *Rel*

15.1 Background

The methodology chosen for the exposition of *Rel* as a category is analogous to the one used in [8] for the exposition of *Rel* as a semicategory. The general references for this section are Chapter I-7 in [7] and nLab [1][8].

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

lemmas (**in** *arr-Rel*) [*cat-Rel-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (**in** *arr-Rel*) [*cat-cs-intros*, *cat-Rel-cs-intros*] =
arr-Rel-axioms'

lemmas [*cat-Rel-cs-simps*] =
dg-Rel-shared-cs-simps
arr-Rel.arr-Rel-length
arr-Rel-comp-Rel-id-Rel-left
arr-Rel-comp-Rel-id-Rel-right
arr-Rel.arr-Rel-converse-Rel-converse-Rel
arr-Rel-converse-Rel-eq-iff
arr-Rel-converse-Rel-comp-Rel
arr-Rel-comp-Rel-converse-Rel-left-if-v11
arr-Rel-comp-Rel-converse-Rel-right-if-v11

lemmas [*cat-Rel-cs-intros*] =
dg-Rel-shared-cs-intros
arr-Rel-comp-Rel
arr-Rel.arr-Rel-converse-Rel

lemmas [*cat-cs-simps*] = *incl-Rel-ArrVal-app*

15.2 *Rel* as a category

15.2.1 Definition and elementary properties

definition *cat-Rel* :: $V \Rightarrow V$
where *cat-Rel* α =
[
Vset α,
set $\{T.\ \textit{arr-Rel}\ \alpha\ T\}$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$,
$(\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Rel}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$,
VLambda (*Vset* α) *id-Rel*
]$_{\circ}$

Components.

lemma *cat-Rel-components*:
shows *cat-Rel* $\alpha(\!|\textit{Obj}|\!)$ = *Vset* α
and *cat-Rel* $\alpha(\!|\textit{Arr}|\!) = \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}$
and *cat-Rel* $\alpha(\!|\textit{Dom}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$
and *cat-Rel* $\alpha(\!|\textit{Cod}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$
and *cat-Rel* $\alpha(\!|\textit{Comp}|\!) = (\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Rel}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$

[8] https://ncatlab.org/nlab/show/Rel

and *cat-Rel* $\alpha(\!|CId|\!)$ = *VLambda* (*Vset* α) *id-Rel*
⟨*proof*⟩

Slicing.

lemma *cat-smc-cat-Rel*: *cat-smc* (*cat-Rel* α) = *smc-Rel* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-cat-Rel*, *unfolded slicing-simps*]:
cat-Rel-Obj-iff = *smc-Rel-Obj-iff*
and *cat-Rel-Arr-iff*[*cat-Rel-cs-simps*] = *smc-Rel-Arr-iff*
and *cat-Rel-Dom-vsv*[*cat-Rel-cs-intros*] = *smc-Rel-Dom-vsv*
and *cat-Rel-Dom-vdomain*[*cat-Rel-cs-simps*] = *smc-Rel-Dom-vdomain*
and *cat-Rel-Dom-app*[*cat-Rel-cs-simps*] = *smc-Rel-Dom-app*
and *cat-Rel-Dom-vrange* = *smc-Rel-Dom-vrange*
and *cat-Rel-Cod-vsv*[*cat-Rel-cs-intros*] = *smc-Rel-Cod-vsv*
and *cat-Rel-Cod-vdomain*[*cat-Rel-cs-simps*] = *smc-Rel-Cod-vdomain*
and *cat-Rel-Cod-app*[*cat-Rel-cs-simps*] = *smc-Rel-Cod-app*
and *cat-Rel-Cod-vrange* = *smc-Rel-Cod-vrange*
and *cat-Rel-is-arrI*[*cat-Rel-cs-intros*] = *smc-Rel-is-arrI*
and *cat-Rel-is-arrD* = *smc-Rel-is-arrD*
and *cat-Rel-is-arrE* = *smc-Rel-is-arrE*
and *cat-Rel-is-arr-ArrValE* = *smc-Rel-is-arr-ArrValE*

lemmas-with [*folded cat-smc-cat-Rel*, *unfolded slicing-simps*, *unfolded cat-smc-cat-Rel*]:
cat-Rel-composable-arrs-dg-Rel = *smc-Rel-composable-arrs-dg-Rel*
and *cat-Rel-Comp* = *smc-Rel-Comp*
and *cat-Rel-Comp-app*[*cat-Rel-cs-simps*] = *smc-Rel-Comp-app*
and *cat-Rel-Comp-vdomain*[*simp*] = *smc-Rel-Comp-vdomain*
and *cat-Rel-is-monic-arrI* = *smc-Rel-is-monic-arrI*
and *cat-Rel-is-monic-arrD* = *smc-Rel-is-monic-arrD*
and *cat-Rel-is-monic-arr* = *smc-Rel-is-monic-arr*
and *cat-Rel-is-monic-arr-is-epic-arr* = *smc-Rel-is-monic-arr-is-epic-arr*
and *cat-Rel-is-epic-arr-is-monic-arr* = *smc-Rel-is-epic-arr-is-monic-arr*
and *cat-Rel-is-epic-arrI* = *smc-Rel-is-epic-arrI*
and *cat-Rel-is-epic-arrD* = *smc-Rel-is-epic-arrD*
and *cat-Rel-is-epic-arr* = *smc-Rel-is-epic-arr*

lemmas [*cat-cs-simps*] = *cat-Rel-is-arrD*(*2*,*3*)

lemmas [*cat-Rel-cs-intros*] = *cat-Rel-is-arrI*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-cat-Rel*, *unfolded slicing-simps*]:
cat-Rel-Hom-vifunion-in-Vset = *smc-Rel-Hom-vifunion-in-Vset*
and *cat-Rel-incl-Rel-is-arr* = *smc-Rel-incl-Rel-is-arr*
and *cat-Rel-incl-Rel-is-arr′*[*cat-Rel-cs-intros*] = *smc-Rel-incl-Rel-is-arr′*
and *cat-Rel-Comp-vrange* = *smc-Rel-Comp-vrange*
and *cat-Rel-obj-terminal* = *smc-Rel-obj-terminal*
and *cat-Rel-obj-initial* = *smc-Rel-obj-initial*
and *cat-Rel-obj-terminal-obj-initial* = *smc-Rel-obj-terminal-obj-initial*
and *cat-Rel-obj-null* = *smc-Rel-obj-null*
and *cat-Rel-is-zero-arr* = *smc-Rel-is-zero-arr*

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

15.2.2 Identity

lemma (in $\mathcal{Z}$**)** *cat-Rel-CId-app*[*cat-Rel-cs-simps*]:
assumes $T \in_{\circ}$ *Vset* α

shows *cat-Rel* $\alpha(\!|CId|\!)(\!|T|\!)$ = *id-Rel T*
⟨*proof*⟩

lemmas [*cat-Rel-cs-simps*] = $\mathcal{Z}$.*cat-Rel-CId-app*

15.2.3 *Rel* is a category

lemma (in $\mathcal{Z}$**)** *category-cat-Rel*: *category* α (*cat-Rel* α)
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *category-cat-Rel'*[*cat-Rel-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\alpha'' = \alpha$
shows *category* α' (*cat-Rel* α'')
⟨*proof*⟩

lemmas [*cat-Rel-cs-intros*] = $\mathcal{Z}$.*category-cat-Rel'*

15.3 Canonical dagger for *Rel*

15.3.1 Definition and elementary properties

definition *cf-dag-Rel* :: $V \Rightarrow V$ ($\langle\dagger_{C.Rel}\rangle$)
where $\dagger_{C.Rel}\ \alpha$ =
[
vid-on (*cat-Rel* $\alpha(\!|Obj|\!)$),
VLambda (*cat-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*,
op-cat (*cat-Rel* α),
cat-Rel α
]$_\circ$

Components.

lemma *cf-dag-Rel-components*:
shows $\dagger_{C.Rel}\ \alpha(\!|ObjMap|\!)$ = *vid-on* (*cat-Rel* $\alpha(\!|Obj|\!)$)
and $\dagger_{C.Rel}\ \alpha(\!|ArrMap|\!)$ = *VLambda* (*cat-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*
and $\dagger_{C.Rel}\ \alpha(\!|HomDom|\!)$ = *op-cat* (*cat-Rel* α)
and $\dagger_{C.Rel}\ \alpha(\!|HomCod|\!)$ = *cat-Rel* α
⟨*proof*⟩

Slicing.

lemma *cf-smcf-cf-dag-Rel*: *cf-smcf* ($\dagger_{C.Rel}\ \alpha$) = $\dagger_{SMC.Rel}\ \alpha$
⟨*proof*⟩

lemmas-with [*folded cat-smc-cat-Rel cf-smcf-cf-dag-Rel*, *unfolded slicing-simps*]:
cf-dag-Rel-ObjMap-vsv[*cat-Rel-cs-intros*] = *smcf-dag-Rel-ObjMap-vsv*
and *cf-dag-Rel-ObjMap-vdomain*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ObjMap-vdomain*
and *cf-dag-Rel-ObjMap-app*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ObjMap-app*
and *cf-dag-Rel-ObjMap-vrange*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ObjMap-vrange*
and *cf-dag-Rel-ArrMap-vsv*[*cat-Rel-cs-intros*] = *smcf-dag-Rel-ArrMap-vsv*
and *cf-dag-Rel-ArrMap-vdomain*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ArrMap-vdomain*
and *cf-dag-Rel-ArrMap-app*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ArrMap-app*
and *cf-dag-Rel-ArrMap-vrange*[*cat-Rel-cs-simps*] = *smcf-dag-Rel-ArrMap-vrange*
and *cf-dag-Rel-app-is-arr*[*cat-Rel-cs-intros*] = *smcf-dag-Rel-app-is-arr*
and *cf-dag-Rel-ArrMap-app-vdomain*[*cat-cs-simps*] =
smcf-dag-Rel-ArrMap-app-vdomain
and *cf-dag-Rel-ArrMap-app-vrange*[*cat-cs-simps*] =
smcf-dag-Rel-ArrMap-app-vrange
and *cf-dag-Rel-ArrMap-app-iff*[*cat-cs-simps*] = *smcf-dag-Rel-ArrMap-app-iff*
and *cf-dag-Rel-ArrMap-smc-Rel-Comp*[*cat-Rel-cs-simps*] =
smcf-dag-Rel-ArrMap-smc-Rel-Comp

15.3.2 Canonical dagger is a contravariant isomorphism of *Rel*

lemma (in $\mathcal{Z}$**)** *cf-dag-Rel-is-iso-functor*:
$\dagger_{C.Rel}\ \alpha : op\text{-}cat\ (cat\text{-}Rel\ \alpha) \mapsto\mapsto_{C.iso\alpha}\ cat\text{-}Rel\ \alpha$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cf-dag-Rel-is-iso-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{A}' = op\text{-}cat\ (cat\text{-}Rel\ \alpha)$
and $\mathfrak{B}' = cat\text{-}Rel\ \alpha$
and $\alpha' = \alpha$
shows $\dagger_{C.Rel}\ \alpha : \mathfrak{A}' \mapsto\mapsto_{C.iso\alpha'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*cf-dag-Rel-is-iso-functor'*

15.3.3 Further properties of the canonical dagger

lemma (in $\mathcal{Z}$**)** *cf-cn-comp-cf-dag-Rel-cf-dag-Rel*:
$\dagger_{C.Rel}\ \alpha\ _{CF}\circ\ \dagger_{C.Rel}\ \alpha = cf\text{-}id\ (cat\text{-}Rel\ \alpha)$
⟨*proof*⟩

15.4 Isomorphism

context
begin

private lemma *cat-Rel-is-iso-arr-right-vsubset*:
assumes $S : B \mapsto_{cat\text{-}Rel\ \alpha} A$
and $T : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $S \circ_{A\, cat\text{-}Rel\ \alpha} T = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|A|\!)$
and $T \circ_{A\, cat\text{-}Rel\ \alpha} S = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|B|\!)$
shows $S(\!|ArrVal|\!) \subseteq_{\circ} (T(\!|ArrVal|\!))^{-1}{}_{\circ}$
⟨*proof*⟩ **lemma** *cat-Rel-is-iso-arr-left-vsubset*:
assumes $S : B \mapsto_{cat\text{-}Rel\ \alpha} A$
and $T : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $S \circ_{A\, cat\text{-}Rel\ \alpha} T = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|A|\!)$
and $T \circ_{A\, cat\text{-}Rel\ \alpha} S = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|B|\!)$
shows $(T(\!|ArrVal|\!))^{-1}{}_{\circ} \subseteq_{\circ} S(\!|ArrVal|\!)$
⟨*proof*⟩ **lemma** *is-iso-arr-dag*:
assumes $S : B \mapsto_{cat\text{-}Rel\ \alpha} A$
and $T : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $S \circ_{A\, cat\text{-}Rel\ \alpha} T = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|A|\!)$
and $T \circ_{A\, cat\text{-}Rel\ \alpha} S = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|B|\!)$
shows $S = \dagger_{C.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!)$
⟨*proof*⟩

lemma *cat-Rel-is-iso-arrI*[*intro*]:
assumes $T : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $v11\ (T(\!|ArrVal|\!))$
and $\mathcal{D}_{\circ}\ (T(\!|ArrVal|\!)) = A$
and $\mathcal{R}_{\circ}\ (T(\!|ArrVal|\!)) = B$
shows $T : A \mapsto_{iso\, cat\text{-}Rel\ \alpha} B$
⟨*proof*⟩

lemma *cat-Rel-is-iso-arrD*[*dest*]:
assumes $T : A \mapsto_{iso\, cat\text{-}Rel\ \alpha} B$
shows $T : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $v11\ (T(\!|ArrVal|\!))$
and $\mathcal{D}_{\circ}\ (T(\!|ArrVal|\!)) = A$

and $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

end

lemmas [*cat-Rel-cs-simps*] = *cat-Rel-is-iso-arrD*(*3*,*4*)

lemma *cat-Rel-is-iso-arr*:
$T : A \mapsto_{iso} \textit{cat-Rel}\ \alpha\ B \longleftrightarrow$
$T : A \mapsto_{\textit{cat-Rel}\ \alpha} B \wedge$
v11 $(T(\!|ArrVal|\!)) \wedge$
$\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A \wedge$
$\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

15.5 The inverse arrow

lemma *cat-Rel-the-inverse*[*cat-Rel-cs-simps*]:
assumes $T : A \mapsto_{iso} \textit{cat-Rel}\ \alpha\ B$
shows ${T^{-1}}_{C\,\textit{cat-Rel}\ \alpha} = {T^{-1}}_{Rel}$
⟨*proof*⟩

16 *Par*

16.1 Background

The methodology chosen for the exposition of *Par* as a category is analogous to the one used in [8] for the exposition of *Par* as a semicategory.

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

lemmas (in *arr-Par*) [*cat-Par-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (in *arr-Par*) [*cat-cs-intros*, *cat-Par-cs-intros*] =
arr-Par-axioms'

lemmas [*cat-Par-cs-simps*] =
dg-Rel-shared-cs-simps
arr-Par.arr-Par-length
arr-Par-comp-Par-id-Par-left
arr-Par-comp-Par-id-Par-right

lemmas [*cat-Par-cs-intros*] =
arr-Par-comp-Par

16.2 *Par* as a category

16.2.1 Definition and elementary properties

definition *cat-Par* :: $V \Rightarrow V$
where *cat-Par* α =
[
Vset α,
set $\{T.\ \textit{arr-Par}\ \alpha\ T\}$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Par}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Par}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$,
$(\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Par}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$,
VLambda (*Vset* α) *id-Par*
]$_{\circ}$

Components.

lemma *cat-Par-components*:
shows *cat-Par* $\alpha(\!|\textit{Obj}|\!)$ = *Vset* α
and *cat-Par* $\alpha(\!|\textit{Arr}|\!)$ = *set* $\{T.\ \textit{arr-Par}\ \alpha\ T\}$
and *cat-Par* $\alpha(\!|\textit{Dom}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Par}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$
and *cat-Par* $\alpha(\!|\textit{Cod}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Par}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$
and *cat-Par* $\alpha(\!|\textit{Comp}|\!) = (\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Par}\ \alpha).\ ST(\!|0|\!) \circ_{Par} ST(\!|1_{\mathbb{N}}|\!))$
and *cat-Par* $\alpha(\!|\textit{CId}|\!)$ = *VLambda* (*Vset* α) *id-Par*
⟨*proof*⟩

Slicing.

lemma *cat-smc-cat-Par*: *cat-smc* (*cat-Par* α) = *smc-Par* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-cat-Par*, *unfolded slicing-simps*]:
cat-Par-Obj-iff = *smc-Par-Obj-iff*
and *cat-Par-Arr-iff*[*cat-Par-cs-simps*] = *smc-Par-Arr-iff*
and *cat-Par-Dom-vsv*[*cat-Par-cs-intros*] = *smc-Par-Dom-vsv*
and *cat-Par-Dom-vdomain*[*cat-Par-cs-simps*] = *smc-Par-Dom-vdomain*

and *cat-Par-Dom-vrange* = *smc-Par-Dom-vrange*
and *cat-Par-Dom-app*[*cat-Par-cs-simps*] = *smc-Par-Dom-app*
and *cat-Par-Cod-vsv*[*cat-Par-cs-intros*] = *smc-Par-Cod-vsv*
and *cat-Par-Cod-vdomain*[*cat-Par-cs-simps*] = *smc-Par-Cod-vdomain*
and *cat-Par-Cod-vrange* = *smc-Par-Cod-vrange*
and *cat-Par-Cod-app*[*cat-Par-cs-simps*] = *smc-Par-Cod-app*
and *cat-Par-is-arrI* = *smc-Par-is-arrI*
and *cat-Par-is-arrD* = *smc-Par-is-arrD*
and *cat-Par-is-arrE* = *smc-Par-is-arrE*

lemmas-with [*folded cat-smc-cat-Par*, *unfolded slicing-simps*]:
cat-Par-composable-arrs-dg-Par = *smc-Par-composable-arrs-dg-Par*
and *cat-Par-Comp* = *smc-Par-Comp*
and *cat-Par-Comp-app*[*cat-Par-cs-simps*] = *smc-Par-Comp-app*
and *cat-Par-Comp-vdomain*[*cat-Par-cs-simps*] = *smc-Par-Comp-vdomain*
and *cat-Par-is-monic-arrI* = *smc-Par-is-monic-arrI*
and *cat-Par-is-monic-arrD* = *smc-Par-is-monic-arrD*
and *cat-Par-is-monic-arr* = *smc-Par-is-monic-arr*
and *cat-Par-is-epic-arrI* = *smc-Par-is-epic-arrI*
and *cat-Par-is-epic-arrD* = *smc-Par-is-epic-arrD*
and *cat-Par-is-epic-arr* = *smc-Par-is-epic-arr*

lemmas [*cat-cs-simps*] = *cat-Par-is-arrD*(*2*,*3*)

lemmas [*cat-Par-cs-intros*] = *cat-Par-is-arrI*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-cat-Par*, *unfolded slicing-simps*]:
cat-Par-Hom-vifunion-in-Vset = *smc-Par-Hom-vifunion-in-Vset*
and *cat-Par-incl-Par-is-arr* = *smc-Par-incl-Par-is-arr*
and *cat-Par-incl-Par-is-arr'*[*cat-Par-cs-intros*] = *smc-Par-incl-Par-is-arr'*
and *cat-Par-Comp-vrange* = *smc-Par-Comp-vrange*
and *cat-Par-obj-terminal* = *smc-Par-obj-terminal*
and *cat-Par-obj-initial* = *smc-Par-obj-initial*
and *cat-Par-obj-terminal-obj-initial* = *smc-Par-obj-terminal-obj-initial*
and *cat-Par-obj-null* = *smc-Par-obj-null*
and *cat-Par-is-zero-arr* = *smc-Par-is-zero-arr*

lemmas [*cat-Par-cs-intros*] = $\mathcal{Z}$.*cat-Par-incl-Par-is-arr'*

16.2.2 Identity

lemma *cat-Par-CId-app*[*cat-Par-cs-simps*]:
assumes $A \in_{\circ} Vset\ \alpha$
shows *cat-Par* $\alpha(\!|CId|\!)(\!|A|\!) = id\text{-}Par\ A$
⟨*proof*⟩

lemma *id-Par-CId-app-app*[*cat-cs-simps*]:
assumes $A \in_{\circ} Vset\ \alpha$ **and** $a \in_{\circ} A$
shows *cat-Par* $\alpha(\!|CId|\!)(\!|A|\!)(\!|ArrVal|\!)(\!|a|\!) = a$
⟨*proof*⟩

16.2.3 *Par* is a category

lemma (in $\mathcal{Z}$**)** *category-cat-Par*: *category* α (*cat-Par* α)
⟨*proof*⟩

16.2.4 *Par* is a wide replete subcategory of *Rel*

lemma (in $\mathcal{Z}$**)** *wide-replete-subcategory-cat-Par-cat-Rel*:

cat-Par $\alpha \subseteq_{C.wr\alpha}$ *cat-Rel* α
⟨*proof*⟩

16.3 Isomorphism

lemma *cat-Par-is-iso-arrI*[*intro*]:
assumes $T : A \mapsto_{cat\text{-}Par\ \alpha} B$
and *v11* $(T(\!|ArrVal|\!))$
and $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
and $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
shows $T : A \mapsto_{iso\, cat\text{-}Par\ \alpha} B$
⟨*proof*⟩

lemma *cat-Par-is-iso-arrD*[*dest*]:
assumes $T : A \mapsto_{iso\, cat\text{-}Par\ \alpha} B$
shows $T : A \mapsto_{cat\text{-}Par\ \alpha} B$
and *v11* $(T(\!|ArrVal|\!))$
and $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
and $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

lemma *cat-Par-is-iso-arr*:
$T : A \mapsto_{iso\, cat\text{-}Par\ \alpha} B \longleftrightarrow$
$T : A \mapsto_{cat\text{-}Par\ \alpha} B \wedge$
v11 $(T(\!|ArrVal|\!)) \wedge$
$\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A \wedge$
$\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

16.4 The inverse arrow

abbreviation (*input*) *converse-Par* :: $V \Rightarrow V$ $(\langle(\text{-}^{-1}{}_{Par})\rangle\ [1000]\ 999)$
where $a^{-1}{}_{Par} \equiv a^{-1}{}_{Rel}$

lemma *cat-Par-the-inverse*[*cat-Par-cs-simps*]:
assumes $T : A \mapsto_{iso\, cat\text{-}Par\ \alpha} B$
shows $T^{-1}{}_{C\, cat\text{-}Par\ \alpha} = T^{-1}{}_{Par}$
⟨*proof*⟩

17 *Set*

17.1 Background

The methodology chosen for the exposition of *Set* as a category is analogous to the one used in [8] for the exposition of *Set* as a semicategory.

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

lemmas (**in** *arr-Set*) [*cat-Set-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (**in** *arr-Set*) [*cat-cs-intros*, *cat-Set-cs-intros*] =
arr-Set-axioms'

lemmas [*cat-Set-cs-simps*] =
dg-Rel-shared-cs-simps
arr-Set.arr-Set-ArrVal-vdomain
arr-Set-comp-Set-id-Set-left
arr-Set-comp-Set-id-Set-right

lemmas [*cat-Set-cs-intros*] =
dg-Rel-shared-cs-intros
arr-Set-comp-Set

named-theorems *cat-rel-par-Set-cs-intros*
named-theorems *cat-rel-par-Set-cs-simps*
named-theorems *cat-rel-Par-set-cs-intros*
named-theorems *cat-rel-Par-set-cs-simps*
named-theorems *cat-Rel-par-set-cs-intros*
named-theorems *cat-Rel-par-set-cs-simps*

17.2 *Set* as a category

17.2.1 Definition and elementary properties

definition *cat-Set* :: $V \Rightarrow V$
where *cat-Set* α =
$[$
Vset α,
set $\{T.\ \textit{arr-Set}\ \alpha\ T\}$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$,
$(\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Set}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$,
VLambda (*Vset* α) *id-Set*
$]_{\circ}$

Components.

lemma *cat-Set-components*:
shows *cat-Set* $\alpha(\!|\textit{Obj}|\!)$ = *Vset* α
and *cat-Set* $\alpha(\!|\textit{Arr}|\!)$ = *set* $\{T.\ \textit{arr-Set}\ \alpha\ T\}$
and *cat-Set* $\alpha(\!|\textit{Dom}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$
and *cat-Set* $\alpha(\!|\textit{Cod}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$
and *cat-Set* $\alpha(\!|\textit{Comp}|\!)$ =
$(\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Set}\ \alpha).\ ST(\!|0|\!) \circ_{Par} ST(\!|1_{\mathbb{N}}|\!))$
and *cat-Set* $\alpha(\!|\textit{CId}|\!)$ = *VLambda* (*Vset* α) *id-Set*
⟨*proof*⟩

Slicing.

lemma *cat-smc-cat-Set*: *cat-smc* (*cat-Set* α) = *smc-Set* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-cat-Set*, *unfolded slicing-simps*]:
cat-Set-Obj-iff = *smc-Set-Obj-iff*
and *cat-Set-Arr-iff*[*cat-Set-cs-simps*] = *smc-Set-Arr-iff*
and *cat-Set-Dom-vsv*[*intro*] = *smc-Set-Dom-vsv*
and *cat-Set-Dom-vdomain*[*simp*] = *smc-Set-Dom-vdomain*
and *cat-Set-Dom-vrange* = *smc-Set-Dom-vrange*
and *cat-Set-Dom-app* = *smc-Set-Dom-app*
and *cat-Set-Cod-vsv*[*intro*] = *smc-Set-Cod-vsv*
and *cat-Set-Cod-vdomain*[*simp*] = *smc-Set-Cod-vdomain*
and *cat-Set-Cod-vrange* = *smc-Set-Cod-vrange*
and *cat-Set-Cod-app*[*cat-Set-cs-simps*] = *smc-Set-Cod-app*
and *cat-Set-is-arrI* = *smc-Set-is-arrI*
and *cat-Set-is-arrD* = *smc-Set-is-arrD*
and *cat-Set-is-arrE* = *smc-Set-is-arrE*
and *cat-Set-ArrVal-vdomain*[*cat-cs-simps*] = *smc-Set-ArrVal-vdomain*
and *cat-Set-ArrVal-app-vrange*[*cat-Set-cs-intros*] = *smc-Set-ArrVal-app-vrange*

lemmas [*cat-cs-simps*] = *cat-Set-is-arrD*(*2*,*3*)

lemmas [*cat-Set-cs-intros*] =
cat-Set-is-arrI

lemmas-with [*folded cat-smc-cat-Set*, *unfolded slicing-simps*]:
cat-Set-composable-arrs-dg-Set = *smc-Set-composable-arrs-dg-Set*
and *cat-Set-Comp* = *smc-Set-Comp*
and *cat-Set-Comp-app*[*cat-Set-cs-simps*] = *smc-Set-Comp-app*
and *cat-Set-Comp-vdomain*[*cat-Set-cs-simps*] = *smc-Set-Comp-vdomain*
and *cat-Set-is-monic-arrI* = *smc-Set-is-monic-arrI*
and *cat-Set-is-monic-arrD* = *smc-Set-is-monic-arrD*
and *cat-Set-is-monic-arr* = *smc-Set-is-monic-arr*
and *cat-Set-is-epic-arrI* = *smc-Set-is-epic-arrI*
and *cat-Set-is-epic-arrD* = *smc-Set-is-epic-arrD*
and *cat-Set-is-epic-arr* = *smc-Set-is-epic-arr*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-cat-Set*, *unfolded slicing-simps*]:
cat-Set-Hom-vifunion-in-Vset = *smc-Set-Hom-vifunion-in-Vset*
and *cat-Set-incl-Set-is-arr* = *smc-Set-incl-Set-is-arr*
and *cat-Set-Comp-ArrVal* = *smc-Set-Comp-ArrVal*
and *cat-Set-Comp-vrange* = *smc-Set-Comp-vrange*
and *cat-Set-obj-terminal* = *smc-Set-obj-terminal*
and *cat-Set-obj-initial* = *smc-Set-obj-initial*
and *cat-Set-obj-null* = *smc-Set-obj-null*
and *cat-Set-is-zero-arr* = *smc-Set-is-zero-arr*

lemmas [*cat-cs-simps*] =
$\mathcal{Z}$.*cat-Set-Comp-ArrVal*

lemma (in $\mathcal{Z}$**)** *cat-Set-incl-Set-is-arr'*[*cat-cs-intros*, *cat-Set-cs-intros*]:
assumes $A \in_{\circ}$ *cat-Set* α(|*Obj*|)
and $B \in_{\circ}$ *cat-Set* α(|*Obj*|)
and $A \subseteq_{\circ} B$
and $A' = A$
and $B' = B$
and $\mathfrak{C}' =$ *cat-Set* α

shows *incl-Set A B* : $A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-Set-cs-intros*] = $\mathcal{Z}$.*cat-Set-incl-Set-is-arr'*

17.2.2 Identity

lemma *cat-Set-CId-app*[*cat-Set-cs-simps*]:
assumes $A \in_\circ$ *Vset* α
shows *cat-Set* $\alpha(\lfloor CId \rfloor)(\lfloor A \rfloor)$ = *id-Set A*
⟨*proof*⟩

lemma *cat-Set-CId-app-app*[*cat-cs-simps*]:
assumes $A \in_\circ$ *cat-Set* $\alpha(\lfloor Obj \rfloor)$ **and** $a \in_\circ A$
shows *cat-Set* $\alpha(\lfloor CId \rfloor)(\lfloor A \rfloor)(\lfloor ArrVal \rfloor)(\lfloor a \rfloor) = a$
⟨*proof*⟩

17.2.3 *Set* is a category

lemma (**in** $\mathcal{Z}$) *category-cat-Set*: *category* α (*cat-Set* α)
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *category-cat-Set'*:
assumes $\beta = \alpha$
shows *category* β (*cat-Set* α)
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*category-cat-Set'*

17.2.4 *Set* is a wide replete subcategory of *Par*

lemma (**in** $\mathcal{Z}$) *wide-replete-subcategory-cat-Set-cat-Par*:
cat-Set $\alpha \subseteq_{C.wr\alpha}$ *cat-Par* α
⟨*proof*⟩

17.2.5 *Set* is a subcategory of *Set*

lemma (**in** $\mathcal{Z}$) *subcategory-cat-Set-cat-Set*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *cat-Set* $\alpha \subseteq_{C\beta}$ *cat-Set* β
⟨*proof*⟩

17.2.6 Further properties

lemma *cat-Set-Comp-ArrVal-vrange*:
assumes $S : B \mapsto_{cat\text{-}Set\ \alpha} C$ **and** $T : A \mapsto_{cat\text{-}Set\ \alpha} B$
shows $\mathcal{R}_\circ\ ((S \circ_{A\ cat\text{-}Set\ \alpha} T)(\lfloor ArrVal \rfloor)) \subseteq_\circ \mathcal{R}_\circ\ (S(\lfloor ArrVal \rfloor))$
⟨*proof*⟩

17.3 Isomorphism

lemma *cat-Set-is-iso-arrI*[*intro*]:
— See [1][9]).
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$
and *v11* $(T(\lfloor ArrVal \rfloor))$
and $\mathcal{D}_\circ\ (T(\lfloor ArrVal \rfloor)) = A$
and $\mathcal{R}_\circ\ (T(\lfloor ArrVal \rfloor)) = B$

[9] https://ncatlab.org/nlab/show/isomorphism

shows $T : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$
⟨*proof*⟩

lemma *cat-Set-is-iso-arrD*[*dest*]:
assumes $T : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$
shows $T : A \mapsto_{\textit{cat-Set}\ \alpha} B$
and $\textit{v11}\ (T(\!|\textit{ArrVal}|\!))$
and $\mathcal{D}_{\circ}\ (T(\!|\textit{ArrVal}|\!)) = A$
and $\mathcal{R}_{\circ}\ (T(\!|\textit{ArrVal}|\!)) = B$
⟨*proof*⟩

lemma *cat-Set-is-iso-arr*:
$T : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B \longleftrightarrow$
$T : A \mapsto_{\textit{cat-Set}\ \alpha} B \wedge$
$\textit{v11}\ (T(\!|\textit{ArrVal}|\!)) \wedge$
$\mathcal{D}_{\circ}\ (T(\!|\textit{ArrVal}|\!)) = A \wedge$
$\mathcal{R}_{\circ}\ (T(\!|\textit{ArrVal}|\!)) = B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cat-Set-is-iso-arr-if-monic-and-epic*:
assumes $F : A \mapsto_{mon} \textit{cat-Set}\ \alpha\ B$ **and** $F : A \mapsto_{epi} \textit{cat-Set}\ \alpha\ B$
shows $F : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$
⟨*proof*⟩

17.4 The inverse arrow

lemma *cat-Set-ArrVal-app-is-arr*[*cat-cs-intros*]:
assumes $f : a \mapsto_{\mathfrak{A}} b$
and $\textit{category}\ \alpha\ \mathfrak{A}$
and $F : \textit{Hom}\ \mathfrak{A}\ a\ b \mapsto_{\textit{cat-Set}\ \alpha} \textit{Hom}\ \mathfrak{B}\ c\ d$
shows $F(\!|\textit{ArrVal}|\!)(\!|f|\!) : c \mapsto_{\mathfrak{B}} d$
⟨*proof*⟩

abbreviation (*input*) *converse-Set* :: $V \Rightarrow V$ $(\langle(\text{-}^{-1}{}_{Set})\rangle\ [1000]\ 999)$
where ${a^{-1}}_{Set} \equiv {a^{-1}}_{Rel}$

lemma *cat-Set-the-inverse*[*cat-Set-cs-simps*]:
assumes $T : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$
shows ${T^{-1}}_{\mathbin{C}\ \textit{cat-Set}\ \alpha} = {T^{-1}}_{Set}$
⟨*proof*⟩

lemma *cat-Set-the-inverse-app*[*cat-cs-intros*]:
assumes $T : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$
and $a \in_{\circ} A$
and [*cat-cs-simps*]: $T(\!|\textit{ArrVal}|\!)(\!|a|\!) = b$
shows $({T^{-1}}_{\mathbin{C}\ \textit{cat-Set}\ \alpha})(\!|\textit{ArrVal}|\!)(\!|b|\!) = a$
⟨*proof*⟩

lemma *cat-Set-ArrVal-app-the-inverse-is-arr*[*cat-cs-intros*]:
assumes $f : c \mapsto_{\mathfrak{B}} d$
and $\textit{category}\ \alpha\ \mathfrak{B}$
and $F : \textit{Hom}\ \mathfrak{A}\ a\ b \mapsto_{iso} \textit{cat-Set}\ \alpha\ \textit{Hom}\ \mathfrak{B}\ c\ d$
shows ${F^{-1}}_{\mathbin{C}\ \textit{cat-Set}\ \alpha}(\!|\textit{ArrVal}|\!)(\!|f|\!) : a \mapsto_{\mathfrak{A}} b$
⟨*proof*⟩

lemma *cat-Set-app-the-inverse-app*[*cat-cs-simps*]:
assumes $F : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ B$ **and** $b \in_{\circ} B$
shows $F(\!|\textit{ArrVal}|\!)(\!|{F^{-1}}_{\mathbin{C}\ \textit{cat-Set}\ \alpha}(\!|\textit{ArrVal}|\!)(\!|b|\!)|\!) = b$

⟨*proof*⟩

lemma *cat-Set-the-inverse-app-app*[*cat-cs-simps*]:
assumes $F : A \mapsto_{iso} cat\text{-}Set\ \alpha\ B$ **and** $a \in_\circ A$
shows $F^{-1}{}_{C\ cat\text{-}Set\ \alpha}(\lfloor ArrVal \rfloor)(\lfloor F(\lfloor ArrVal \rfloor)(\lfloor a \rfloor)\rfloor) = a$
⟨*proof*⟩

17.5 Conversion of a single-valued relation to an arrow in *Set*

17.5.1 Definition and elementary properties

definition *cat-Set-arr-of-vsv* :: $V \Rightarrow V \Rightarrow V$
where $cat\text{-}Set\text{-}arr\text{-}of\text{-}vsv\ f\ B = [f, \mathcal{D}_\circ\ f, B]_\circ$

Components.

lemma *cat-Set-arr-of-vsv-components*:
shows [*cat-Set-cs-simps*]: $cat\text{-}Set\text{-}arr\text{-}of\text{-}vsv\ f\ B(\lfloor ArrVal \rfloor) = f$
and [*cat-Set-cs-simps*]: $cat\text{-}Set\text{-}arr\text{-}of\text{-}vsv\ f\ B(\lfloor ArrDom \rfloor) = \mathcal{D}_\circ\ f$
and [*cat-cs-simps*, *cat-Set-cs-simps*]: $cat\text{-}Set\text{-}arr\text{-}of\text{-}vsv\ f\ B(\lfloor ArrCod \rfloor) = B$
⟨*proof*⟩

17.5.2 Conversion of a single-valued relation to an arrow in *Set* is an arrow in *Set*

lemma (in $\mathcal{Z}$**)** *cat-Set-arr-of-vsv-is-arr*:
assumes *vsv r*
and $\mathcal{R}_\circ\ r \subseteq_\circ B$
and $\mathcal{D}_\circ\ r \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj \rfloor)$
and $B \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj \rfloor)$
shows $cat\text{-}Set\text{-}arr\text{-}of\text{-}vsv\ r\ B : \mathcal{D}_\circ\ r \mapsto_{cat\text{-}Set\ \alpha} B$
⟨*proof*⟩

17.6 Left restriction for *Set*

17.6.1 Definition and elementary properties

definition *vlrestriction-Set* :: $V \Rightarrow V \Rightarrow V$ (**infixr** $\langle \restriction^l{}_{Set} \rangle$ *80*)
where $T \restriction^l{}_{Set}\ C = [T(\lfloor ArrVal \rfloor) \restriction^l{}_\circ\ C, C, T(\lfloor ArrCod \rfloor)]_\circ$

Components.

lemma *vlrestriction-Set-components*:
shows [*cat-Set-cs-simps*]: $(T \restriction^l{}_{Set}\ C)(\lfloor ArrVal \rfloor) = T(\lfloor ArrVal \rfloor) \restriction^l{}_\circ\ C$
and [*cat-cs-simps*, *cat-Set-cs-simps*]: $(T \restriction^l{}_{Set}\ C)(\lfloor ArrDom \rfloor) = C$
and [*cat-cs-simps*, *cat-Set-cs-simps*]: $(T \restriction^l{}_{Set}\ C)(\lfloor ArrCod \rfloor) = T(\lfloor ArrCod \rfloor)$
⟨*proof*⟩

17.6.2 Arrow value

lemma *vlrestriction-Set-ArrVal-vdomain*[*cat-cs-simps*]:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $C \subseteq_\circ A$
shows $\mathcal{D}_\circ\ ((T \restriction^l{}_{Set}\ C)(\lfloor ArrVal \rfloor)) = C$
⟨*proof*⟩

lemma *vlrestriction-Set-ArrVal-app*[*cat-cs-simps*]:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $C \subseteq_\circ A$ **and** $x \in_\circ C$
shows $(T \restriction^l{}_{Set}\ C)(\lfloor ArrVal \rfloor)(\lfloor x \rfloor) = T(\lfloor ArrVal \rfloor)(\lfloor x \rfloor)$
⟨*proof*⟩

17.6.3 Left restriction for *Set* is an arrow in *Set*

lemma *vlrestriction-Set-is-arr*:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $C \subseteq_\circ A$
shows $T \restriction^l_{Set}\ C : C \mapsto_{cat\text{-}Set\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vlrestriction-Set-is-monic-arr*:
assumes $T : A \mapsto_{mon cat\text{-}Set\ \alpha} B$ **and** $C \subseteq_\circ A$
shows $T \restriction^l_{Set}\ C : C \mapsto_{mon cat\text{-}Set\ \alpha} B$
⟨*proof*⟩

17.7 Right restriction for *Set*

17.7.1 Definition and elementary properties

definition *vrrestriction-Set* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\restriction^r_{Set}$› *80*)
where $T \restriction^r_{Set}\ C = [T(\!|ArrVal|\!) \restriction^r_\circ C,\ T(\!|ArrDom|\!),\ C]_\circ$

Components.

lemma *vrrestriction-Set-components*:
shows [*cat-Set-cs-simps*]: $(T \restriction^r_{Set}\ C)(\!|ArrVal|\!) = T(\!|ArrVal|\!) \restriction^r_\circ C$
and [*cat-cs-simps*, *cat-Set-cs-simps*]: $(T \restriction^r_{Set}\ C)(\!|ArrDom|\!) = T(\!|ArrDom|\!)$
and [*cat-cs-simps*, *cat-Set-cs-simps*]: $(T \restriction^r_{Set}\ C)(\!|ArrCod|\!) = C$
⟨*proof*⟩

17.7.2 Arrow value

lemma *vrrestriction-Set-ArrVal-app*[*cat-cs-simps*]:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) \subseteq_\circ C$
shows $(T \restriction^r_{Set}\ C)(\!|ArrVal|\!) = T(\!|ArrVal|\!)$
⟨*proof*⟩

17.7.3 Right restriction for *Set* is an arrow in *Set*

lemma *vrrestriction-Set-is-arr*:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$
and $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) \subseteq_\circ C$
and $C \in_\circ cat\text{-}Set\ \alpha(\!|Obj|\!)$
shows $T \restriction^r_{Set}\ C : A \mapsto_{cat\text{-}Set\ \alpha} C$
⟨*proof*⟩

lemma *vrrestriction-Set-is-arr'*[*cat-cs-intros*]:
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$
and $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) \subseteq_\circ C$
and $C \in_\circ cat\text{-}Set\ \alpha(\!|Obj|\!)$
and $C' = C$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $T \restriction^r_{Set}\ C : A \mapsto_{\mathfrak{C}'} C'$
⟨*proof*⟩

17.7.4 Further properties

lemma
assumes $T : A \mapsto_{cat\text{-}Set\ \alpha} B$
shows *vrrestriction-Set-vrange-is-arr*:
$T \restriction^r_{Set}\ \mathcal{R}_\circ\ (T(\!|ArrVal|\!)) : A \mapsto_{cat\text{-}Set\ \alpha} \mathcal{R}_\circ\ (T(\!|ArrVal|\!))$
and *vrrestriction-Set-vrange-ArrVal-app*[*cat-cs-simps*, *cat-Set-cs-simps*]:
$(T \restriction^r_{Set}\ \mathcal{R}_\circ\ (T(\!|ArrVal|\!)))(\!|ArrVal|\!) = T(\!|ArrVal|\!)$

⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vrrestriction-Set-vrange-is-iso-arr*:
assumes $T : A \mapsto_{mon} \textit{cat-Set}\ \alpha\ B$
shows $T \restriction^r_{Set}\ \mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!)) : A \mapsto_{iso} \textit{cat-Set}\ \alpha\ \mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!))$
⟨*proof*⟩

17.7.5 Connections

lemma *cat-Set-Comp-vrrestriction-Set*:
assumes $S : B \mapsto_{\textit{cat-Set}\ \alpha}\ C$
and $T : A \mapsto_{\textit{cat-Set}\ \alpha}\ B$
and $\mathcal{R}_\circ\ (S(\!|\textit{ArrVal}|\!)) \subseteq_\circ D$
and $D \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
shows $S \restriction^r_{Set}\ D \circ_{A\,\textit{cat-Set}\ \alpha}\ T = (S \circ_{A\,\textit{cat-Set}\ \alpha}\ T) \restriction^r_{Set}\ D$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cat-Set-CId-vrrestriction-Set*[*cat-cs-simps*]:
assumes $A \subseteq_\circ B$ **and** $B \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{cat-Set}\ \alpha(\!|\textit{CId}|\!)(\!|A|\!) \restriction^r_{Set}\ B = \textit{incl-Set}\ A\ B$
⟨*proof*⟩

lemma *cat-Set-Comp-incl-Rel-vrrestriction-Set*[*cat-cs-simps*]:
assumes $F : A \mapsto_{\textit{cat-Set}\ \alpha}\ B$ **and** $C \subseteq_\circ B$ **and** $\mathcal{R}_\circ\ (F(\!|\textit{ArrVal}|\!)) \subseteq_\circ C$
shows $\textit{incl-Rel}\ C\ B \circ_{A\,\textit{cat-Set}\ \alpha}\ F \restriction^r_{Set}\ C = F$
⟨*proof*⟩

17.8 Projection arrows for *vtimes*

17.8.1 Definition and elementary properties

definition $\textit{vfst-arrow} :: V \Rightarrow V \Rightarrow V$
where $\textit{vfst-arrow}\ A\ B = [(\lambda ab{\in_\circ} A \times_\circ B.\ \textit{vfst}\ ab),\ A \times_\circ B,\ A]_\circ$

definition $\textit{vsnd-arrow} :: V \Rightarrow V \Rightarrow V$
where $\textit{vsnd-arrow}\ A\ B = [(\lambda ab{\in_\circ} A \times_\circ B.\ \textit{vsnd}\ ab),\ A \times_\circ B,\ B]_\circ$

Components.

lemma *vfst-arrow-components*:
shows $\textit{vfst-arrow}\ A\ B(\!|\textit{ArrVal}|\!) = (\lambda ab{\in_\circ} A \times_\circ B.\ \textit{vfst}\ ab)$
and [*cat-cs-simps*]: $\textit{vfst-arrow}\ A\ B(\!|\textit{ArrDom}|\!) = A \times_\circ B$
and [*cat-cs-simps*]: $\textit{vfst-arrow}\ A\ B(\!|\textit{ArrCod}|\!) = A$
⟨*proof*⟩

lemma *vsnd-arrow-components*:
shows $\textit{vsnd-arrow}\ A\ B(\!|\textit{ArrVal}|\!) = (\lambda ab{\in_\circ} A \times_\circ B.\ \textit{vsnd}\ ab)$
and [*cat-cs-simps*]: $\textit{vsnd-arrow}\ A\ B(\!|\textit{ArrDom}|\!) = A \times_\circ B$
and [*cat-cs-simps*]: $\textit{vsnd-arrow}\ A\ B(\!|\textit{ArrCod}|\!) = B$
⟨*proof*⟩

17.8.2 Arrow value

mk-VLambda *vfst-arrow-components*(*1*)
|*vsv vfst-arrow-ArrVal-vsv*[*cat-cs-intros*]|
|*vdomain vfst-arrow-ArrVal-vdomain*[*cat-cs-simps*]|
|*app vfst-arrow-ArrVal-app'*|

mk-VLambda *vsnd-arrow-components*(*1*)
|*vsv vsnd-arrow-ArrVal-vsv*[*cat-cs-intros*]|

|*vdomain vsnd-arrow-ArrVal-vdomain*[*cat-cs-simps*]|
|*app vsnd-arrow-ArrVal-app'*|

lemma *vfst-arrow-ArrVal-app*[*cat-cs-simps*]:
assumes $ab = \langle a, b \rangle$ **and** $ab \in_\circ A \times_\circ B$
shows *vfst-arrow A B*(|*ArrVal*|)(|*ab*|) = *a*
⟨*proof*⟩

lemma *vfst-arrow-vrange*: $\mathcal{R}_\circ$ (*vfst-arrow A B*(|*ArrVal*|)) $\subseteq_\circ A$
⟨*proof*⟩

lemma *vsnd-arrow-ArrVal-app*[*cat-cs-simps*]:
assumes $ab = \langle a, b \rangle$ **and** $ab \in_\circ A \times_\circ B$
shows *vsnd-arrow A B*(|*ArrVal*|)(|*ab*|) = *b*
⟨*proof*⟩

lemma *vsnd-arrow-vrange*: $\mathcal{R}_\circ$ (*vsnd-arrow A B*(|*ArrVal*|)) $\subseteq_\circ B$
⟨*proof*⟩

17.8.3 Projection arrows are arrows in the category *Set*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-arr-Vset*:
assumes $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α
shows *vfst-arrow A B* : $A \times_\circ B \mapsto_{\textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-arr*:
assumes $A \in_\circ$ *cat-Set* α(|*Obj*|) **and** $B \in_\circ$ *cat-Set* α(|*Obj*|)
shows *vfst-arrow A B* : $A \times_\circ B \mapsto_{\textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_\circ$ *cat-Set* α(|*Obj*|)
and $B \in_\circ$ *cat-Set* α(|*Obj*|)
and $AB = A \times_\circ B$
and $A' = A$
and $\mathfrak{C}' =$ *cat-Set* α
shows *vfst-arrow A B* : $AB \mapsto_{\mathfrak{C}'} A'$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Set-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-arr-Vset*:
assumes $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α
shows *vsnd-arrow A B* : $A \times_\circ B \mapsto_{\textit{cat-Set}\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-arr*:
assumes $A \in_\circ$ *cat-Set* α(|*Obj*|) **and** $B \in_\circ$ *cat-Set* α(|*Obj*|)
shows *vsnd-arrow A B* : $A \times_\circ B \mapsto_{\textit{cat-Set}\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_\circ$ *cat-Set* α(|*Obj*|)
and $B \in_\circ$ *cat-Set* α(|*Obj*|)
and $AB = A \times_\circ B$
and $B' = B$
and $\mathfrak{C}' =$ *cat-Set* α

shows *vsnd-arrow A B* : $AB \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Set-arr'*

17.8.4 Projection arrows are arrows in the category *Par*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Par-arr*:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
shows *vfst-arrow A B* : $A \times_\circ B \mapsto_{cat\text{-}Par\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Par-arr'*[*cat-rel-Par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $AB = A \times_\circ B$
and $A' = A$
and $\mathfrak{C}' =$ *cat-Par* α
shows *vfst-arrow A B* : $AB \mapsto_{\mathfrak{C}'} A'$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Par-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Par-arr*:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
shows *vsnd-arrow A B* : $A \times_\circ B \mapsto_{cat\text{-}Par\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Par-arr'*[*cat-rel-Par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $AB = A \times_\circ B$
and $B' = B$
and $\mathfrak{C}' =$ *cat-Par* α
shows *vsnd-arrow A B* : $AB \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Par-arr'*

17.8.5 Projection arrows are arrows in the category *Rel*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Rel-arr*:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
shows *vfst-arrow A B* : $A \times_\circ B \mapsto_{cat\text{-}Rel\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Rel-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $AB = A \times_\circ B$
and $A' = A$
and $\mathfrak{C}' =$ *cat-Rel* α
shows *vfst-arrow A B* : $AB \mapsto_{\mathfrak{C}'} A'$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Rel-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Rel-arr*:

assumes $A \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$ **and** $B \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{vsnd-arrow}\ A\ B : A \times_{\circ} B \mapsto_{\textit{cat-Rel}\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Rel-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $AB = A \times_{\circ} B$
and $B' = B$
and $\mathfrak{C}' = \textit{cat-Rel}\ \alpha$
shows $\textit{vsnd-arrow}\ A\ B : AB \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Rel-arr'*

17.8.6 Projection arrows are isomorphisms in the category *Set*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-iso-arr-Vset*:
assumes $A \in_{\circ} \textit{Vset}\ \alpha$ **and** $b \in_{\circ} \textit{Vset}\ \alpha$
shows $\textit{vfst-arrow}\ A\ (\textit{set}\ \{b\}) : A \times_{\circ} \textit{set}\ \{b\} \mapsto_{iso \textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-iso-arr*:
assumes $A \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$ **and** $b \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{vfst-arrow}\ A\ (\textit{set}\ \{b\}) : A \times_{\circ} \textit{set}\ \{b\} \mapsto_{iso \textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Set-iso-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $b \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $AB = A \times_{\circ} \textit{set}\ \{b\}$
and $A' = A$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{vfst-arrow}\ A\ (\textit{set}\ \{b\}) : AB \mapsto_{iso \mathfrak{C}'} A$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Set-iso-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-iso-arr-Vset*:
assumes $a \in_{\circ} \textit{Vset}\ \alpha$ **and** $B \in_{\circ} \textit{Vset}\ \alpha$
shows $\textit{vsnd-arrow}\ (\textit{set}\ \{a\})\ B : \textit{set}\ \{a\} \times_{\circ} B \mapsto_{iso \textit{cat-Set}\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-iso-arr*:
assumes $a \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$ **and** $B \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{vsnd-arrow}\ (\textit{set}\ \{a\})\ B : \textit{set}\ \{a\} \times_{\circ} B \mapsto_{iso \textit{cat-Set}\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Set-iso-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $a \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $AB = \textit{set}\ \{a\} \times_{\circ} B$
and $A' = A$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{vsnd-arrow}\ (\textit{set}\ \{a\})\ B : AB \mapsto_{iso \mathfrak{C}'} B$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Set-iso-arr'*

17.8.7 Projection arrows are isomorphisms in the category *Par*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Par-iso-arr*:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$ **and** $b \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
shows *vfst-arrow* A $(set\ \{b\}) : A \times_\circ set\ \{b\} \mapsto_{iso} cat\text{-}Par\ \alpha\ A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Par-iso-arr'*[*cat-rel-Par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $b \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $AB = A \times_\circ set\ \{b\}$
and $A' = A$
and $\mathfrak{C}' = cat\text{-}Par\ \alpha$
shows *vfst-arrow* A $(set\ \{b\}) : AB \mapsto_{iso} \mathfrak{C}'\ A$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Par-iso-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Par-iso-arr*:
assumes $a \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
shows *vsnd-arrow* $(set\ \{a\})\ B : set\ \{a\} \times_\circ B \mapsto_{iso} cat\text{-}Par\ \alpha\ B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Par-iso-arr'*[*cat-rel-Par-set-cs-intros*]:
assumes $a \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Par* $\alpha(\!|Obj|\!)$
and $AB = set\ \{a\} \times_\circ B$
and $A' = A$
and $\mathfrak{C}' = cat\text{-}Par\ \alpha$
shows *vsnd-arrow* $(set\ \{a\})\ B : AB \mapsto_{iso} \mathfrak{C}'\ B$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Par-iso-arr'*

17.8.8 Projection arrows are isomorphisms in the category *Rel*

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Rel-iso-arr*:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$ **and** $b \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
shows *vfst-arrow* A $(set\ \{b\}) : A \times_\circ set\ \{b\} \mapsto_{iso} cat\text{-}Rel\ \alpha\ A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfst-arrow-is-cat-Rel-iso-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $b \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $AB = A \times_\circ set\ \{b\}$
and $A' = A$
and $\mathfrak{C}' = cat\text{-}Rel\ \alpha$
shows *vfst-arrow* A $(set\ \{b\}) : AB \mapsto_{iso} \mathfrak{C}'\ A$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] = $\mathcal{Z}$.*vfst-arrow-is-cat-Rel-iso-arr'*

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Rel-iso-arr*:
assumes $a \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
shows *vsnd-arrow* $(set\ \{a\})\ B : set\ \{a\} \times_\circ B \mapsto_{iso} cat\text{-}Rel\ \alpha\ B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vsnd-arrow-is-cat-Rel-iso-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $a \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$

and $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $AB =$ *set* $\{a\} \times_\circ B$
and $A' = A$
and $\mathfrak{C}' =$ *cat-Rel* α
shows *vsnd-arrow* (*set* $\{a\}$) $B : AB \mapsto_{iso\mathfrak{C}'} B$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] = $\mathcal{Z}$.*vsnd-arrow-is-cat-Rel-iso-arr'*

17.9 Projection arrow for *vproduct*

definition *vprojection-arrow* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$
where *vprojection-arrow I A i* = [*vprojection I A i*, $(\prod_\circ i \in_\circ I.\ A\ i)$, *A i*]$_\circ$

Components.

lemma *vprojection-arrow-components*:
shows *vprojection-arrow I A i*$(\!|ArrVal|\!)$ = *vprojection I A i*
and *vprojection-arrow I A i*$(\!|ArrDom|\!)$ = $(\prod_\circ i \in_\circ I.\ A\ i)$
and *vprojection-arrow I A i*$(\!|ArrCod|\!)$ = *A i*
⟨*proof*⟩

17.9.1 Projection arrow value

mk-VLambda *vprojection-arrow-components*(*1*)[*unfolded vprojection-def*]
|*vsv vprojection-arrow-ArrVal-vsv*[*cat-Set-cs-intros*]|
|*vdomain vprojection-arrow-ArrVal-vdomain*[*cat-Set-cs-simps*]|
|*app vprojection-arrow-ArrVal-app*[*cat-Set-cs-simps*]|

17.9.2 Projection arrow is an arrow in the category *Set*

lemma (**in** $\mathcal{Z}$) *arr-Set-vprojection-arrow*:
assumes $i \in_\circ I$ **and** *VLambda I A* $\in_\circ$ *Vset* α
shows *arr-Set* α (*vprojection-arrow I A i*)
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *vprojection-arrow-is-arr*:
assumes $i \in_\circ I$ **and** *VLambda I A* $\in_\circ$ *Vset* α
shows *vprojection-arrow I A i* : $(\prod_\circ i \in_\circ I.\ A\ i) \mapsto_{cat\text{-}Set\ \alpha} A\ i$
⟨*proof*⟩

17.10 Canonical injection arrow for *vdunion*

definition *vcinjection-arrow* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$
where *vcinjection-arrow I A i* = [*vcinjection A i*, *A i*, $(\coprod_\circ i \in_\circ I.\ A\ i)$]$_\circ$

Components.

lemma *vcinjection-arrow-components*:
shows *vcinjection-arrow I A i*$(\!|ArrVal|\!)$ = *vcinjection A i*
and *vcinjection-arrow I A i*$(\!|ArrDom|\!)$ = *A i*
and *vcinjection-arrow I A i*$(\!|ArrCod|\!)$ = $(\coprod_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

17.10.1 Canonical injection arrow value

mk-VLambda *vcinjection-arrow-components*(*1*)[*unfolded vcinjection-def*]
|*vsv vcinjection-arrow-ArrVal-vsv*[*cat-Set-cs-intros*]|
|*vdomain vcinjection-arrow-ArrVal-vdomain*[*cat-Set-cs-simps*]|
|*app vcinjection-arrow-ArrVal-app*[*cat-Set-cs-simps*]|

17.10.2 Canonical injection arrow is an arrow in the category *Set*

lemma (in $\mathcal{Z}$**)** *arr-Set-vcinjection-arrow*:
assumes $i \in_\circ I$ **and** $VLambda\ I\ A \in_\circ Vset\ \alpha$
shows $arr\text{-}Set\ \alpha\ (vcinjection\text{-}arrow\ I\ A\ i)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vcinjection-arrow-is-arr*:
assumes $i \in_\circ I$ **and** $VLambda\ I\ A \in_\circ Vset\ \alpha$
shows $vcinjection\text{-}arrow\ I\ A\ i : A\ i \mapsto_{cat\text{-}Set\ \alpha} (\coprod_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vcinjection-arrow-is-arr'*[*cat-cs-intros*]:
assumes $i \in_\circ I$
and $VLambda\ I\ A \in_\circ Vset\ \alpha$
and $A' = A\ i$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
and $P' = (\coprod_\circ i \in_\circ I.\ A\ i)$
shows $vcinjection\text{-}arrow\ I\ A\ i : A' \mapsto_{\mathfrak{C}'} P'$
⟨*proof*⟩

17.11 Product arrow value for *Rel*

17.11.1 Definition and elementary properties

definition $prod\text{-}2\text{-}Rel\text{-}ArrVal :: V \Rightarrow V \Rightarrow V$
where $prod\text{-}2\text{-}Rel\text{-}ArrVal\ S\ T =$
$set\ \{\langle\langle a,\ b\rangle,\ \langle c,\ d\rangle\rangle \mid a\ b\ c\ d.\ \langle a,\ c\rangle \in_\circ S \wedge \langle b,\ d\rangle \in_\circ T\}$

lemma *small-prod-2-Rel-ArrVal*[*simp*]:
$small\ \{\langle\langle a,\ b\rangle,\ \langle c,\ d\rangle\rangle \mid a\ b\ c\ d.\ \langle a,\ c\rangle \in_\circ S \wedge \langle b,\ d\rangle \in_\circ T\}$
(**is** ‹*small ?S*›)
⟨*proof*⟩

Rules.

lemma *prod-2-Rel-ArrValI*:
assumes $ab\text{-}cd = \langle\langle a,\ b\rangle,\ \langle c,\ d\rangle\rangle$
and $\langle a,\ c\rangle \in_\circ S$
and $\langle b,\ d\rangle \in_\circ T$
shows $ab\text{-}cd \in_\circ prod\text{-}2\text{-}Rel\text{-}ArrVal\ S\ T$
⟨*proof*⟩

lemma *prod-2-Rel-ArrValD*[*dest*]:
assumes $\langle\langle a,\ b\rangle,\ \langle c,\ d\rangle\rangle \in_\circ prod\text{-}2\text{-}Rel\text{-}ArrVal\ S\ T$
shows $\langle a,\ c\rangle \in_\circ S$ **and** $\langle b,\ d\rangle \in_\circ T$
⟨*proof*⟩

lemma *prod-2-Rel-ArrValE*[*elim*!]:
assumes $ab\text{-}cd \in_\circ prod\text{-}2\text{-}Rel\text{-}ArrVal\ S\ T$
obtains $a\ b\ c\ d$ **where** $ab\text{-}cd = \langle\langle a,\ b\rangle,\ \langle c,\ d\rangle\rangle$
and $\langle a,\ c\rangle \in_\circ S$
and $\langle b,\ d\rangle \in_\circ T$
⟨*proof*⟩

Elementary properties

lemma *prod-2-Rel-ArrVal-vsubset-vprod*:
$prod\text{-}2\text{-}Rel\text{-}ArrVal\ S\ T \subseteq_\circ ((\mathcal{D}_\circ\ S \times_\circ \mathcal{D}_\circ\ T) \times_\circ (\mathcal{R}_\circ\ S \times_\circ \mathcal{R}_\circ\ T))$
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-vbrelation*: *vbrelation* (*prod-2-Rel-ArrVal S T*)
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-vdomain*: $\mathcal{D}_\circ$ (*prod-2-Rel-ArrVal S T*) = $\mathcal{D}_\circ$ *S* $\times_\circ$ $\mathcal{D}_\circ$ *T*
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-vrange*: $\mathcal{R}_\circ$ (*prod-2-Rel-ArrVal S T*) = $\mathcal{R}_\circ$ *S* $\times_\circ$ $\mathcal{R}_\circ$ *T*
⟨*proof*⟩

17.11.2 Further properties

lemma
assumes *vsv g* **and** *vsv f*
shows *prod-2-Rel-ArrVal-vsv*: *vsv* (*prod-2-Rel-ArrVal g f*)
and *prod-2-Rel-ArrVal-app*:
$\bigwedge a\ b.\ [\![\ a \in_\circ \mathcal{D}_\circ\ g;\ b \in_\circ \mathcal{D}_\circ\ f\ ]\!] \Longrightarrow$
prod-2-Rel-ArrVal g f(|⟨*a*,*b*⟩|) = ⟨*g*(|*a*|), *f*(|*b*|)⟩
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-v11*:
assumes *v11 g* **and** *v11 f*
shows *v11* (*prod-2-Rel-ArrVal g f*)
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-vcomp*:
prod-2-Rel-ArrVal S′ T′ $\circ_\circ$ *prod-2-Rel-ArrVal S T* =
prod-2-Rel-ArrVal (*S′* $\circ_\circ$ *S*) (*T′* $\circ_\circ$ *T*)
⟨*proof*⟩

lemma *prod-2-Rel-ArrVal-vid-on*[*cat-cs-simps*]:
prod-2-Rel-ArrVal (*vid-on A*) (*vid-on B*) = *vid-on* (*A* $\times_\circ$ *B*)
⟨*proof*⟩

17.12 Product arrow for *Rel*

17.12.1 Definition and elementary properties

definition *prod-2-Rel* :: *V* ⇒ *V* ⇒ *V* (**infixr** ‹$_A\times_{Rel}$› *80*)
where *prod-2-Rel S T* =
[
prod-2-Rel-ArrVal (*S*(|*ArrVal*|)) (*T*(|*ArrVal*|)),
S(|*ArrDom*|) $\times_\circ$ *T*(|*ArrDom*|),
S(|*ArrCod*|) $\times_\circ$ *T*(|*ArrCod*|)
]$_\circ$

abbreviation (*input*) *prod-2-Par* :: *V* ⇒ *V* ⇒ *V* (**infixr** ‹$_A\times_{Par}$› *80*)
where *prod-2-Par* ≡ *prod-2-Rel*
abbreviation (*input*) *prod-2-Set* :: *V* ⇒ *V* ⇒ *V* (**infixr** ‹$_A\times_{Set}$› *80*)
where *prod-2-Set* ≡ *prod-2-Rel*

Components.

lemma *prod-2-Rel-components*:
shows (*S* $_A\times_{Rel}$ *T*)(|*ArrVal*|) = *prod-2-Rel-ArrVal* (*S*(|*ArrVal*|)) (*T*(|*ArrVal*|))
and [*cat-cs-simps*]: (*S* $_A\times_{Rel}$ *T*)(|*ArrDom*|) = *S*(|*ArrDom*|) $\times_\circ$ *T*(|*ArrDom*|)
and [*cat-cs-simps*]: (*S* $_A\times_{Rel}$ *T*)(|*ArrCod*|) = *S*(|*ArrCod*|) $\times_\circ$ *T*(|*ArrCod*|)
⟨*proof*⟩

17.12.2 Product arrow for *Rel* is an arrow in *Rel*

lemma *prod-2-Rel-is-cat-Rel-arr*:
assumes $S : A \mapsto_{cat\text{-}Rel\ \alpha} B$ **and** $T : C \mapsto_{cat\text{-}Rel\ \alpha} D$
shows $S\ {}_A\times_{Rel}\ T : A \times_\circ C \mapsto_{cat\text{-}Rel\ \alpha} B \times_\circ D$
⟨*proof*⟩

lemma *prod-2-Rel-is-cat-Rel-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $S : A \mapsto_{cat\text{-}Rel\ \alpha} B$
and $T : C \mapsto_{cat\text{-}Rel\ \alpha} D$
and $A' = A \times_\circ C$
and $B' = B \times_\circ D$
and $\mathfrak{C}' = cat\text{-}Rel\ \alpha$
shows $S\ {}_A\times_{Rel}\ T : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

17.12.3 Product arrow for *Rel* is an arrow in *Set*

lemma *prod-2-Rel-app*[*cat-rel-par-Set-cs-simps*]:
assumes $S : A \mapsto_{cat\text{-}Set\ \alpha} B$
and $T : C \mapsto_{cat\text{-}Set\ \alpha} D$
and $a \in_\circ A$
and $c \in_\circ C$
and $ac = \langle a, c \rangle$
shows $(S\ {}_A\times_{Set}\ T)(\!|ArrVal|\!)(\!|ac|\!) = \langle S(\!|ArrVal|\!)(\!|a|\!),\ T(\!|ArrVal|\!)(\!|c|\!)\rangle$
⟨*proof*⟩

lemma *prod-2-Rel-is-cat-Set-arr*:
assumes $S : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $T : C \mapsto_{cat\text{-}Set\ \alpha} D$
shows $S\ {}_A\times_{Set}\ T : A \times_\circ C \mapsto_{cat\text{-}Set\ \alpha} B \times_\circ D$
⟨*proof*⟩

lemma *prod-2-Rel-is-cat-Set-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $S : A \mapsto_{cat\text{-}Set\ \alpha} B$
and $T : C \mapsto_{cat\text{-}Set\ \alpha} D$
and $AC = A \times_\circ C$
and $BD = B \times_\circ D$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $S\ {}_A\times_{Set}\ T : AC \mapsto_{\mathfrak{C}'} BD$
⟨*proof*⟩

17.12.4 Product arrow for *Rel* is an isomorphism in *Set*

lemma *prod-2-Rel-is-cat-Set-iso-arr*:
assumes $S : A \mapsto_{iso\,cat\text{-}Set\ \alpha} B$ **and** $T : C \mapsto_{iso\,cat\text{-}Set\ \alpha} D$
shows $S\ {}_A\times_{Set}\ T : A \times_\circ C \mapsto_{iso\,cat\text{-}Set\ \alpha} B \times_\circ D$
⟨*proof*⟩

lemma *prod-2-Rel-is-cat-Set-iso-arr'*[*cat-rel-par-Set-cs-intros*]:
assumes $S : A \mapsto_{iso\,cat\text{-}Set\ \alpha} B$
and $T : C \mapsto_{iso\,cat\text{-}Set\ \alpha} D$
and $AC = A \times_\circ C$
and $BD = B \times_\circ D$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $S\ {}_A\times_{Set}\ T : AC \mapsto_{iso\,\mathfrak{C}'} BD$
⟨*proof*⟩

17.12.5 Further elementary properties

lemma *prod-2-Rel-Comp*:
assumes $G' : B' \mapsto_{cat\text{-}Rel\ \alpha} B''$
and $F' : A' \mapsto_{cat\text{-}Rel\ \alpha} A''$
and $G : B \mapsto_{cat\text{-}Rel\ \alpha} B'$
and $F : A \mapsto_{cat\text{-}Rel\ \alpha} A'$
shows
$G'\ {}_{A\times_{Rel}}\ F' \circ_{A\, cat\text{-}Rel\ \alpha}\ G\ {}_{A\times_{Rel}}\ F =$
$(G' \circ_{A\, cat\text{-}Rel\ \alpha}\ G)\ {}_{A\times_{Rel}}\ (F' \circ_{A\, cat\text{-}Rel\ \alpha}\ F)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *prod-2-Rel-CId*[*cat-cs-simps*]:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$ **and** $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
shows
$(cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|A|\!))\ {}_{A\times_{Rel}}\ (cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|B|\!)) = cat\text{-}Rel\ \alpha(\!|CId|\!)(\!|A \times_\circ B|\!)$
⟨*proof*⟩

lemma *cf-dag-Rel-ArrMap-app-prod-2-Rel*:
assumes $S : A \mapsto_{cat\text{-}Rel\ \alpha} B$ **and** $T : C \mapsto_{cat\text{-}Rel\ \alpha} D$
shows
$\dagger_{C.Rel}\ \alpha(\!|ArrMap|\!)(\!|S\ {}_{A\times_{Rel}}\ T|\!) =$
$(\dagger_{C.Rel}\ \alpha(\!|ArrMap|\!)(\!|S|\!))\ {}_{A\times_{Rel}}\ (\dagger_{C.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!))$
⟨*proof*⟩

17.13 Product functor for *Rel*

definition *cf-prod-2-Rel* :: $V \Rightarrow V$
where *cf-prod-2-Rel* $\mathfrak{A}$ =
$[$
$(\lambda AB{\in_\circ}(\mathfrak{A} \times_C \mathfrak{A})(\!|Obj|\!).\ AB(\!|0|\!) \times_\circ AB(\!|1_{\mathbb{N}}|\!))$,
$(\lambda ST{\in_\circ}(\mathfrak{A} \times_C \mathfrak{A})(\!|Arr|\!).\ (ST(\!|0|\!))\ {}_{A\times_{Rel}}\ (ST(\!|1_{\mathbb{N}}|\!)))$,
$\mathfrak{A} \times_C \mathfrak{A}$,
$\mathfrak{A}$
$]_\circ$

Components.

lemma *cf-prod-2-Rel-components*:
shows *cf-prod-2-Rel* $\mathfrak{A}(\!|ObjMap|\!) = (\lambda AB{\in_\circ}(\mathfrak{A} \times_C \mathfrak{A})(\!|Obj|\!).\ AB(\!|0|\!) \times_\circ AB(\!|1_{\mathbb{N}}|\!))$
and *cf-prod-2-Rel* $\mathfrak{A}(\!|ArrMap|\!) =$
$(\lambda ST{\in_\circ}(\mathfrak{A} \times_C \mathfrak{A})(\!|Arr|\!).\ (ST(\!|0|\!))\ {}_{A\times_{Rel}}\ (ST(\!|1_{\mathbb{N}}|\!)))$
and [*cat-cs-simps*]: *cf-prod-2-Rel* $\mathfrak{A}(\!|HomDom|\!) = \mathfrak{A} \times_C \mathfrak{A}$
and [*cat-cs-simps*]: *cf-prod-2-Rel* $\mathfrak{A}(\!|HomCod|\!) = \mathfrak{A}$
⟨*proof*⟩

17.13.1 Object map

mk-VLambda *cf-prod-2-Rel-components*(*1*)
|*vsv cf-prod-2-Rel-ObjMap-vsv*[*cat-cs-intros*]|
|*vdomain cf-prod-2-Rel-ObjMap-vdomain*[*cat-cs-simps*]|

lemma *cf-prod-2-Rel-ObjMap-app*[*cat-cs-simps*]:
assumes $AB = [A, B]_\circ$ **and** $AB \in_\circ (\mathfrak{A} \times_C \mathfrak{A})(\!|Obj|\!)$
shows $A \otimes_{HM.O\, cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{A}} B = A \times_\circ B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cf-prod-2-Rel-ObjMap-vrange*:
$\mathcal{R}_\circ\ (cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha)(\!|ObjMap|\!)) \subseteq_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
⟨*proof*⟩

17.13.2 Arrow map

mk-VLambda *cf-prod-2-Rel-components*(*2*)
|*vsv cf-prod-2-Rel-ArrMap-vsv*[*cat-cs-intros*]|
|*vdomain cf-prod-2-Rel-ArrMap-vdomain*[*cat-cs-simps*]|

lemma *cf-prod-2-Rel-ArrMap-app*[*cat-cs-simps*]:
assumes $GF = [G, F]_\circ$ **and** $GF \in_\circ (\mathfrak{A} \times_C \mathfrak{A})(\!|Arr|\!)$
shows $G \otimes_{HM.A} cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{A}\ F = G\ _{A\times_{Rel}}\ F$
⟨*proof*⟩

17.13.3 Product functor for *Rel* is a functor

lemma (in $\mathcal{Z}$**)** *cf-prod-2-Rel-is-functor*:
$cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha) : cat\text{-}Rel\ \alpha \times_C cat\text{-}Rel\ \alpha \mapsto\mapsto_{C\alpha} cat\text{-}Rel\ \alpha$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cf-prod-2-Rel-is-functor'*[*cat-cs-intros*]:
assumes $\mathfrak{A}' = cat\text{-}Rel\ \alpha \times_C cat\text{-}Rel\ \alpha$
and $\mathfrak{B}' = cat\text{-}Rel\ \alpha$
and $\alpha' = \alpha$
shows $cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha) : \mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$.*cf-prod-2-Rel-is-functor'*

17.14 Product universal property arrow for *Set*

17.14.1 Definition and elementary properties

definition $cat\text{-}Set\text{-}obj\text{-}prod\text{-}up :: V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where $cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi =$
$[(\lambda a \in_\circ A.\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ArrVal|\!)(\!|a|\!))),\ A,\ (\prod_\circ i \in_\circ I.\ F\ i)]_\circ$

Components.

lemma *cat-Set-obj-prod-up-components*:
shows $cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi(\!|ArrVal|\!) =$
$(\lambda a \in_\circ A.\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ArrVal|\!)(\!|a|\!)))$
and [*cat-Set-cs-simps*]:
$cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi(\!|ArrDom|\!) = A$
and [*cat-Set-cs-simps*]:
$cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi(\!|ArrCod|\!) = (\prod_\circ i \in_\circ I.\ F\ i)$
⟨*proof*⟩

17.14.2 Arrow value

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

lemma *cat-Set-obj-prod-up-ArrVal-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : A \mapsto_{cat\text{-}Set\ \alpha} F\ i$
shows $\mathcal{R}_\circ\ (cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi(\!|ArrVal|\!)) \subseteq_\circ (\prod_\circ i \in_\circ I.\ F\ i)$
⟨*proof*⟩

lemma *cat-Set-obj-prod-up-ArrVal-app-vdomain*[*cat-Set-cs-simps*]:
assumes $a \in_\circ A$
shows $\mathcal{D}_\circ\ (cat\text{-}Set\text{-}obj\text{-}prod\text{-}up\ I\ F\ A\ \varphi(\!|ArrVal|\!)(\!|a|\!)) = I$

⟨*proof*⟩

lemma *cat-Set-obj-prod-up-ArrVal-app-component*[*cat-Set-cs-simps*]:
assumes $a \in_\circ A$ **and** $i \in_\circ I$
shows *cat-Set-obj-prod-up I F A* $\varphi(\!|ArrVal|\!)(\!|a|\!)(\!|i|\!) = \varphi\ i(\!|ArrVal|\!)(\!|a|\!)$
⟨*proof*⟩

lemma *cat-Set-obj-prod-up-ArrVal-app-vrange*:
assumes $a \in_\circ A$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : A \mapsto_{cat\text{-}Set\ \alpha} F\ i$
shows $\mathcal{R}_\circ$ (*cat-Set-obj-prod-up I F A* $\varphi(\!|ArrVal|\!)(\!|a|\!)) \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ F\ i)$
⟨*proof*⟩

17.14.3 Product universal property arrow for *Set* is an arrow in *Set*

lemma (in $\mathcal{Z}$**)** *cat-Set-obj-prod-up-cat-Set-is-arr*:
assumes $A \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and *VLambda I F* $\in_\circ$ *Vset* α
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : A \mapsto_{cat\text{-}Set\ \alpha} F\ i$
shows *cat-Set-obj-prod-up I F A* $\varphi : A \mapsto_{cat\text{-}Set\ \alpha} (\prod_\circ i \in_\circ I.\ F\ i)$
⟨*proof*⟩

17.14.4 Further properties

lemma (in $\mathcal{Z}$**)** *cat-Set-cf-comp-proj-obj-prod-up*:
assumes $A \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and *VLambda I F* $\in_\circ$ *Vset* α
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : A \mapsto_{cat\text{-}Set\ \alpha} F\ i$
and $i \in_\circ I$
shows
$\varphi\ i =$ *vprojection-arrow I F i* $\circ_{A\,cat\text{-}Set\ \alpha}$ *cat-Set-obj-prod-up I F A* φ
(**is** ‹$\varphi\ i =$ *?Fi* $\circ_{A\,cat\text{-}Set\ \alpha}$ *?*φ›)
⟨*proof*⟩

17.15 Coproduct universal property arrow for *Set*

17.15.1 Definition and elementary properties

definition *cat-Set-obj-coprod-up* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *cat-Set-obj-coprod-up I F A* φ =
$[(\lambda ix \in_\circ (\coprod_\circ i \in_\circ I.\ F\ i).\ \varphi\ (vfst\ ix)(\!|ArrVal|\!)(\!|vsnd\ ix|\!)), (\coprod_\circ i \in_\circ I.\ F\ i), A]_\circ$

Components.

lemma *cat-Set-obj-coprod-up-components*:
shows *cat-Set-obj-coprod-up I F A* $\varphi(\!|ArrVal|\!) =$
$(\lambda ix \in_\circ (\coprod_\circ i \in_\circ I.\ F\ i).\ \varphi\ (vfst\ ix)(\!|ArrVal|\!)(\!|vsnd\ ix|\!))$
and [*cat-Set-cs-simps*]:
cat-Set-obj-coprod-up I F A $\varphi(\!|ArrDom|\!) = (\coprod_\circ i \in_\circ I.\ F\ i)$
and [*cat-Set-cs-simps*]:
cat-Set-obj-coprod-up I F A $\varphi(\!|ArrCod|\!) = A$
⟨*proof*⟩

17.15.2 Arrow value

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

lemma *cat-Set-obj-coprod-up-ArrVal-app*[*cat-cs-simps*]:

assumes $ix = \langle i,\ x\rangle$ **and** $\langle i,\ x\rangle \in_\circ (\coprod_\circ i \in_\circ I.\ F\ i)$
shows *cat-Set-obj-coprod-up I F A* $\varphi(\!|ArrVal|\!)(\!|ix|\!) = \varphi\ i(\!|ArrVal|\!)(\!|x|\!)$
⟨*proof*⟩

lemma *cat-Set-obj-coprod-up-ArrVal-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : F\ i \mapsto_{cat\text{-}Set\ \alpha} A$
shows $\mathcal{R}_\circ$ (*cat-Set-obj-coprod-up I F A* $\varphi(\!|ArrVal|\!)) \subseteq_\circ A$
⟨*proof*⟩

17.15.3 Coproduct universal property arrow for *Set* is an arrow in *Set*

lemma (in $\mathcal{Z}$**)** *cat-Set-obj-coprod-up-cat-Set-is-arr*:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\!|Obj|\!)$
and *VLambda I F* $\in_\circ$ *Vset* α
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : F\ i \mapsto_{cat\text{-}Set\ \alpha} A$
shows *cat-Set-obj-coprod-up I F A* $\varphi : (\coprod_\circ i \in_\circ I.\ F\ i) \mapsto_{cat\text{-}Set\ \alpha} A$
⟨*proof*⟩

17.15.4 Further properties

lemma (in $\mathcal{Z}$**)** *cat-Set-cf-comp-coprod-up-vcia*:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\!|Obj|\!)$
and *VLambda I F* $\in_\circ$ *Vset* α
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : F\ i \mapsto_{cat\text{-}Set\ \alpha} A$
and $i \in_\circ I$
shows
$\varphi\ i = cat\text{-}Set\text{-}obj\text{-}coprod\text{-}up\ I\ F\ A\ \varphi \circ_{A\, cat\text{-}Set\ \alpha}\ vcinjection\text{-}arrow\ I\ F\ i$
(**is** ‹$\varphi\ i = ?\varphi \circ_{A\, cat\text{-}Set\ \alpha}\ ?Fi$›)
⟨*proof*⟩

17.16 Equalizer object for the category *Set*

The definition of the (non-categorical concept of an) equalizer can be found in [2][10]

definition *vequalizer* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *vequalizer X f g* = *set* $\{x.\ x \in_\circ X \wedge f(\!|ArrVal|\!)(\!|x|\!) = g(\!|ArrVal|\!)(\!|x|\!)\}$

lemma *small-vequalizer*[*simp*]:
small $\{x.\ x \in_\circ X \wedge f(\!|ArrVal|\!)(\!|x|\!) = g(\!|ArrVal|\!)(\!|x|\!)\}$
⟨*proof*⟩

Rules.

lemma *vequalizerI*:
assumes $x \in_\circ X$ **and** $f(\!|ArrVal|\!)(\!|x|\!) = g(\!|ArrVal|\!)(\!|x|\!)$
shows $x \in_\circ$ *vequalizer X f g*
⟨*proof*⟩

lemma *vequalizerD*[*dest*]:
assumes $x \in_\circ$ *vequalizer X f g*
shows $x \in_\circ X$ **and** $f(\!|ArrVal|\!)(\!|x|\!) = g(\!|ArrVal|\!)(\!|x|\!)$
⟨*proof*⟩

lemma *vequalizerE*[*elim*]:
assumes $x \in_\circ$ *vequalizer X f g*
obtains $x \in_\circ X$ **and** $f(\!|ArrVal|\!)(\!|x|\!) = g(\!|ArrVal|\!)(\!|x|\!)$
⟨*proof*⟩

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

Elementary results.

lemma *vequalizer-vsubset-vdomain*[*cat-Set-cs-intros*]: *vequalizer a g f* $\subseteq_\circ$ *a*
⟨*proof*⟩

lemma *Limit-vequalizer-in-Vset*[*cat-Set-cs-intros*]:
assumes *Limit* α **and** *a* $\in_\circ$ *cat-Set* $\alpha(\lfloor Obj \rfloor)$
shows *vequalizer a g f* $\in_\circ$ *cat-Set* $\alpha(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *vequalizer-flip*: *vequalizer a f g* = *vequalizer a g f*
⟨*proof*⟩

lemma *cat-Set-incl-Set-commute*:
assumes $\mathfrak{g} : \mathfrak{a} \mapsto_{cat\text{-}Set\ \alpha} \mathfrak{b}$ **and** $\mathfrak{f} : \mathfrak{a} \mapsto_{cat\text{-}Set\ \alpha} \mathfrak{b}$
shows
$\mathfrak{g} \circ_{A\, cat\text{-}Set\ \alpha}$ *incl-Set* (*vequalizer* $\mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}$) $\mathfrak{a}$ =
$\mathfrak{f} \circ_{A\, cat\text{-}Set\ \alpha}$ *incl-Set* (*vequalizer* $\mathfrak{a}\ \mathfrak{f}\ \mathfrak{g}$) $\mathfrak{a}$
(**is** ‹$\mathfrak{g} \circ_{A\, cat\text{-}Set\ \alpha}$ *?incl* = $\mathfrak{f} \circ_{A\, cat\text{-}Set\ \alpha}$ *?incl*›)
⟨*proof*⟩

17.17 Application of a function to a finite sequence as an arrow in *Set*

definition *vfsequence-map* :: *V* ⇒ *V*
where *vfsequence-map F* =
[
(λ*xs*∈$_\circ$*vfsequences-on* (*F*(⌊*ArrDom*⌋)). *F*(⌊*ArrVal*⌋) $\circ_\circ$ *xs*),
vfsequences-on (*F*(⌊*ArrDom*⌋)),
vfsequences-on (*F*(⌊*ArrCod*⌋))
]$_\circ$

Components.

lemma *vfsequence-map-components*:
shows *vfsequence-map F*(⌊*ArrVal*⌋) =
(λ*xs*∈$_\circ$*vfsequences-on* (*F*(⌊*ArrDom*⌋)). *F*(⌊*ArrVal*⌋) $\circ_\circ$ *xs*)
and [*cat-cs-simps*]: *vfsequence-map F*(⌊*ArrDom*⌋) = *vfsequences-on* (*F*(⌊*ArrDom*⌋))
and [*cat-cs-simps*]: *vfsequence-map F*(⌊*ArrCod*⌋) = *vfsequences-on* (*F*(⌊*ArrCod*⌋))
⟨*proof*⟩

17.17.1 Arrow value

mk-VLambda *vfsequence-map-components*(*1*)
|*vsv vfsequence-map-ArrVal-vsv*[*cat-cs-intros*, *cat-Set-cs-intros*]|
|*vdomain vfsequence-map-ArrVal-vdomain*[*cat-cs-simps*, *cat-Set-cs-simps*]|
|*app vfsequence-map-ArrVal-app*|

lemma *vfsequence-map-ArrVal-app-app*:
assumes *F* : *A* $\mapsto_{cat\text{-}Set\ \alpha}$ *B*
and *xs* $\in_\circ$ *vfsequences-on A*
and *i* $\in_\circ$ $\mathcal{D}_\circ$ *xs*
shows *vfsequence-map F*(⌊*ArrVal*⌋)(⌊*xs*⌋)(⌊*i*⌋) = *F*(⌊*ArrVal*⌋)(⌊*xs*(⌊*i*⌋)⌋)
⟨*proof*⟩

17.17.2 Application of a function to a finite sequence is an arrow in *Set*

lemma *vfsequence-map-is-arr*:
assumes *F* : *A* $\mapsto_{cat\text{-}Set\ \alpha}$ *B*
shows *vfsequence-map F* : *vfsequences-on A* $\mapsto_{cat\text{-}Set\ \alpha}$ *vfsequences-on B*
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfsequence-map-is-monic-arr*:
assumes $F : A \mapsto_{mon cat\text{-}Set\ \alpha} B$
shows *vfsequence-map* F : *vfsequences-on* $A \mapsto_{mon cat\text{-}Set\ \alpha}$ *vfsequences-on* B
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *vfsequence-map-is-epic-arr*:
assumes $F : A \mapsto_{epi cat\text{-}Set\ \alpha} B$
shows *vfsequence-map* F : *vfsequences-on* $A \mapsto_{epi cat\text{-}Set\ \alpha}$ *vfsequences-on* B
⟨*proof*⟩

lemma *vfsequence-map-is-iso-arr*:
assumes $F : A \mapsto_{iso cat\text{-}Set\ \alpha} B$
shows *vfsequence-map* F : *vfsequences-on* $A \mapsto_{iso cat\text{-}Set\ \alpha}$ *vfsequences-on* B
⟨*proof*⟩

17.18 An injection from the range of an arrow in *Set* into its domain

17.18.1 Definition and elementary properties

definition *vrange-iso* :: $V \Rightarrow V$
where *vrange-iso* F =
$[$
$(\lambda y \in_\circ \mathcal{R}_\circ\ (F(\!|ArrVal|\!)).\ (SOME\ x.\ x \in_\circ F(\!|ArrDom|\!) \wedge y = F(\!|ArrVal|\!)(\!|x|\!)))$,
$\mathcal{R}_\circ\ (F(\!|ArrVal|\!))$,
$F(\!|ArrDom|\!)$
$]_\circ$

Components.

lemma *vrange-iso-components*:
shows *vrange-iso* $F(\!|ArrVal|\!)$ =
$(\lambda y \in_\circ \mathcal{R}_\circ\ (F(\!|ArrVal|\!)).\ (SOME\ x.\ x \in_\circ F(\!|ArrDom|\!) \wedge y = F(\!|ArrVal|\!)(\!|x|\!)))$
and [*cat-cs-simps*]: *vrange-iso* $F(\!|ArrDom|\!) = \mathcal{R}_\circ\ (F(\!|ArrVal|\!))$
and [*cat-cs-simps*]: *vrange-iso* $F(\!|ArrCod|\!) = F(\!|ArrDom|\!)$
⟨*proof*⟩

17.18.2 Arrow value

mk-VLambda *vrange-iso-components*(*1*)
|*vsv vrange-iso-ArrVal-vsv*[*cat-cs-intros*]|
|*vdomain vrange-iso-ArrVal-vdomain*[*cat-cs-simps*]|
|*app vrange-iso-ArrVal-app*|

lemma *vrange-iso-ArrVal-rules*:
assumes $F : A \mapsto_{cat\text{-}Set\ \alpha} B$ **and** $y \in_\circ \mathcal{R}_\circ\ (F(\!|ArrVal|\!))$
shows *vrange-iso* $F(\!|ArrVal|\!)(\!|y|\!) \in_\circ A$
and $y = F(\!|ArrVal|\!)(\!|$*vrange-iso* $F(\!|ArrVal|\!)(\!|y|\!)|\!)$
⟨*proof*⟩

17.18.3 An injection from the range of a function into its domain is a monic in *Set*

lemma *vrange-iso-is-arr*:
assumes $F : A \mapsto_{cat\text{-}Set\ \alpha} B$
shows *vrange-iso* $F : \mathcal{R}_\circ\ (F(\!|ArrVal|\!)) \mapsto_{cat\text{-}Set\ \alpha} A$
⟨*proof*⟩

lemma *vrange-iso-is-arr'*:
assumes $F : A \mapsto_{cat\text{-}Set\ \alpha} B$
and $B' = \mathcal{R}_\circ\ (F(\!|ArrVal|\!))$

and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{vrange-iso}\ F : B' \mapsto_{\mathfrak{C}'} A$
⟨*proof*⟩

lemma *vrange-iso-is-monic-arr*:
assumes $F : A \mapsto_{\textit{cat-Set}\ \alpha} B$
shows $\textit{vrange-iso}\ F : \mathcal{R}_{\circ}\ (F(\!|\textit{ArrVal}|\!)) \mapsto_{mon\textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma *vrange-iso-is-monic-arr′*:
assumes $F : A \mapsto_{\textit{cat-Set}\ \alpha} B$
and $B' = \mathcal{R}_{\circ}\ (F(\!|\textit{ArrVal}|\!))$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{vrange-iso}\ F : B' \mapsto_{mon\mathfrak{C}'} A$
⟨*proof*⟩

17.19 Auxiliary

This subsection is reserved for insignificant helper lemmas and rules that are used in applied formalization elsewhere.

lemma (in $\mathcal{Z}$**)** *cat-Rel-CId-is-cat-Set-arr*:
assumes $A \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{cat-Rel}\ \alpha(\!|\textit{CId}|\!)(\!|A|\!) : A \mapsto_{\textit{cat-Set}\ \alpha} A$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *cat-Rel-CId-is-cat-Set-arr′*[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_{\circ} \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $B' = A$
and $C' = A$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{cat-Rel}\ \alpha(\!|\textit{CId}|\!)(\!|A|\!) : B' \mapsto_{\mathfrak{C}'} C'$
⟨*proof*⟩

18 *GRPH*

18.1 Background

The methodology for the exposition of *GRPH* as a category is analogous to the one used in [8] for the exposition of *GRPH* as a semicategory.

named-theorems *cat-GRPH-simps*
named-theorems *cat-GRPH-intros*

18.2 Definition and elementary properties

definition *cat-GRPH* :: $V \Rightarrow V$
  where *cat-GRPH* α =
    [
      set $\{\mathfrak{C}.\ digraph\ \alpha\ \mathfrak{C}\}$,
      all-dghms α,
      $(\lambda\mathfrak{F}\in_\circ all\text{-}dghms\ \alpha.\ \mathfrak{F}(\!|HomDom|\!))$,
      $(\lambda\mathfrak{F}\in_\circ all\text{-}dghms\ \alpha.\ \mathfrak{F}(\!|HomCod|\!))$,
      $(\lambda\mathfrak{GF}\in_\circ composable\text{-}arrs\ (dg\text{-}GRPH\ \alpha).\ \mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$,
      $(\lambda\mathfrak{C}\in_\circ set\ \{\mathfrak{C}.\ digraph\ \alpha\ \mathfrak{C}\}.\ dghm\text{-}id\ \mathfrak{C})$
    ]$_\circ$

Components.

lemma *cat-GRPH-components*:
  shows *cat-GRPH* $\alpha(\!|Obj|\!)$ = *set* $\{\mathfrak{C}.\ digraph\ \alpha\ \mathfrak{C}\}$
    and *cat-GRPH* $\alpha(\!|Arr|\!)$ = *all-dghms* α
    and *cat-GRPH* $\alpha(\!|Dom|\!) = (\lambda\mathfrak{F}\in_\circ all\text{-}dghms\ \alpha.\ \mathfrak{F}(\!|HomDom|\!))$
    and *cat-GRPH* $\alpha(\!|Cod|\!) = (\lambda\mathfrak{F}\in_\circ all\text{-}dghms\ \alpha.\ \mathfrak{F}(\!|HomCod|\!))$
    and *cat-GRPH* $\alpha(\!|Comp|\!)$ =
      $(\lambda\mathfrak{GF}\in_\circ composable\text{-}arrs\ (dg\text{-}GRPH\ \alpha).\ \mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
    and *cat-GRPH* $\alpha(\!|CId|\!) = (\lambda\mathfrak{C}\in_\circ set\ \{\mathfrak{C}.\ digraph\ \alpha\ \mathfrak{C}\}.\ dghm\text{-}id\ \mathfrak{C})$
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-GRPH*: *cat-smc* (*cat-GRPH* α) = *smc-GRPH* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-GRPH*, *unfolded slicing-simps*]:
  — Digraph
  cat-GRPH-ObjI = *smc-GRPH-ObjI*
  and *cat-GRPH-ObjD* = *smc-GRPH-ObjD*
  and *cat-GRPH-ObjE* = *smc-GRPH-ObjE*
  and *cat-GRPH-Obj-iff*[*cat-GRPH-simps*] = *smc-GRPH-Obj-iff*
  and *cat-GRPH-Dom-app*[*cat-GRPH-simps*] = *smc-GRPH-Dom-app*
  and *cat-GRPH-Cod-app*[*cat-GRPH-simps*] = *smc-GRPH-Cod-app*
  and *cat-GRPH-is-arrI* = *smc-GRPH-is-arrI*
  and *cat-GRPH-is-arrD* = *smc-GRPH-is-arrD*
  and *cat-GRPH-is-arrE* = *smc-GRPH-is-arrE*
  and *cat-GRPH-is-arr-iff*[*cat-GRPH-simps*] = *smc-GRPH-is-arr-iff*

lemmas-with [*folded cat-smc-GRPH*, *unfolded slicing-simps*, *unfolded cat-smc-GRPH*]:
  — Semicategory
  cat-GRPH-Comp-vdomain = *smc-GRPH-Comp-vdomain*
  and *cat-GRPH-composable-arrs-dg-GRPH* = *smc-GRPH-composable-arrs-dg-GRPH*
  and *cat-GRPH-Comp* = *smc-GRPH-Comp*
  and *cat-GRPH-Comp-app*[*cat-GRPH-simps*] = *smc-GRPH-Comp-app*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-GRPH*, *unfolded slicing-simps*]:
— Semicategory
cat-GRPH-obj-initialI = *smc-GRPH-obj-initialI*
and *cat-GRPH-obj-initialD* = *smc-GRPH-obj-initialD*
and *cat-GRPH-obj-initialE* = *smc-GRPH-obj-initialE*
and *cat-GRPH-obj-initial-iff*[*cat-GRPH-simps*] = *smc-GRPH-obj-initial-iff*
and *cat-GRPH-obj-terminalI* = *smc-GRPH-obj-terminalI*
and *cat-GRPH-obj-terminalE* = *smc-GRPH-obj-terminalE*

18.3 Identity

lemma *cat-GRPH-CId-app*[*cat-GRPH-simps*]:
assumes *digraph* α $\mathfrak{C}$
shows *cat-GRPH* $\alpha(\!|CId|\!)(\!|\mathfrak{C}|\!)$ = *dghm-id* $\mathfrak{C}$
⟨*proof*⟩

lemma *cat-GRPH-CId-vdomain*: $\mathcal{D}_\circ$ (*cat-GRPH* $\alpha(\!|CId|\!)$) = *set* $\{\mathfrak{C}.\ digraph\ \alpha\ \mathfrak{C}\}$
⟨*proof*⟩

lemma *cat-GRPH-CId-vrange*: $\mathcal{R}_\circ$ (*cat-GRPH* $\alpha(\!|CId|\!)$) $\subseteq_\circ$ *all-dghms* α
⟨*proof*⟩

18.4 *GRPH* is a category

lemma (in $\mathcal{Z}$**)** *tiny-category-cat-GRPH*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-category* β (*cat-GRPH* α)
⟨*proof*⟩

18.5 Isomorphism

lemma *cat-GRPH-is-iso-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}GRPH\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-is-iso-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}GRPH\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-is-iso-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}GRPH\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-is-iso-arr-iff*[*cat-GRPH-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}GRPH\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

18.6 Isomorphic objects

lemma *cat-GRPH-obj-isoI*:
assumes $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{obj cat\text{-}GRPH\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-obj-isoD*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}GRPH\ \alpha} \mathfrak{B}$

shows $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-obj-isoE*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}GRPH\ \alpha} \mathfrak{B}$
obtains $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-GRPH-obj-iso-iff*: $\mathfrak{A} \approx_{obj cat\text{-}GRPH\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

19 *SemiCAT*

19.1 Background

The methodology for the exposition of *SemiCAT* as a category is analogous to the one used in [8] for the exposition of *SemiCAT* as a semicategory.

named-theorems *cat-SemiCAT-simps*
named-theorems *cat-SemiCAT-intros*

19.2 Definition and elementary properties

definition *cat-SemiCAT* :: $V \Rightarrow V$
  where *cat-SemiCAT* α =
    [
      set $\{\mathfrak{C}.\ \textit{semicategory}\ \alpha\ \mathfrak{C}\}$,
      all-smcfs α,
      $(\lambda\mathfrak{F}\in_\circ \textit{all-smcfs}\ \alpha.\ \mathfrak{F}(\!|\textit{HomDom}|\!))$,
      $(\lambda\mathfrak{F}\in_\circ \textit{all-smcfs}\ \alpha.\ \mathfrak{F}(\!|\textit{HomCod}|\!))$,
      $(\lambda\mathfrak{GF}\in_\circ \textit{composable-arrs}\ (\textit{dg-SemiCAT}\ \alpha).\ \mathfrak{GF}(\!|0|\!) \circ_{SMCF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$,
      $(\lambda\mathfrak{C}\in_\circ \textit{set}\ \{\mathfrak{C}.\ \textit{semicategory}\ \alpha\ \mathfrak{C}\}.\ \textit{smcf-id}\ \mathfrak{C})$
    ]$_\circ$

Components.

lemma *cat-SemiCAT-components*:
  shows *cat-SemiCAT* $\alpha(\!|\textit{Obj}|\!)$ = *set* $\{\mathfrak{C}.\ \textit{semicategory}\ \alpha\ \mathfrak{C}\}$
    and *cat-SemiCAT* $\alpha(\!|\textit{Arr}|\!)$ = *all-smcfs* α
    and *cat-SemiCAT* $\alpha(\!|\textit{Dom}|\!)$ = $(\lambda\mathfrak{F}\in_\circ \textit{all-smcfs}\ \alpha.\ \mathfrak{F}(\!|\textit{HomDom}|\!))$
    and *cat-SemiCAT* $\alpha(\!|\textit{Cod}|\!)$ = $(\lambda\mathfrak{F}\in_\circ \textit{all-smcfs}\ \alpha.\ \mathfrak{F}(\!|\textit{HomCod}|\!))$
    and *cat-SemiCAT* $\alpha(\!|\textit{Comp}|\!)$ =
      $(\lambda\mathfrak{GF}\in_\circ \textit{composable-arrs}\ (\textit{dg-SemiCAT}\ \alpha).\ \mathfrak{GF}(\!|0|\!) \circ_{SMCF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
    and *cat-SemiCAT* $\alpha(\!|\textit{CId}|\!)$ = $(\lambda\mathfrak{C}\in_\circ \textit{set}\ \{\mathfrak{C}.\ \textit{semicategory}\ \alpha\ \mathfrak{C}\}.\ \textit{smcf-id}\ \mathfrak{C})$
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-SemiCAT*: *cat-smc* (*cat-SemiCAT* α) = *smc-SemiCAT* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-SemiCAT*, *unfolded slicing-simps*]:
  — Digraph
  cat-SemiCAT-ObjI = *smc-SemiCAT-ObjI*
  and *cat-SemiCAT-ObjD* = *smc-SemiCAT-ObjD*
  and *cat-SemiCAT-ObjE* = *smc-SemiCAT-ObjE*
  and *cat-SemiCAT-Obj-iff*[*cat-SemiCAT-simps*] = *smc-SemiCAT-Obj-iff*
  and *cat-SemiCAT-Dom-app*[*cat-SemiCAT-simps*] = *smc-SemiCAT-Dom-app*
  and *cat-SemiCAT-Cod-app*[*cat-SemiCAT-simps*] = *smc-SemiCAT-Cod-app*
  and *cat-SemiCAT-is-arrI* = *smc-SemiCAT-is-arrI*
  and *cat-SemiCAT-is-arrD* = *smc-SemiCAT-is-arrD*
  and *cat-SemiCAT-is-arrE* = *smc-SemiCAT-is-arrE*
  and *cat-SemiCAT-is-arr-iff*[*cat-SemiCAT-simps*] = *smc-SemiCAT-is-arr-iff*

lemmas-with [
    folded cat-smc-SemiCAT, *unfolded slicing-simps*, *unfolded cat-smc-SemiCAT*
    ]:
  — Semicategory
  cat-SemiCAT-Comp-vdomain = *smc-SemiCAT-Comp-vdomain*
  and *cat-SemiCAT-composable-arrs-dg-SemiCAT* =
    smc-SemiCAT-composable-arrs-dg-SemiCAT

and *cat-SemiCAT-Comp* = *smc-SemiCAT-Comp*
and *cat-SemiCAT-Comp-app*[*cat-SemiCAT-simps*] = *smc-SemiCAT-Comp-app*
and *cat-SemiCAT-Comp-vrange* = *smc-SemiCAT-Comp-vrange*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-SemiCAT*, *unfolded slicing-simps*]:
— Semicategory
cat-SemiCAT-obj-initialI = *smc-SemiCAT-obj-initialI*
and *cat-SemiCAT-obj-initialD* = *smc-SemiCAT-obj-initialD*
and *cat-SemiCAT-obj-initialE* = *smc-SemiCAT-obj-initialE*
and *cat-SemiCAT-obj-initial-iff*[*cat-SemiCAT-simps*] =
smc-SemiCAT-obj-initial-iff
and *cat-SemiCAT-obj-terminalI* = *smc-SemiCAT-obj-terminalI*
and *cat-SemiCAT-obj-terminalE* = *smc-SemiCAT-obj-terminalE*

19.3 Identity

lemma *cat-SemiCAT-CId-app*[*cat-SemiCAT-simps*]:
assumes *semicategory* α $\mathfrak{C}$
shows *cat-SemiCAT* $\alpha(\!|CId|\!)(\!|\mathfrak{C}|\!)$ = *smcf-id* $\mathfrak{C}$
⟨*proof*⟩

lemma *cat-SemiCAT-CId-vdomain*[*cat-SemiCAT-simps*]:
$\mathcal{D}_\circ$ (*cat-SemiCAT* $\alpha(\!|CId|\!)$) = *set* $\{\mathfrak{C}.\ semicategory\ \alpha\ \mathfrak{C}\}$
⟨*proof*⟩

lemma *cat-SemiCAT-CId-vrange*: $\mathcal{R}_\circ$ (*cat-SemiCAT* $\alpha(\!|CId|\!)$) $\subseteq_\circ$ *all-smcfs* α
⟨*proof*⟩

19.4 *SemiCAT* is a category

lemma (in $\mathcal{Z}$**)** *tiny-category-cat-SemiCAT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-category* β (*cat-SemiCAT* α)
⟨*proof*⟩

19.5 Isomorphism

lemma *cat-SemiCAT-is-iso-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-is-iso-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-is-iso-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-is-iso-arr-iff*[*cat-SemiCAT-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}SemiCAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

19.6 Isomorphic objects

lemma *cat-SemiCAT-obj-isoI*:

assumes $\mathfrak{A} \approx_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{obj cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-obj-isoD*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-obj-isoE*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}SemiCAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{A} \approx_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-SemiCAT-obj-iso-iff*[*cat-SemiCAT-simps*]:
$\mathfrak{A} \approx_{obj cat\text{-}SemiCAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{A} \approx_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

20 CAT as a digraph

20.1 Background

CAT is usually defined as a category of categories and functors (e.g., see Chapter I-2 in [7]). However, there is little that can prevent one from exposing CAT as a digraph and provide additional structure gradually in subsequent theories. Thus, in this section, α-CAT is defined as a digraph of categories and functors in the set V_α, and α-Cat is defined as a digraph of tiny categories and tiny functors in V_α.

named-theorems *dg-CAT-simps*
named-theorems *dg-CAT-intros*

20.2 Definition and elementary properties

definition *dg-CAT* :: $V \Rightarrow V$
  where *dg-CAT* α =
    [
      set $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$,
      all-cfs α,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomDom*$\rfloor))$,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomCod*$\rfloor))$
    ]$_\circ$

Components.

lemma *dg-CAT-components*:
  shows *dg-CAT* $\alpha(\lfloor$*Obj*$\rfloor)$ = *set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$
    and *dg-CAT* $\alpha(\lfloor$*Arr*$\rfloor)$ = *all-cfs* α
    and *dg-CAT* $\alpha(\lfloor$*Dom*$\rfloor)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomDom*$\rfloor))$
    and *dg-CAT* $\alpha(\lfloor$*Cod*$\rfloor)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomCod*$\rfloor))$
  ⟨*proof*⟩

20.3 Object

lemma *dg-CAT-ObjI*:
  assumes *category* α $\mathfrak{A}$
  shows $\mathfrak{A} \in_\circ$ *dg-CAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  ⟨*proof*⟩

lemma *dg-CAT-ObjD*:
  assumes $\mathfrak{A} \in_\circ$ *dg-CAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  shows *category* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-CAT-ObjE*:
  assumes $\mathfrak{A} \in_\circ$ *dg-CAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  obtains *category* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-CAT-Obj-iff*[*dg-CAT-simps*]: $\mathfrak{A} \in_\circ$ *dg-CAT* $\alpha(\lfloor$*Obj*$\rfloor) \longleftrightarrow$ *category* α $\mathfrak{A}$
  ⟨*proof*⟩

20.4 Domain and codomain

lemma [*dg-CAT-simps*]:
  assumes $\mathfrak{F} \in_\circ$ *all-cfs* α
  shows *dg-CAT-Dom-app*: *dg-CAT* $\alpha(\lfloor$*Dom*$\rfloor)(\lfloor\mathfrak{F}\rfloor)$ = $\mathfrak{F}(\lfloor$*HomDom*$\rfloor)$
    and *dg-CAT-Cod-app*: *dg-CAT* $\alpha(\lfloor$*Cod*$\rfloor)(\lfloor\mathfrak{F}\rfloor)$ = $\mathfrak{F}(\lfloor$*HomCod*$\rfloor)$
  ⟨*proof*⟩

20.5 CAT is a digraph

lemma (in $\mathcal{Z}$**)** *tiny-category-dg-CAT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$
shows *tiny-digraph* β (*dg-CAT* α)
⟨*proof*⟩

20.6 Arrow with a domain and a codomain

lemma *dg-CAT-is-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}CAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-CAT-is-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}CAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-CAT-is-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}CAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-CAT-is-arr-iff*[*dg-CAT-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}CAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

21 *CAT* as a semicategory

21.1 Background

The subsection presents the theory of the semicategories of α-categories. It continues the development that was initiated in section 20.

named-theorems *smc-CAT-simps*
named-theorems *smc-CAT-intros*

21.2 Definition and elementary properties

definition *smc-CAT* :: $V \Rightarrow V$
  where *smc-CAT* α =
    [
      set $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$,
      all-cfs α,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$,
      $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-CAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{CF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
    ]$_\circ$

Components.

lemma *smc-CAT-components*:
  shows *smc-CAT* $\alpha(\!|$*Obj*$|\!)$ = *set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$
    and *smc-CAT* $\alpha(\!|$*Arr*$|\!)$ = *all-cfs* α
    and *smc-CAT* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$
    and *smc-CAT* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$
    and *smc-CAT* $\alpha(\!|$*Comp*$|\!)$ = $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-CAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{CF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
  ⟨*proof*⟩

Slicing.

lemma *smc-dg-CAT*: *smc-dg* (*smc-CAT* α) = *dg-CAT* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-CAT*, *unfolded slicing-simps*]:
  smc-CAT-ObjI = *dg-CAT-ObjI*
  and *smc-CAT-ObjD* = *dg-CAT-ObjD*
  and *smc-CAT-ObjE* = *dg-CAT-ObjE*
  and *smc-CAT-Obj-iff*[*smc-CAT-simps*] = *dg-CAT-Obj-iff*
  and *smc-CAT-Dom-app*[*smc-CAT-simps*] = *dg-CAT-Dom-app*
  and *smc-CAT-Cod-app*[*smc-CAT-simps*] = *dg-CAT-Cod-app*
  and *smc-CAT-is-arrI* = *dg-CAT-is-arrI*
  and *smc-CAT-is-arrD* = *dg-CAT-is-arrD*
  and *smc-CAT-is-arrE* = *dg-CAT-is-arrE*
  and *smc-CAT-is-arr-iff*[*smc-CAT-simps*] = *dg-CAT-is-arr-iff*

21.3 Composable arrows

lemma *smc-CAT-composable-arrs-dg-CAT*:
  composable-arrs (*dg-CAT* α) = *composable-arrs* (*smc-CAT* α)
  ⟨*proof*⟩

lemma *smc-CAT-Comp*:
  smc-CAT $\alpha(\!|$*Comp*$|\!)$ = $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*smc-CAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{SMCF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
  ⟨*proof*⟩

21.4 Composition

lemma *smc-CAT-Comp-app*[*smc-CAT-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto_{smc\text{-}CAT\ \alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto_{smc\text{-}CAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{A\, smc\text{-}CAT\ \alpha} \mathfrak{F} = \mathfrak{G} \circ_{SMCF} \mathfrak{F}$
⟨*proof*⟩

lemma *smc-CAT-Comp-vdomain*: $\mathcal{D}_\circ\ (smc\text{-}CAT\ \alpha(\!|Comp|\!)) = composable\text{-}arrs\ (smc\text{-}CAT\ \alpha)$
⟨*proof*⟩

lemma *smc-CAT-Comp-vrange*: $\mathcal{R}_\circ\ (smc\text{-}CAT\ \alpha(\!|Comp|\!)) \subseteq_\circ all\text{-}cfs\ \alpha$
⟨*proof*⟩

21.5 CAT is a category

lemma (**in** $\mathcal{Z}$) *tiny-semicategory-smc-CAT*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *tiny-semicategory* β $(smc\text{-}CAT\ \alpha)$
⟨*proof*⟩

21.6 Initial object

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-initialI*: *obj-initial* $(smc\text{-}CAT\ \alpha)$ *cat-0*
— See [1][11]).
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-initialD*:
assumes *obj-initial* $(smc\text{-}CAT\ \alpha)\ \mathfrak{A}$
shows $\mathfrak{A} =$ *cat-0*
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-initialE*:
assumes *obj-initial* $(smc\text{-}CAT\ \alpha)\ \mathfrak{A}$
obtains $\mathfrak{A} =$ *cat-0*
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-initial-iff*[*smc-CAT-simps*]:
obj-initial $(smc\text{-}CAT\ \alpha)\ \mathfrak{A} \longleftrightarrow \mathfrak{A} =$ *cat-0*
⟨*proof*⟩

21.7 Terminal object

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-terminalI*:
— See [1][12].
assumes $a \in_\circ Vset\ \alpha$ **and** $f \in_\circ Vset\ \alpha$
shows *obj-terminal* $(smc\text{-}CAT\ \alpha)$ $(cat\text{-}1\ a\ f)$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-CAT-obj-terminalE*:
assumes *obj-terminal* $(smc\text{-}CAT\ \alpha)\ \mathfrak{B}$
obtains $a\ f$ **where** $a \in_\circ Vset\ \alpha$ **and** $f \in_\circ Vset\ \alpha$ **and** $\mathfrak{B} = cat\text{-}1\ a\ f$
⟨*proof*⟩

[11] https://ncatlab.org/nlab/show/initial+object
[12] https://ncatlab.org/nlab/show/terminal+object

22 *CAT*

22.1 Background

The subsection presents the theory of the categories of α-categories. It continues the development that was initiated in sections 20-21.

named-theorems *cat-CAT-simps*
named-theorems *cat-CAT-intros*

22.2 Definition and elementary properties

definition *cat-CAT* :: $V \Rightarrow V$
  where *cat-CAT* α =
    [
      set $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$,
      all-cfs α,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$,
      $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$,
      $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-CAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{CF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$,
      $(\lambda\mathfrak{C}\in_\circ$*set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}.$ *cf-id* $\mathfrak{C})$
    ]$_\circ$

Components.

lemma *cat-CAT-components*:
  shows *cat-CAT* $\alpha(\!|$*Obj*$|\!)$ = *set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$
    and *cat-CAT* $\alpha(\!|$*Arr*$|\!)$ = *all-cfs* α
    and *cat-CAT* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$
    and *cat-CAT* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-cfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$
    and *cat-CAT* $\alpha(\!|$*Comp*$|\!)$ =
      $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-CAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{CF} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
    and *cat-CAT* $\alpha(\!|$*CId*$|\!)$ = $(\lambda\mathfrak{C}\in_\circ$*set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}.$ *cf-id* $\mathfrak{C})$
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-CAT*: *cat-smc* (*cat-CAT* α) = *smc-CAT* α
⟨*proof*⟩

lemmas-with [*folded cat-smc-CAT*, *unfolded slicing-simps*]:
  — Digraph
  cat-CAT-ObjI = *smc-CAT-ObjI*
  and *cat-CAT-ObjD* = *smc-CAT-ObjD*
  and *cat-CAT-ObjE* = *smc-CAT-ObjE*
  and *cat-CAT-Obj-iff*[*cat-CAT-simps*] = *smc-CAT-Obj-iff*
  and *cat-CAT-Dom-app*[*cat-CAT-simps*] = *smc-CAT-Dom-app*
  and *cat-CAT-Cod-app*[*cat-CAT-simps*] = *smc-CAT-Cod-app*
  and *cat-CAT-is-arrI* = *smc-CAT-is-arrI*
  and *cat-CAT-is-arrD* = *smc-CAT-is-arrD*
  and *cat-CAT-is-arrE* = *smc-CAT-is-arrE*
  and *cat-CAT-is-arr-iff*[*cat-CAT-simps*] = *smc-CAT-is-arr-iff*

lemmas-with [*folded cat-smc-CAT*, *unfolded slicing-simps*, *unfolded cat-smc-CAT*]:
  — Semicategory
  cat-CAT-Comp-vdomain = *smc-CAT-Comp-vdomain*
  and *cat-CAT-composable-arrs-dg-CAT* = *smc-CAT-composable-arrs-dg-CAT*
  and *cat-CAT-Comp* = *smc-CAT-Comp*
  and *cat-CAT-Comp-app*[*cat-CAT-simps*] = *smc-CAT-Comp-app*
  and *cat-CAT-Comp-vrange* = *smc-CAT-Comp-vrange*

lemmas-with (in $\mathcal{Z}$**)** [*folded cat-smc-CAT*, *unfolded slicing-simps*]:
— Semicategory
cat-CAT-obj-initialI = *smc-CAT-obj-initialI*
and *cat-CAT-obj-initialD* = *smc-CAT-obj-initialD*
and *cat-CAT-obj-initialE* = *smc-CAT-obj-initialE*
and *cat-CAT-obj-initial-iff*[*cat-CAT-simps*] = *smc-CAT-obj-initial-iff*
and *cat-CAT-obj-terminalI* = *smc-CAT-obj-terminalI*
and *cat-CAT-obj-terminalE* = *smc-CAT-obj-terminalE*

22.3 Identity

lemma *cat-CAT-CId-app*[*cat-CAT-simps*]:
assumes *category* α $\mathfrak{C}$
shows *cat-CAT* $\alpha(\!|CId|\!)(\!|\mathfrak{C}|\!)$ = *cf-id* $\mathfrak{C}$
⟨*proof*⟩

lemma *cat-CAT-CId-vdomain*: $\mathcal{D}_\circ$ (*cat-CAT* $\alpha(\!|CId|\!)$) = *set* $\{\mathfrak{C}.$ *category* α $\mathfrak{C}\}$
⟨*proof*⟩

lemma *cat-CAT-CId-vrange*: $\mathcal{R}_\circ$ (*cat-CAT* $\alpha(\!|CId|\!)$) $\subseteq_\circ$ *all-cfs* α
⟨*proof*⟩

22.4 *CAT* is a category

lemma (in $\mathcal{Z}$**)** *tiny-category-cat-CAT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-category* β (*cat-CAT* α)
⟨*proof*⟩

lemmas [*cat-cs-intros*] = $\mathcal{Z}$*.tiny-category-cat-CAT*

22.5 Isomorphism

lemma *cat-CAT-is-iso-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C.iso\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}CAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-is-iso-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}CAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-is-iso-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}CAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-is-iso-arr-iff*[*cat-CAT-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{iso cat\text{-}CAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{C.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

22.6 Isomorphic objects

lemma *cat-CAT-obj-isoI*:
assumes $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{obj cat\text{-}CAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-obj-isoD*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}CAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-obj-isoE*:
assumes $\mathfrak{A} \approx_{obj cat\text{-}CAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *cat-CAT-obj-iso-iff*[*cat-CAT-simps*]:
$\mathfrak{A} \approx_{obj cat\text{-}CAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{A} \approx_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

23 *FUNCT* and *Funct* as digraphs

23.1 Background

A general reference for this section is Chapter II-4 in [7].

named-theorems *dg-FUNCT-cs-simps*
named-theorems *dg-FUNCT-cs-intros*
named-theorems *cat-map-cs-simps*
named-theorems *cat-map-cs-intros*
named-theorems *cat-map-extra-cs-simps*

23.2 Functor map

23.2.1 Definition and elementary properties

definition *cf-map* :: $V \Rightarrow V$
  where *cf-map* $\mathfrak{F}$ = $[\mathfrak{F}(\!|ObjMap|\!), \mathfrak{F}(\!|ArrMap|\!)]_\circ$

abbreviation *cf-maps* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$ ≡ *set* $\{$*cf-map* $\mathfrak{F}$ | $\mathfrak{F}$. $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$

abbreviation *tm-cf-maps* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$ ≡ *set* $\{$*cf-map* $\mathfrak{F}$ | $\mathfrak{F}$. $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$

lemma *tm-cf-maps-subset-cf-maps*:
  $\{$*cf-map* $\mathfrak{F}$ | $\mathfrak{F}$. $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\} \subseteq \{$*cf-map* $\mathfrak{F}$ | $\mathfrak{F}$. $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
  ⟨*proof*⟩

Components.

lemma *cf-map-components*[*cat-map-cs-simps*]:
  shows *cf-map* $\mathfrak{F}(\!|ObjMap|\!)$ = $\mathfrak{F}(\!|ObjMap|\!)$
    and *cf-map* $\mathfrak{F}(\!|ArrMap|\!)$ = $\mathfrak{F}(\!|ArrMap|\!)$
  ⟨*proof*⟩

Sequence characterization.

lemma *dg-FUNCT-Obj-components*:
  shows $[FOM, FAM]_\circ(\!|ObjMap|\!)$ = FOM
    and $[FOM, FAM]_\circ(\!|ArrMap|\!)$ = FAM
  ⟨*proof*⟩

lemma *cf-map-vfsequence*[*cat-map-cs-intros*]: *vfsequence* (*cf-map* $\mathfrak{F}$)
  ⟨*proof*⟩

lemma *cf-map-vdomain*[*cat-map-cs-simps*]: $\mathcal{D}_\circ$ (*cf-map* $\mathfrak{F}$) = $2_{\mathbb{N}}$
  ⟨*proof*⟩

lemma (**in** *is-functor*) *cf-map-vsubset-cf*: *cf-map* $\mathfrak{F} \subseteq_\circ \mathfrak{F}$
  ⟨*proof*⟩

Size.

lemma (**in** *is-functor*) *cf-map-ObjMap-in-Vset*:
  assumes $\alpha \in_\circ \beta$
  shows *cf-map* $\mathfrak{F}(\!|ObjMap|\!) \in_\circ Vset\ \beta$
  ⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-map-ObjMap-in-Vset*: *cf-map* $\mathfrak{F}(\!|ObjMap|\!) \in_\circ Vset\ \alpha$
  ⟨*proof*⟩

lemma (**in** *is-functor*) *cf-map-ArrMap-in-Vset*:
assumes $\alpha \in_\circ \beta$
shows *cf-map* $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-map-ArrMap-in-Vset*: *cf-map* $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-map-in-Vset-4*: *cf-map* $\mathfrak{F} \in_\circ Vset\ (\alpha + 4_{\mathbb{N}})$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-map-in-Vset*: *cf-map* $\mathfrak{F} \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-map-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *cf-map* $\mathfrak{F} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma *cf-maps-subset-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $\{cf\text{-}map\ \mathfrak{F} \mid \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\} \subseteq elts\ (Vset\ \beta)$
⟨*proof*⟩

lemma *small-cf-maps*[*simp*]: *small* $\{cf\text{-}map\ \mathfrak{F} \mid \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-tm-cf-maps*[*simp*]: *small* $\{cf\text{-}map\ \mathfrak{F} \mid \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *cf-maps-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B} \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *tm-cf-maps-vsubset-Vset*: *tm-cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B} \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

Rules.

lemma (**in** *is-functor*) *cf-mapsI*: *cf-map* $\mathfrak{F} \in_\circ$ *cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-mapsI*: *cf-map* $\mathfrak{F} \in_\circ$ *tm-cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-mapsI'*:
assumes $\mathfrak{F}' =$ *cf-map* $\mathfrak{F}$
shows $\mathfrak{F}' \in_\circ$ *cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-mapsI'*:
assumes $\mathfrak{F}' =$ *cf-map* $\mathfrak{F}$
shows $\mathfrak{F}' \in_\circ$ *tm-cf-maps* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-map-cs-intros*] =
is-functor.cf-mapsI

lemmas *cf-mapsI'*[*cat-map-cs-intros*] =
is-functor.cf-mapsI'[*rotated*]

lemmas [*cat-map-cs-intros*] =
is-tm-functor.tm-cf-mapsI

lemmas *tm-cf-mapsI'*[*cat-map-cs-intros*] =
is-tm-functor.tm-cf-mapsI'[*rotated*]

lemma *cf-mapsE*[*elim*]:
assumes $\mathfrak{F} \in_{\circ}$ *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
obtains $\mathfrak{G}$ **where** $\mathfrak{F}$ = *cf-map* $\mathfrak{G}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *tm-cf-mapsE*[*elim*]:
assumes $\mathfrak{F} \in_{\circ}$ *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
obtains $\mathfrak{G}$ **where** $\mathfrak{F}$ = *cf-map* $\mathfrak{G}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

The opposite functor map.

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

lemmas [*cat-op-simps*] = *is-functor.cf-map-op-cf*

Elementary properties.

lemma *tm-cf-maps-vsubset-cf-maps*: *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$ $\subseteq_{\circ}$ *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *tm-cf-maps-in-cf-maps*:
assumes $\mathfrak{F} \in_{\circ}$ *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
shows $\mathfrak{F} \in_{\circ}$ *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *cf-map-inj*:
assumes *cf-map* $\mathfrak{F}$ = *cf-map* $\mathfrak{G}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{F} = \mathfrak{G}$
⟨*proof*⟩

lemma *cf-map-eq-iff*[*cat-map-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *cf-map* $\mathfrak{F}$ = *cf-map* $\mathfrak{G} \longleftrightarrow \mathfrak{F} = \mathfrak{G}$
⟨*proof*⟩

lemma *cf-map-eqI*:
assumes $\mathfrak{F} \in_{\circ}$ *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
and $\mathfrak{G} \in_{\circ}$ *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
and $\mathfrak{F}(\lfloor ObjMap \rfloor) = \mathfrak{G}(\lfloor ObjMap \rfloor)$
and $\mathfrak{F}(\lfloor ArrMap \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)$
shows $\mathfrak{F} = \mathfrak{G}$
⟨*proof*⟩

23.3 Conversion of a functor map to a functor

23.3.1 Definition and elementary properties

definition *cf-of-cf-map* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ = $[\mathfrak{F}(\!|ObjMap|\!), \mathfrak{F}(\!|ArrMap|\!), \mathfrak{A}, \mathfrak{B}]_\circ$

Components.

lemma *cf-of-cf-map-components*:
shows *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ObjMap|\!)$ = $\mathfrak{F}(\!|ObjMap|\!)$
and *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|ArrMap|\!)$ = $\mathfrak{F}(\!|ArrMap|\!)$
and *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|HomDom|\!)$ = $\mathfrak{A}$
and *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}(\!|HomCod|\!)$ = $\mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-map-extra-cs-simps*] = *cf-of-cf-map-components*(*1–2*)
lemmas [*cat-map-cs-simps*] = *cf-of-cf-map-components*(*3–4*)

23.3.2 The conversion of a functor map to a functor is a functor

lemma (**in** *is-functor*) *cf-of-cf-map-is-functor*:
cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$) : $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-of-cf-map-is-functor′*:
assumes $\mathfrak{F}'$ = *cf-map* $\mathfrak{F}$
and $\mathfrak{A}'$ = $\mathfrak{A}$
and $\mathfrak{B}'$ = $\mathfrak{B}$
shows *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}'$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-map-cs-intros*] = *is-functor.cf-of-cf-map-is-functor′*

23.3.3 The value of the conversion of a functor map to a functor

lemma (**in** *is-functor*) *cf-of-cf-map-of-cf-map*[*cat-map-cs-simps*]:
cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$) = $\mathfrak{F}$
⟨*proof*⟩

lemmas [*cat-map-cs-simps*] = *is-functor.cf-of-cf-map-of-cf-map*

23.4 Natural transformation arrow

23.4.1 Definition and elementary properties

definition *ntcf-arrow* :: $V \Rightarrow V$
where *ntcf-arrow* $\mathfrak{N}$ = $[\mathfrak{N}(\!|NTMap|\!)$, *cf-map* $(\mathfrak{N}(\!|NTDom|\!))$, *cf-map* $(\mathfrak{N}(\!|NTCod|\!))]_\circ$

abbreviation *ntcf-arrows* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$ ≡
set $\{$*ntcf-arrow* $\mathfrak{N}$ | $\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}\}$

abbreviation *tm-ntcf-arrows* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$ ≡
set $\{$*ntcf-arrow* $\mathfrak{N}$ | $\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.tm\alpha}$ $\mathfrak{B}\}$

lemma *tm-ntcf-arrows-subset-ntcf-arrows*:
$\{$*ntcf-arrow* $\mathfrak{N}$ | $\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C.tm\alpha}$ $\mathfrak{B}\}$ ⊆
$\{$*ntcf-arrow* $\mathfrak{N}$ | $\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}\}$
⟨*proof*⟩

Components.

lemma *ntcf-arrow-components*:

shows [*cat-map-cs-simps*]: *ntcf-arrow* $\mathfrak{N}(\!|NTMap|\!) = \mathfrak{N}(\!|NTMap|\!)$
and *ntcf-arrow* $\mathfrak{N}(\!|NTDom|\!) = cf\text{-}map\ (\mathfrak{N}(\!|NTDom|\!))$
and *ntcf-arrow* $\mathfrak{N}(\!|NTCod|\!) = cf\text{-}map\ (\mathfrak{N}(\!|NTCod|\!))$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-arrow-components'*:
shows *ntcf-arrow* $\mathfrak{N}(\!|NTMap|\!) = \mathfrak{N}(\!|NTMap|\!)$
and *ntcf-arrow* $\mathfrak{N}(\!|NTDom|\!) = cf\text{-}map\ \mathfrak{F}$
and *ntcf-arrow* $\mathfrak{N}(\!|NTCod|\!) = cf\text{-}map\ \mathfrak{G}$
⟨*proof*⟩

lemmas [*cat-map-cs-simps*] = *is-ntcf.ntcf-arrow-components'*(*2*,*3*)

Elementary properties.

lemma *dg-FUNCT-Arr-components*:
shows $[NTM,\ NTD,\ NTC]_{\circ}(\!|NTMap|\!) = NTM$
and $[NTM,\ NTD,\ NTC]_{\circ}(\!|NTDom|\!) = NTD$
and $[NTM,\ NTD,\ NTC]_{\circ}(\!|NTCod|\!) = NTC$
⟨*proof*⟩

lemma *ntcf-arrow-vfsequence*[*cat-map-cs-intros*]: *vfsequence* (*ntcf-arrow* $\mathfrak{N}$)
⟨*proof*⟩

lemma *ntcf-arrow-vdomain*[*cat-map-cs-simps*]: $\mathcal{D}_{\circ}$ (*ntcf-arrow* $\mathfrak{N}$) = $3_{\mathbf{N}}$
⟨*proof*⟩

Size.

lemma (**in** *is-ntcf*) *ntcf-arrow-NTMap-in-Vset*:
assumes $\alpha \in_{\circ} \beta$
shows *ntcf-arrow* $\mathfrak{N}(\!|NTMap|\!) \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-tm-ntcf*) *tm-ntcf-arrow-NTMap-in-Vset*:
ntcf-arrow $\mathfrak{N}(\!|NTMap|\!) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-arrow-NTDom-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows *ntcf-arrow* $\mathfrak{N}(\!|NTDom|\!) \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-tm-ntcf*) *tm-ntcf-arrow-NTDom-in-Vset*:
ntcf-arrow $\mathfrak{N}(\!|NTDom|\!) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-arrow-NTCod-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows *ntcf-arrow* $\mathfrak{N}(\!|NTCod|\!) \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-tm-ntcf*) *tm-ntcf-arrow-NTCod-in-Vset*:
ntcf-arrow $\mathfrak{N}(\!|NTCod|\!) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-arrow-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows *ntcf-arrow* $\mathfrak{N} \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (in *is-tm-ntcf*) *tm-ntcf-arrow-in-Vset*: *ntcf-arrow* $\mathfrak{N} \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma *ntcf-arrows-subset-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows
$\{ntcf\text{-}arrow\ \mathfrak{N} \mid \mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\} \subseteq elts\ (Vset\ \beta)$
⟨*proof*⟩

lemma *tm-ntcf-arrows-subset-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows
$\{ntcf\text{-}arrow\ \mathfrak{N} \mid \mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\} \subseteq$
$elts\ (Vset\ \beta)$
⟨*proof*⟩

lemma *small-ntcf-arrows*[*simp*]:
$small\ \{ntcf\text{-}arrow\ \mathfrak{N} \mid \mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma *small-tm-ntcf-arrows*[*simp*]:
$small\ \{ntcf\text{-}arrow\ \mathfrak{N} \mid \mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

lemma (in *is-ntcf*) *ntcf-arrow-in-Vset-7*: *ntcf-arrow* $\mathfrak{N} \in_{\circ} Vset\ (\alpha + \gamma_{\mathbb{N}})$
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *ntcf-arrows-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
shows $ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B} \in_{\circ} Vset\ \beta$
⟨*proof*⟩

lemma (in $\mathcal{Z}$) *tm-ntcf-arrows-vsubset-Vset*: $tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B} \subseteq_{\circ} Vset\ \alpha$
⟨*proof*⟩

Rules.

lemma (in *is-ntcf*) *ntcf-arrowsI*: *ntcf-arrow* $\mathfrak{N} \in_{\circ} ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tm-ntcf*) *tm-ntcf-arrowsI*: *ntcf-arrow* $\mathfrak{N} \in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (in *is-ntcf*) *ntcf-arrowsI'*:
assumes $\mathfrak{N}' = ntcf\text{-}arrow\ \mathfrak{N}$
shows $\mathfrak{N}' \in_{\circ} ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tm-ntcf*) *tm-ntcf-arrowsI'*:
assumes $\mathfrak{N}' = ntcf\text{-}arrow\ \mathfrak{N}$
shows $\mathfrak{N}' \in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-map-cs-intros*] =
is-ntcf.ntcf-arrowsI

lemmas *ntcf-arrowsI'*[*cat-map-cs-intros*] =
is-ntcf.ntcf-arrowsI'[*rotated*]

lemmas [*cat-map-cs-intros*] =
is-tm-ntcf.tm-ntcf-arrowsI

lemmas *tm-ntcf-arrowsI'*[*cat-map-cs-intros*] =
is-tm-ntcf.tm-ntcf-arrowsI'[*rotated*]

lemma *ntcf-arrowsE*[*elim*]:
assumes $\mathfrak{N} \in_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
obtains $\mathfrak{M}$ $\mathfrak{F}$ $\mathfrak{G}$ **where** $\mathfrak{N}$ = *ntcf-arrow* $\mathfrak{M}$ **and** $\mathfrak{M} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *tm-ntcf-arrowsE*[*elim*]:
assumes $\mathfrak{N} \in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
obtains $\mathfrak{M}$ $\mathfrak{F}$ $\mathfrak{G}$ **where** $\mathfrak{N}$ = *ntcf-arrow* $\mathfrak{M}$
and $\mathfrak{M} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

Elementary properties.

lemma *tm-ntcf-arrows-vsubset-ntcf-arrows*:
tm-ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B} \subseteq_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *tm-ntcf-arrows-in-cf-arrows*[*cat-map-cs-intros*]:
assumes $\mathfrak{N} \in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
shows $\mathfrak{N} \in_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-arrow-inj*:
assumes *ntcf-arrow* $\mathfrak{M}$ = *ntcf-arrow* $\mathfrak{N}$
and $\mathfrak{M} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{M} = \mathfrak{N}$
⟨*proof*⟩

lemma *ntcf-arrow-eq-iff*[*cat-map-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{M}$ = *ntcf-arrow* $\mathfrak{N} \longleftrightarrow \mathfrak{M} = \mathfrak{N}$
⟨*proof*⟩

lemma *ntcf-arrow-eqI*:
assumes $\mathfrak{M} \in_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
and $\mathfrak{N} \in_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
and $\mathfrak{M}(\lfloor NTMap \rfloor) = \mathfrak{N}(\lfloor NTMap \rfloor)$
and $\mathfrak{M}(\lfloor NTDom \rfloor) = \mathfrak{N}(\lfloor NTDom \rfloor)$
and $\mathfrak{M}(\lfloor NTCod \rfloor) = \mathfrak{N}(\lfloor NTCod \rfloor)$
shows $\mathfrak{M} = \mathfrak{N}$
⟨*proof*⟩

23.5 Conversion of a natural transformation arrow to a natural transformation

23.5.1 Definition and elementary properties

definition *ntcf-of-ntcf-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$ =
[
$\mathfrak{N}(\lfloor NTMap \rfloor)$,

cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}$(|*NTDom*|)),
cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}$(|*NTCod*|)),
$\mathfrak{A}$,
$\mathfrak{B}$
]$_\circ$

Components.

lemma *ntcf-of-ntcf-arrow-components*:
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$(|*NTMap*|) = $\mathfrak{N}$(|*NTMap*|)
and *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$(|*NTDom*|) = *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}$(|*NTDom*|))
and *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$(|*NTCod*|) = *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{N}$(|*NTCod*|))
and *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$(|*NTDGDom*|) = $\mathfrak{A}$
and *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$(|*NTDGCod*|) = $\mathfrak{B}$
⟨*proof*⟩

lemmas [*cat-map-extra-cs-simps*] = *ntcf-of-ntcf-arrow-components*(*1*)
lemmas [*cat-map-cs-simps*] = *ntcf-of-ntcf-arrow-components*(*2–5*)

23.5.2 The conversion of a natural transformation arrow to a natural transformation is a natural transformation

lemma (**in** *is-ntcf*) *ntcf-of-ntcf-arrow-is-ntcf*:
ntcf-of-ntcf-arrow $\mathfrak{A}$ $\mathfrak{B}$ (*ntcf-arrow* $\mathfrak{N}$) : $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-ntcf*) *ntcf-of-ntcf-arrow-is-ntcf′*:
assumes $\mathfrak{N}'$ = *ntcf-arrow* $\mathfrak{N}$ **and** $\mathfrak{A}'$ = $\mathfrak{A}$ **and** $\mathfrak{B}'$ = $\mathfrak{B}$
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}'$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-map-cs-intros*] = *is-ntcf.ntcf-of-ntcf-arrow-is-ntcf′*

23.5.3 The composition of the conversion of a natural transformation arrow to a natural transformation

lemma (**in** *is-ntcf*) *ntcf-of-ntcf-arrow*[*cat-map-cs-simps*]:
ntcf-of-ntcf-arrow $\mathfrak{A}$ $\mathfrak{B}$ (*ntcf-arrow* $\mathfrak{N}$) = $\mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-map-cs-simps*] = *is-ntcf.ntcf-of-ntcf-arrow*

23.6 Composition of the natural transformation arrows

definition *ntcf-arrow-vcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-arrow-vcomp* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{M}$ $\mathfrak{N}$ =
ntcf-arrow (*ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{M}$ $\cdot_{NTCF}$ *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$)

syntax *-ntcf-arrow-vcomp* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
(‹(-/ $\cdot_{NTCF}$-,- -)› [*55*, *56*, *57*, *58*] *55*)
syntax-consts *-ntcf-arrow-vcomp* ⇌ *ntcf-arrow-vcomp*
translations $\mathfrak{M}$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ $\mathfrak{N}$ ⇌ *CONST ntcf-arrow-vcomp* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{M}$ $\mathfrak{N}$

Components.

lemma (**in** *is-ntcf*) *ntcf-arrow-vcomp-components*:
(*ntcf-arrow* $\mathfrak{M}$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ *ntcf-arrow* $\mathfrak{N}$)(|*NTMap*|) = ($\mathfrak{M}$ $\cdot_{NTCF}$ $\mathfrak{N}$)(|*NTMap*|)
(*ntcf-arrow* $\mathfrak{M}$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ *ntcf-arrow* $\mathfrak{N}$)(|*NTDom*|) = *cf-map* (($\mathfrak{M}$ $\cdot_{NTCF}$ $\mathfrak{N}$)(|*NTDom*|))
(*ntcf-arrow* $\mathfrak{N}$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ *ntcf-arrow* $\mathfrak{M}$)(|*NTCod*|) = *cf-map* (($\mathfrak{N}$ $\cdot_{NTCF}$ $\mathfrak{M}$)(|*NTCod*|))
⟨*proof*⟩

lemmas [*cat-map-cs-simps*] = *is-ntcf.ntcf-arrow-vcomp-components*

Elementary properties.

lemma *ntcf-arrow-vcomp-ntcf-vcomp*[*cat-map-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{M}$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ *ntcf-arrow* $\mathfrak{N}$ = *ntcf-arrow* ($\mathfrak{M}$ $\cdot_{NTCF}$ $\mathfrak{N}$)
⟨*proof*⟩

23.7 Identity natural transformation arrow

definition *ntcf-arrow-id* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ = *ntcf-arrow* (*ntcf-id* (*cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$))

Components.

lemma (**in** *is-functor*) *ntcf-arrow-id-components*:
(*ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$))(⌊*NTMap*⌋) = *ntcf-id* $\mathfrak{F}$(⌊*NTMap*⌋)
(*ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$))(⌊*NTDom*⌋) = *cf-map* (*ntcf-id* $\mathfrak{F}$(⌊*NTDom*⌋))
(*ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$))(⌊*NTCod*⌋) = *cf-map* (*ntcf-id* $\mathfrak{F}$(⌊*NTCod*⌋))
⟨*proof*⟩

lemmas [*cat-map-cs-simps*] = *is-functor.ntcf-arrow-id-components*

Identity natural transformation arrow is a natural transformation arrow.

lemma *ntcf-arrow-id-ntcf-id*[*cat-map-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ (*cf-map* $\mathfrak{F}$) = *ntcf-arrow* (*ntcf-id* $\mathfrak{F}$)
⟨*proof*⟩

23.8 *FUNCT*

23.8.1 Definition and elementary properties

definition *dg-FUNCT* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$ =
[
cf-maps α $\mathfrak{A}$ $\mathfrak{B}$,
ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B}$,
($\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(⌊*NTDom*⌋)),
($\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(⌊*NTCod*⌋))
]$_\circ$

lemmas [*dg-FUNCT-cs-simps*] = *cat-map-cs-simps*
lemmas [*dg-FUNCT-cs-intros*] = *cat-map-cs-intros*

Components.

lemma *dg-FUNCT-components*:
shows *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(⌊*Obj*⌋) = *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
and *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(⌊*Arr*⌋) = *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
and *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(⌊*Dom*⌋) = ($\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(⌊*NTDom*⌋))
and *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(⌊*Cod*⌋) = ($\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(⌊*NTCod*⌋))
⟨*proof*⟩

23.8.2 Objects

lemma (**in** *is-functor*) *dg-FUNCT-ObjI*: *cf-map* $\mathfrak{F}$ $\in_\circ$ *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(⌊*Obj*⌋)
⟨*proof*⟩

23.8.3 Domain and codomain

mk-VLambda *dg-FUNCT-components*(*3*)
|*vsv dg-FUNCT-Dom-vsv*[*dg-FUNCT-cs-intros*]|
|*vdomain dg-FUNCT-Dom-vdomain*[*dg-FUNCT-cs-simps*]|

mk-VLambda *dg-FUNCT-components*(*4*)
|*vsv dg-FUNCT-Cod-vsv*[*dg-FUNCT-cs-intros*]|
|*vdomain dg-FUNCT-Cod-vdomain*[*dg-FUNCT-cs-simps*]|

lemma (**in** *is-ntcf*)
shows *dg-FUNCT-Dom-app*: *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Dom*|)(|*ntcf-arrow* $\mathfrak{N}$|) = *cf-map* $\mathfrak{F}$
and *dg-FUNCT-Cod-app*: *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Cod*|)(|*ntcf-arrow* $\mathfrak{N}$|) = *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemma (**in** *is-ntcf*)
assumes $\mathfrak{N}'$ = *ntcf-arrow* $\mathfrak{N}$
shows *dg-FUNCT-Dom-app'*: *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Dom*|)(|$\mathfrak{N}'$|) = *cf-map* $\mathfrak{F}$
and *dg-FUNCT-Cod-app'*: *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Cod*|)(|$\mathfrak{N}'$|) = *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemmas [*dg-FUNCT-cs-simps*] =
is-ntcf.dg-FUNCT-Dom-app'
is-ntcf.dg-FUNCT-Cod-app'

lemma
shows *dg-FUNCT-Dom-vrange*: $\mathcal{R}_\circ$ (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Dom*|)) $\subseteq_\circ$ *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Obj*|)
and *dg-FUNCT-Cod-vrange*: $\mathcal{R}_\circ$ (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Cod*|)) $\subseteq_\circ$ *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Obj*|)
⟨*proof*⟩

23.8.4 *FUNCT* is a tiny digraph

lemma (**in** $\mathcal{Z}$) *tiny-digraph-dg-FUNCT*:
assumes $\mathcal{Z}$ β **and** α $\in_\circ$ β
shows *tiny-digraph* β (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

23.8.5 Arrow with a domain and a codomain

lemma *dg-FUNCT-is-arrI*:
assumes $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{N}$: *cf-map* $\mathfrak{F}$ $\mapsto_{dg\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}}$ *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemma *dg-FUNCT-is-arrI'*:
assumes $\mathfrak{N}'$ = *ntcf-arrow* $\mathfrak{N}$
and $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
and $\mathfrak{F}'$ = *cf-map* $\mathfrak{F}$
and $\mathfrak{G}'$ = *cf-map* $\mathfrak{G}$
shows $\mathfrak{N}'$: $\mathfrak{F}'$ $\mapsto_{dg\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}}$ $\mathfrak{G}'$
⟨*proof*⟩

lemmas [*dg-FUNCT-cs-intros*] = *dg-FUNCT-is-arrI'*

lemma *dg-FUNCT-is-arrD*[*dest*]:
assumes $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{dg\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}}$ $\mathfrak{G}$
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$:
cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mapsto_{CF}$ *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{C\alpha}$ $\mathfrak{B}$
and $\mathfrak{N}$ = *ntcf-arrow* (*ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$)

and $\mathfrak{F}$ = *cf-map* (*cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$)
and $\mathfrak{G}$ = *cf-map* (*cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$)
⟨*proof*⟩

lemma *dg-FUNCT-is-arrE*[*elim*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{dg\text{-}FUNCT}\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G}$
obtains $\mathfrak{N}'\ \mathfrak{F}'\ \mathfrak{G}'$
where $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}$ = *ntcf-arrow* $\mathfrak{N}'$
and $\mathfrak{F}$ = *cf-map* $\mathfrak{F}'$
and $\mathfrak{G}$ = *cf-map* $\mathfrak{G}'$
⟨*proof*⟩

23.9 *Funct*

23.9.1 Definition and elementary properties

definition *dg-Funct* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$ =
[
tm-cf-maps α $\mathfrak{A}$ $\mathfrak{B}$,
tm-ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B}$,
$(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTDom|\!))$,
$(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTCod|\!))$
]$_{\circ}$

Components.

lemma *dg-Funct-components*:
shows *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Obj|\!)$ = *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
and *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Arr|\!)$ = *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
and *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Dom|\!)$ = $(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTDom|\!))$
and *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Cod|\!)$ = $(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTCod|\!))$
⟨*proof*⟩

23.9.2 Objects

lemma (**in** *is-tm-functor*) *dg-Funct-ObjI*: *cf-map* $\mathfrak{F} \in_{\circ}$ *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

23.9.3 Domain and codomain

mk-VLambda *dg-Funct-components*(*3*)
|*vsv dg-Funct-Dom-vsv*[*dg-FUNCT-cs-intros*]|
|*vdomain dg-Funct-Dom-vdomain*[*dg-FUNCT-cs-simps*]|

mk-VLambda *dg-Funct-components*(*4*)
|*vsv dg-Funct-Cod-vsv*[*dg-FUNCT-cs-intros*]|
|*vdomain dg-Funct-Cod-vdomain*[*dg-FUNCT-cs-simps*]|

lemma (**in** *is-tm-ntcf*)
shows *dg-Funct-Dom-app*: *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Dom|\!)(\!|ntcf\text{-}arrow\ \mathfrak{N}|\!)$ = *cf-map* $\mathfrak{F}$
and *dg-Funct-Cod-app*: *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Cod|\!)(\!|ntcf\text{-}arrow\ \mathfrak{N}|\!)$ = *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemma (**in** *is-tm-ntcf*)
assumes $\mathfrak{N}'$ = *ntcf-arrow* $\mathfrak{N}$
shows *dg-Funct-Dom-app'*: *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Dom|\!)(\!|\mathfrak{N}'|\!)$ = *cf-map* $\mathfrak{F}$
and *dg-Funct-Cod-app'*: *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Cod|\!)(\!|\mathfrak{N}'|\!)$ = *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemmas [*dg-FUNCT-cs-simps*] =
is-tm-ntcf.dg-Funct-Dom-app′
is-tm-ntcf.dg-Funct-Cod-app′

lemma
shows *dg-Funct-Dom-vrange*: $\mathcal{R}_{\circ}$ (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Dom|\!)$) $\subseteq_{\circ}$ *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Obj|\!)$
and *dg-Funct-Cod-vrange*: $\mathcal{R}_{\circ}$ (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Cod|\!)$) $\subseteq_{\circ}$ *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

23.9.4 Arrow with a domain and a codomain

lemma *dg-Funct-is-arrI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{N}$: *cf-map* $\mathfrak{F} \mapsto_{dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}}$ *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemma *dg-Funct-is-arrI′*:
assumes $\mathfrak{N}'$ = *ntcf-arrow* $\mathfrak{N}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}'$ = *cf-map* $\mathfrak{F}$
and $\mathfrak{G}'$ = *cf-map* $\mathfrak{G}$
shows $\mathfrak{N}' : \mathfrak{F}' \mapsto_{dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{G}'$
⟨*proof*⟩

lemmas [*dg-FUNCT-cs-intros*] = *dg-Funct-is-arrI′*

lemma *dg-Funct-is-arrD*[*dest*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{G}$
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$:
cf-of-cf-map $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F} \mapsto_{CF.tm}$ *cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N}$ = *ntcf-arrow* (*ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{N}$)
and $\mathfrak{F}$ = *cf-map* (*cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$)
and $\mathfrak{G}$ = *cf-map* (*cf-of-cf-map* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$)
⟨*proof*⟩

lemma *dg-Funct-is-arrE*[*elim*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{G}$
obtains $\mathfrak{N}'$ $\mathfrak{F}'$ $\mathfrak{G}'$ **where** $\mathfrak{N}' : \mathfrak{F}' \mapsto_{CF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N}$ = *ntcf-arrow* $\mathfrak{N}'$
and $\mathfrak{F}$ = *cf-map* $\mathfrak{F}'$
and $\mathfrak{G}$ = *cf-map* $\mathfrak{G}'$
⟨*proof*⟩

23.9.5 *Funct* is a digraph

lemma *digraph-dg-Funct*:
assumes *tiny-category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$
shows *digraph* α (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

23.9.6 *Funct* is a subdigraph of *FUNCT*

lemma *subdigraph-dg-Funct-dg-FUNCT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** *tiny-category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$
shows *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$ $\subseteq_{DG\beta}$ *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

24 *FUNCT* and *Funct* as semicategories

24.1 Background

The subsection presents the theory of the semicategories of α-functors between two α-categories. It continues the development that was initiated in section 23. A general reference for this section is Chapter II-4 in [7].

named-theorems *smc-FUNCT-cs-simps*
named-theorems *smc-FUNCT-cs-intros*

lemmas [*smc-FUNCT-cs-simps*] = *cat-map-cs-simps*
lemmas [*smc-FUNCT-cs-intros*] = *cat-map-cs-intros*

24.2 *FUNCT*

24.2.1 Definition and elementary properties

definition *smc-FUNCT* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$ =
    [
      cf-maps α $\mathfrak{A}$ $\mathfrak{B}$,
      ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B}$,
      $(\lambda\mathfrak{N}\in_{\circ}$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTDom \rfloor))$,
      $(\lambda\mathfrak{N}\in_{\circ}$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTCod \rfloor))$,
      $(\lambda\mathfrak{MN}\in_{\circ}$*composable-arrs* (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}(\lfloor 0 \rfloor) \cdot_{NTCF_{\mathfrak{A},\mathfrak{B}}} \mathfrak{MN}(\lfloor 1_{\mathbb{N}} \rfloor))$
    ]$_{\circ}$

Components.

lemma *smc-FUNCT-components*:
  shows *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Obj \rfloor)$ = *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
    and *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Arr \rfloor)$ = *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
    and *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Dom \rfloor)$ = $(\lambda\mathfrak{N}\in_{\circ}$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTDom \rfloor))$
    and *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Cod \rfloor)$ = $(\lambda\mathfrak{N}\in_{\circ}$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTCod \rfloor))$
    and *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Comp \rfloor)$ =
      $(\lambda\mathfrak{MN}\in_{\circ}$*composable-arrs* (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}(\lfloor 0 \rfloor) \cdot_{NTCF_{\mathfrak{A},\mathfrak{B}}} \mathfrak{MN}(\lfloor 1_{\mathbb{N}} \rfloor))$
  ⟨*proof*⟩

Slicing.

lemma *smc-dg-FUNCT*: *smc-dg* (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) = *dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

context *is-ntcf*
begin

lemmas-with [*folded smc-dg-FUNCT*, *unfolded slicing-simps*]:
  smc-FUNCT-Dom-app = *dg-FUNCT-Dom-app*
  and *smc-FUNCT-Cod-app* = *dg-FUNCT-Cod-app*

end

lemmas [*smc-FUNCT-cs-simps*] =
  is-ntcf.smc-FUNCT-Dom-app
  is-ntcf.smc-FUNCT-Cod-app

lemmas-with [*folded smc-dg-FUNCT*, *unfolded slicing-simps*]:
  smc-FUNCT-Dom-vsv[*intro*] = *dg-FUNCT-Dom-vsv*
  and *smc-FUNCT-Dom-vdomain*[*smc-FUNCT-cs-simps*] = *dg-FUNCT-Dom-vdomain*

and *smc-FUNCT-Cod-vsv*[*intro*] = *dg-FUNCT-Cod-vsv*
and *smc-FUNCT-Cod-vdomain*[*smc-FUNCT-cs-simps*] = *dg-FUNCT-Cod-vdomain*
and *smc-FUNCT-Dom-vrange* = *dg-FUNCT-Dom-vrange*
and *smc-FUNCT-Cod-vrange* = *dg-FUNCT-Cod-vrange*
and *smc-FUNCT-is-arrI* = *dg-FUNCT-is-arrI*
and *smc-FUNCT-is-arrI'*[*smc-FUNCT-cs-intros*] = *dg-FUNCT-is-arrI'*
and *smc-FUNCT-is-arrD* = *dg-FUNCT-is-arrD*
and *smc-FUNCT-is-arrE*[*elim*] = *dg-FUNCT-is-arrE*

24.2.2 Composable arrows

lemma *smc-FUNCT-composable-arrs-dg-FUNCT*:
composable-arrs (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) = *composable-arrs* (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *smc-FUNCT-Comp*:
smc-FUNCT α $\mathfrak{A}$ $\mathfrak{B}$(|*Comp*|) =
($\lambda \mathfrak{GF} \in_{\circ}$ *composable-arrs* (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{GF}$(|*0*|) $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ $\mathfrak{GF}$(|$1_{\mathbb{N}}$|))
⟨*proof*⟩

24.2.3 Composition

lemma *smc-FUNCT-Comp-vsv*[*intro*]: *vsv* (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Comp*|))
⟨*proof*⟩

lemma *smc-FUNCT-Comp-vdomain*:
$\mathcal{D}_{\circ}$ (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Comp*|)) = *composable-arrs* (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *smc-FUNCT-Comp-app*[*smc-FUNCT-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{smc\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{H}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{smc\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{G}$
shows $\mathfrak{M} \circ_{A\, smc\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{N} = \mathfrak{M} \cdot_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{N}$
⟨*proof*⟩

lemma *smc-FUNCT-Comp-vrange*: $\mathcal{R}_{\circ}$ (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(|*Comp*|)) $\subseteq_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

24.2.4 *FUNCT* is a semicategory

lemma (in $\mathcal{Z}$**)** *tiny-semicategory-smc-FUNCT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$
shows *tiny-semicategory* β (*smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

24.3 *Funct*

24.3.1 Definition and elementary properties

definition *smc-Funct* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}$ =
[
tm-cf-maps α $\mathfrak{A}$ $\mathfrak{B}$,
tm-ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B}$,
($\lambda \mathfrak{N} \in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(|*NTDom*|)),
($\lambda \mathfrak{N} \in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$(|*NTCod*|)),
($\lambda \mathfrak{MN} \in_{\circ}$ *composable-arrs* (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}$(|*0*|) $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ $\mathfrak{MN}$(|$1_{\mathbb{N}}$|))
]$_{\circ}$

Components.

lemma *smc-Funct-components*:
shows *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Obj \rfloor)$ = *tm-cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
and *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Arr \rfloor)$ = *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
and *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Dom \rfloor)$ = $(\lambda\mathfrak{N}\in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTDom \rfloor))$
and *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Cod \rfloor)$ = $(\lambda\mathfrak{N}\in_{\circ}$ *tm-ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\lfloor NTCod \rfloor))$
and *smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Comp \rfloor)$ =
$(\lambda\mathfrak{MN}\in_{\circ}$ *composable-arrs* (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}(\lfloor 0 \rfloor)$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ $\mathfrak{MN}(\lfloor 1_{\mathbb{N}} \rfloor))$
⟨*proof*⟩

Slicing.

lemma *smc-dg-Funct*: *smc-dg* (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}$) = *dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

context *is-tm-ntcf*
begin

lemmas-with [*folded smc-dg-Funct*, *unfolded slicing-simps*]:
smc-Funct-Dom-app = *dg-Funct-Dom-app*
and *smc-Funct-Cod-app* = *dg-Funct-Cod-app*

end

lemmas [*smc-FUNCT-cs-simps*] =
is-tm-ntcf.smc-Funct-Dom-app
is-tm-ntcf.smc-Funct-Cod-app

lemmas-with [*folded smc-dg-Funct*, *unfolded slicing-simps*]:
smc-Funct-Dom-vsv[*intro*] = *dg-Funct-Dom-vsv*
and *smc-Funct-Dom-vdomain*[*smc-FUNCT-cs-simps*] = *dg-Funct-Dom-vdomain*
and *smc-Funct-Cod-vsv*[*intro*] = *dg-Funct-Cod-vsv*
and *smc-Funct-Cod-vdomain*[*smc-FUNCT-cs-simps*] = *dg-Funct-Cod-vdomain*
and *smc-Funct-Dom-vrange* = *dg-Funct-Dom-vrange*
and *smc-Funct-Cod-vrange* = *dg-Funct-Cod-vrange*
and *smc-Funct-is-arrI* = *dg-Funct-is-arrI*
and *smc-Funct-is-arrI'*[*smc-FUNCT-cs-intros*] = *dg-Funct-is-arrI'*
and *smc-Funct-is-arrD* = *dg-Funct-is-arrD*
and *smc-Funct-is-arrE*[*elim*] = *dg-Funct-is-arrE*

24.3.2 Composable arrows

lemma *smc-Funct-composable-arrs-dg-FUNCT*:
composable-arrs (*dg-Funct* α $\mathfrak{A}$ $\mathfrak{B}$) = *composable-arrs* (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *smc-Funct-Comp*:
smc-Funct α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Comp \rfloor)$ =
$(\lambda\mathfrak{GF}\in_{\circ}$ *composable-arrs* (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{GF}(\lfloor 0 \rfloor)$ $\cdot_{NTCF\mathfrak{A},\mathfrak{B}}$ $\mathfrak{GF}(\lfloor 1_{\mathbb{N}} \rfloor))$
⟨*proof*⟩

24.3.3 Composition

lemma *smc-Funct-Comp-vsv*[*intro*]: *vsv* (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Comp \rfloor)$)
⟨*proof*⟩

lemma *smc-Funct-Comp-vdomain*:
$\mathcal{D}_{\circ}$ (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}(\lfloor Comp \rfloor)$) = *composable-arrs* (*smc-Funct* α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *smc-Funct-Comp-app*[*smc-FUNCT-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{smc\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{H}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{smc\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{G}$
shows $\mathfrak{M} \circ_{A\ smc\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}} \mathfrak{N} = \mathfrak{M} \cdot_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{N}$
⟨*proof*⟩

lemma *smc-Funct-Comp-vrange*:
assumes *category* $\alpha\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ (smc\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Comp|\!)) \subseteq_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

24.3.4 *Funct* is a semicategory

lemma *semicategory-smc-Funct*:
assumes *tiny-category* $\alpha\ \mathfrak{A}$ **and** *category* $\alpha\ \mathfrak{B}$
shows *semicategory* α (*smc-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}$) (**is** ‹*semicategory* α *?Funct*›)
⟨*proof*⟩

24.3.5 *Funct* is a subsemicategory of *FUNCT*

lemma *subsemicategory-smc-Funct-smc-FUNCT*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$ **and** *tiny-category* $\alpha\ \mathfrak{A}$ **and** *category* $\alpha\ \mathfrak{B}$
shows *smc-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B} \subseteq_{SMC\beta}$ *smc-FUNCT* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

25 *FUNCT* and *Funct*

25.1 Background

The subsection presents the theory of the categories of α-functors between two α-categories. It continues the development that was initiated in sections 23 and 24. A general reference for this section is Chapter II-4 in [7].

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

lemmas (in *is-functor*) [*cat-FUNCT-cs-simps*] = *cat-map-cs-simps*
lemmas (in *is-functor*) [*cat-FUNCT-cs-intros*] = *cat-map-cs-intros*

lemmas [*cat-FUNCT-cs-simps*] = *cat-map-cs-simps*
lemmas [*cat-FUNCT-cs-intros*] = *cat-map-cs-intros*

25.2 *FUNCT*

25.2.1 Definition and elementary properties

definition *cat-FUNCT* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$ =
    [
      cf-maps α $\mathfrak{A}$ $\mathfrak{B}$,
      ntcf-arrows α $\mathfrak{A}$ $\mathfrak{B}$,
      $(\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\!|NTDom|\!))$,
      $(\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\!|NTCod|\!))$,
      $(\lambda\mathfrak{MN}\in_\circ$*composable-arrs* (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}(\!|0|\!) \cdot_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{MN}(\!|1_{\mathbb{N}}|\!))$,
      $(\lambda\mathfrak{F}\in_\circ$*cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$. *ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F})$
    ]$_\circ$

Components.

lemma *cat-FUNCT-components*:
  shows [*cat-FUNCT-cs-simps*]: *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|Obj|\!)$ = *cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$
    and *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|Arr|\!)$ = *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
    and *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|Dom|\!)$ = $(\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\!|NTDom|\!))$
    and *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|Cod|\!)$ = $(\lambda\mathfrak{N}\in_\circ$*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}(\!|NTCod|\!))$
    and *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|Comp|\!)$ =
      $(\lambda\mathfrak{MN}\in_\circ$*composable-arrs* (*dg-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$). $\mathfrak{MN}(\!|0|\!) \cdot_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{MN}(\!|1_{\mathbb{N}}|\!))$
    and *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}(\!|CId|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*cf-maps* α $\mathfrak{A}$ $\mathfrak{B}$. *ntcf-arrow-id* $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F})$
  ⟨*proof*⟩

Slicing.

lemma *cat-smc-FUNCT*: *cat-smc* (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) = *smc-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

context *is-ntcf*
begin

lemmas-with [*folded cat-smc-FUNCT*, *unfolded slicing-simps*]:
  cat-FUNCT-Dom-app = *smc-FUNCT-Dom-app*
  and *cat-FUNCT-Cod-app* = *smc-FUNCT-Cod-app*

end

lemmas [*smc-FUNCT-cs-simps*] =
  is-ntcf.cat-FUNCT-Dom-app

is-ntcf.cat-FUNCT-Cod-app

lemmas-with [*folded cat-smc-FUNCT*, *unfolded slicing-simps*]:
cat-FUNCT-Dom-vsv[*intro*] = *smc-FUNCT-Dom-vsv*
and *cat-FUNCT-Dom-vdomain*[*cat-FUNCT-cs-simps*] = *smc-FUNCT-Dom-vdomain*
and *cat-FUNCT-Cod-vsv*[*intro*] = *smc-FUNCT-Cod-vsv*
and *cat-FUNCT-Cod-vdomain*[*cat-FUNCT-cs-simps*] = *smc-FUNCT-Cod-vdomain*
and *cat-FUNCT-Dom-vrange* = *smc-FUNCT-Dom-vrange*
and *cat-FUNCT-Cod-vrange* = *smc-FUNCT-Cod-vrange*
and *cat-FUNCT-is-arrI* = *smc-FUNCT-is-arrI*
and *cat-FUNCT-is-arrI'*[*cat-FUNCT-cs-intros*] = *smc-FUNCT-is-arrI'*
and *cat-FUNCT-is-arrD* = *smc-FUNCT-is-arrD*
and *cat-FUNCT-is-arrE*[*elim*] = *smc-FUNCT-is-arrE*

lemmas-with [*folded cat-smc-FUNCT*, *unfolded slicing-simps*]:
cat-FUNCT-Comp-app[*cat-FUNCT-cs-simps*] = *smc-FUNCT-Comp-app*

25.2.2 Identity

mk-VLambda *cat-FUNCT-components*(*6*)
|*vsv cat-FUNCT-CId-vsv*[*cat-FUNCT-cs-intros*]|
|*vdomain cat-FUNCT-CId-vdomain*[*cat-FUNCT-cs-simps*]|
|*app cat-FUNCT-CId-app*[*cat-FUNCT-cs-simps*]|

lemma *smc-FUNCT-CId-vrange*: $\mathcal{R}_{\circ}$ (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$(‖*CId*‖)) $\subseteq_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

25.2.3 The conversion of a natural transformation arrow to a natural transformation is a bijection

lemma *bij-betw-ntcf-of-ntcf-arrow*:
bij-betw
(*ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$)
(*elts* (*ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$))
(*elts* (*ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$))
⟨*proof*⟩

lemma *bij-betw-ntcf-of-ntcf-arrow-Hom*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *bij-betw*
(*ntcf-of-ntcf-arrow* $\mathfrak{A}$ $\mathfrak{B}$)
(*elts* (*Hom* (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* $\mathfrak{F}$) (*cf-map* $\mathfrak{G}$)))
(*elts* (*these-ntcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$))
⟨*proof*⟩

25.2.4 *FUNCT* is a category

lemma (**in** $\mathcal{Z}$) *tiny-category-cat-FUNCT*[*cat-FUNCT-cs-intros*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$
shows *tiny-category* β (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) (**is** ‹*tiny-category* β *?FUNCT*›)
⟨*proof*⟩

lemmas (**in** $\mathcal{Z}$) [*cat-FUNCT-cs-intros*] = *tiny-category-cat-FUNCT*

25.2.5 Isomorphism

lemma *cat-FUNCT-is-iso-arrI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{N}$: *cf-map* $\mathfrak{F} \mapsto_{iso}$ *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$ *cf-map* $\mathfrak{G}$

⟨*proof*⟩

lemma *cat-FUNCT-is-iso-arrI'*[*cat-FUNCT-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N}' = ntcf\text{-}arrow\ \mathfrak{N}$
and $\mathfrak{F}' = cf\text{-}map\ \mathfrak{F}$
and $\mathfrak{G}' = cf\text{-}map\ \mathfrak{G}$
shows $\mathfrak{N}' : \mathfrak{F}' \mapsto_{iso} cat\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ cf\text{-}map\ \mathfrak{G}$
⟨*proof*⟩

lemma *cat-FUNCT-is-iso-arrD*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{iso} cat\text{-}FUNCT\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G}$ (**is** $\langle \mathfrak{N} : \mathfrak{F} \mapsto_{iso} ?FUNCT\ \mathfrak{G} \rangle$)
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{N}$:
cf-of-cf-map $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F} \mapsto_{CF.iso} cf\text{-}of\text{-}cf\text{-}map\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} = ntcf\text{-}arrow\ (ntcf\text{-}of\text{-}ntcf\text{-}arrow\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{N})$
and $\mathfrak{F} = cf\text{-}map\ (cf\text{-}of\text{-}cf\text{-}map\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F})$
and $\mathfrak{G} = cf\text{-}map\ (cf\text{-}of\text{-}cf\text{-}map\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G})$
⟨*proof*⟩

25.3 *Funct*

25.3.1 Definition and elementary properties

definition *cat-Funct* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B} =$
[
tm-cf-maps $\alpha\ \mathfrak{A}\ \mathfrak{B}$,
tm-ntcf-arrows $\alpha\ \mathfrak{A}\ \mathfrak{B}$,
$(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTDom|\!))$,
$(\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTCod|\!))$,
$(\lambda\mathfrak{MN}\in_{\circ} composable\text{-}arrs\ (dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}).\ \mathfrak{MN}(\!|0|\!) \bullet_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{MN}(\!|1_{\mathbb{N}}|\!))$,
$(\lambda\mathfrak{F}\in_{\circ} tm\text{-}cf\text{-}maps\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ ntcf\text{-}arrow\text{-}id\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F})$
]$_{\circ}$

Components.

lemma *cat-Funct-components*:
shows [*cat-FUNCT-cs-simps*]: *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Obj|\!) = tm\text{-}cf\text{-}maps\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
and *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Arr|\!) = tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
and *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Dom|\!) = (\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTDom|\!))$
and *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Cod|\!) = (\lambda\mathfrak{N}\in_{\circ} tm\text{-}ntcf\text{-}arrows\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N}(\!|NTCod|\!))$
and *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|Comp|\!) =$
$(\lambda\mathfrak{MN}\in_{\circ} composable\text{-}arrs\ (dg\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}).\ \mathfrak{MN}(\!|0|\!) \bullet_{NTCF\mathfrak{A},\mathfrak{B}} \mathfrak{MN}(\!|1_{\mathbb{N}}|\!))$
and *cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}(\!|CId|\!) = (\lambda\mathfrak{F}\in_{\circ} tm\text{-}cf\text{-}maps\ \alpha\ \mathfrak{A}\ \mathfrak{B}.\ ntcf\text{-}arrow\text{-}id\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F})$
⟨*proof*⟩

Slicing.

lemma *cat-smc-Funct*: *cat-smc* (*cat-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}$) = *smc-Funct* $\alpha\ \mathfrak{A}\ \mathfrak{B}$
⟨*proof*⟩

context *is-tm-ntcf*
begin

lemmas-with [*folded cat-smc-Funct*, *unfolded slicing-simps*]:
cat-Funct-Dom-app = *smc-Funct-Dom-app*
and *cat-Funct-Cod-app* = *smc-Funct-Cod-app*

end

lemmas [*cat-FUNCT-cs-simps*] =
is-tm-ntcf.cat-Funct-Dom-app
is-tm-ntcf.cat-Funct-Cod-app

lemmas-with [*folded cat-smc-Funct*, *unfolded slicing-simps*]:
cat-Funct-Dom-vsv[*cat-FUNCT-cs-intros*] = *smc-Funct-Dom-vsv*
and *cat-Funct-Dom-vdomain*[*cat-FUNCT-cs-simps*] = *smc-Funct-Dom-vdomain*
and *cat-Funct-Cod-vsv*[*cat-FUNCT-cs-intros*] = *smc-Funct-Cod-vsv*
and *cat-Funct-Cod-vdomain*[*cat-FUNCT-cs-simps*] = *smc-Funct-Cod-vdomain*
and *cat-Funct-Dom-vrange* = *smc-Funct-Dom-vrange*
and *cat-Funct-Cod-vrange* = *smc-Funct-Cod-vrange*
and *cat-Funct-is-arrI* = *smc-Funct-is-arrI*
and *cat-Funct-is-arrI'*[*cat-FUNCT-cs-intros*] = *smc-Funct-is-arrI'*
and *cat-Funct-is-arrD* = *smc-Funct-is-arrD*
and *cat-Funct-is-arrE*[*elim*] = *smc-Funct-is-arrE*

lemmas-with [*folded cat-smc-Funct*, *unfolded slicing-simps*]:
cat-Funct-Comp-app[*cat-FUNCT-cs-simps*] = *smc-Funct-Comp-app*

25.3.2 Identity

mk-VLambda *cat-Funct-components*(*6*)
|*vsv cat-Funct-CId-vsv*[*intro*]|
|*vdomain cat-Funct-CId-vdomain*[*cat-FUNCT-cs-simps*]|
|*app cat-Funct-CId-app*[*cat-FUNCT-cs-simps*]|

lemma *smc-Funct-CId-vrange*: $\mathcal{R}_{\circ}$ (*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$(|*CId*|)) $\subseteq_{\circ}$ *ntcf-arrows* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

25.3.3 *Funct* is a category

lemma *category-cat-Funct*:
assumes *tiny-category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$
shows *category* α (*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$) (**is** ‹*category* α *?Funct*›)
⟨*proof*⟩

lemma *category-cat-Funct'*[*cat-FUNCT-cs-intros*]:
assumes *tiny-category* α $\mathfrak{A}$
and *category* α $\mathfrak{B}$
and $\beta = \alpha$
shows *category* α (*cat-Funct* β $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

25.3.4 *Funct* is a subcategory of *FUNCT*

lemma *subcategory-cat-Funct-cat-FUNCT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** *tiny-category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{B}$
shows *cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$ $\subseteq_{C\beta}$ *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$
⟨*proof*⟩

25.3.5 Isomorphism

lemma (**in** *is-tm-iso-ntcf*) *cat-Funct-is-iso-arrI*:
assumes *category* α $\mathfrak{B}$
shows *ntcf-arrow* $\mathfrak{N}$: *cf-map* $\mathfrak{F}$ $\mapsto_{iso}$ *cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$ *cf-map* $\mathfrak{G}$
⟨*proof*⟩

lemma (**in** *is-tm-iso-ntcf*) *cat-Funct-is-iso-arrI'*:
assumes *category* α $\mathfrak{B}$

and $\mathfrak{N}' = ntcf\text{-}arrow\ \mathfrak{N}$
and $\mathfrak{F}' = cf\text{-}map\ \mathfrak{F}$
and $\mathfrak{G}' = cf\text{-}map\ \mathfrak{G}$
shows $\mathfrak{N}' : \mathfrak{F}' \mapsto_{iso} cat\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ cf\text{-}map\ \mathfrak{G}$
⟨*proof*⟩

lemmas [*cat-FUNCT-cs-intros*] =
is-tm-iso-ntcf.cat-Funct-is-iso-arrI'[*rotated 2*]

lemma *cat-Funct-is-iso-arrD*:
assumes *tiny-category* $\alpha\ \mathfrak{A}$
and *category* $\alpha\ \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{iso} cat\text{-}Funct\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G}$ (**is** $\langle \mathfrak{N} : \mathfrak{F} \mapsto_{iso} ?Funct\ \mathfrak{G} \rangle$)
shows *ntcf-of-ntcf-arrow* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{N}$:
cf-of-cf-map $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{F} \mapsto_{CF.tm.iso}$ *cf-of-cf-map* $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N} = ntcf\text{-}arrow\ (ntcf\text{-}of\text{-}ntcf\text{-}arrow\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{N})$
and $\mathfrak{F} = cf\text{-}map\ (cf\text{-}of\text{-}cf\text{-}map\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F})$
and $\mathfrak{G} = cf\text{-}map\ (cf\text{-}of\text{-}cf\text{-}map\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G})$
⟨*proof*⟩

25.4 Diagonal functor

25.4.1 Definition and elementary properties

See Chapter III-3 in [7].

definition *cf-diagonal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ ($\langle \Delta_{CF} \rangle$)
where $\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C} =$
[
$(\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ cf\text{-}map\ (cf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ a))$,
$(\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ ntcf\text{-}arrow\ (ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f))$,
$\mathfrak{C}$,
$cat\text{-}FUNCT\ \alpha\ \mathfrak{J}\ \mathfrak{C}$
]$_\circ$

Components.

lemma *cf-diagonal-components*:
shows $\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|ObjMap|\!) = (\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ cf\text{-}map\ (cf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ a))$
and $\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|ArrMap|\!) = (\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ ntcf\text{-}arrow\ (ntcf\text{-}const\ \mathfrak{J}\ \mathfrak{C}\ f))$
and $\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|HomDom|\!) = \mathfrak{C}$
and $\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|HomCod|\!) = cat\text{-}FUNCT\ \alpha\ \mathfrak{J}\ \mathfrak{C}$
⟨*proof*⟩

25.4.2 Object map

mk-VLambda *cf-diagonal-components*(*1*)
|*vsv cf-diagonal-ObjMap-vsv*[*cat-cs-intros*]|
|*vdomain cf-diagonal-ObjMap-vdomain*[*cat-cs-simps*]|
|*app cf-diagonal-ObjMap-app*[*cat-cs-simps*]|

lemma *cf-diagonal-ObjMap-vrange*:
assumes $\mathcal{Z}\ \beta$**and** $\alpha \in_\circ \beta$ **and** *category* $\alpha\ \mathfrak{J}$ **and** *category* $\alpha\ \mathfrak{C}$
shows $\mathcal{R}_\circ\ (\Delta_{CF}\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|ObjMap|\!)) \subseteq_\circ cat\text{-}FUNCT\ \alpha\ \mathfrak{J}\ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

25.4.3 Arrow map

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

|*vdomain cf-diagonal-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-diagonal-ArrMap-app*[*cat-cs-simps*]|

25.4.4 Diagonal functor is a functor

lemma *cf-diagonal-is-functor*[*cat-cs-intros*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** *category* α $\mathfrak{J}$ **and** *category* α $\mathfrak{C}$
shows Δ_{CF} α $\mathfrak{J}$ $\mathfrak{C}$: $\mathfrak{C}$ $\mapsto\mapsto_{C\beta}$ *cat-FUNCT* α $\mathfrak{J}$ $\mathfrak{C}$ (**is** ‹*?*Δ : $\mathfrak{C}$ $\mapsto\mapsto_{C\beta}$ *?FUNCT*›)
⟨*proof*⟩

lemma *cf-diagonal-is-functor'*[*cat-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_{\circ} \beta$
and *category* α $\mathfrak{J}$
and *category* α $\mathfrak{C}$
and $\mathfrak{A}'$ = $\mathfrak{C}$
and $\mathfrak{B}'$ = *cat-FUNCT* α $\mathfrak{J}$ $\mathfrak{C}$
shows Δ_{CF} α $\mathfrak{J}$ $\mathfrak{C}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\beta}$ $\mathfrak{B}'$
⟨*proof*⟩

25.5 Diagonal functor for functors with tiny maps

25.5.1 Definition and elementary properties

See Chapter III-3 in [7].

definition *tm-cf-diagonal* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$\Delta_{CF.tm}$›)
where $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$ =
[
(λa∈$_{\circ}$$\mathfrak{C}$(|*Obj*|). *cf-map* (*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ a)),
(λf∈$_{\circ}$$\mathfrak{C}$(|*Arr*|). *ntcf-arrow* (*ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f)),
$\mathfrak{C}$,
cat-Funct α $\mathfrak{J}$ $\mathfrak{C}$
]$_{\circ}$

Components.

lemma *tm-cf-diagonal-components*:
shows $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$(|*ObjMap*|) = (λa∈$_{\circ}$$\mathfrak{C}$(|*Obj*|). *cf-map* (*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ a))
and $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$(|*ArrMap*|) = (λf∈$_{\circ}$$\mathfrak{C}$(|*Arr*|). *ntcf-arrow* (*ntcf-const* $\mathfrak{J}$ $\mathfrak{C}$ f))
and $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$(|*HomDom*|) = $\mathfrak{C}$
and $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$(|*HomCod*|) = *cat-Funct* α $\mathfrak{J}$ $\mathfrak{C}$
⟨*proof*⟩

25.5.2 Object map

mk-VLambda *tm-cf-diagonal-components*(*1*)
|*vsv tm-cf-diagonal-ObjMap-vsv*[*cat-cs-intros*]|
|*vdomain tm-cf-diagonal-ObjMap-vdomain*[*cat-cs-simps*]|
|*app tm-cf-diagonal-ObjMap-app*[*cat-cs-simps*]|

lemma *tm-cf-diagonal-ObjMap-vrange*:
assumes *tiny-category* α $\mathfrak{J}$ **and** *category* α $\mathfrak{C}$
shows $\mathcal{R}_{\circ}$ ($\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$(|*ObjMap*|)) $\subseteq_{\circ}$ *cat-Funct* α $\mathfrak{J}$ $\mathfrak{C}$(|*Obj*|)
⟨*proof*⟩

25.5.3 Arrow map

mk-VLambda *tm-cf-diagonal-components*(*2*)
|*vsv tm-cf-diagonal-ArrMap-vsv*[*cat-cs-intros*]|

|*vdomain tm-cf-diagonal-ArrMap-vdomain*[*cat-cs-simps*]|
|*app tm-cf-diagonal-ArrMap-app*[*cat-cs-simps*]|

25.5.4 Diagonal functor for functors with tiny maps is a functor

lemma *tm-cf-diagonal-is-functor*[*cat-cs-intros*]:
assumes *tiny-category* α $\mathfrak{J}$ **and** *category* α $\mathfrak{C}$
shows $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$: $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ *cat-Funct* α $\mathfrak{J}$ $\mathfrak{C}$
(**is** ‹*?*Δ : $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ *?Funct*›)
⟨*proof*⟩

lemma *tm-cf-diagonal-is-functor'*[*cat-cs-intros*]:
assumes *tiny-category* α $\mathfrak{J}$
and *category* α $\mathfrak{C}$
and α' = α
and $\mathfrak{A}$ = $\mathfrak{C}$
and $\mathfrak{B}$ = *cat-Funct* α $\mathfrak{J}$ $\mathfrak{C}$
shows $\Delta_{CF.tm}$ α $\mathfrak{J}$ $\mathfrak{C}$: $\mathfrak{A}$ $\mapsto\mapsto_{C\alpha'}$ $\mathfrak{B}$
⟨*proof*⟩

25.6 Functor raised to the power of a category

25.6.1 Definition and elementary properties

Most of the definitions and the results presented in this and the remaining subsections can be found in [7] and [12] (e.g., see Chapter X-3 in [7]).

definition *exp-cf-cat* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *exp-cf-cat* α $\mathfrak{K}$ $\mathfrak{A}$ =
[
(
$\lambda\mathfrak{S}\in_{\circ}$ *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|))(|*Obj*|).
cf-map ($\mathfrak{K}$ $\circ_{CF}$ *cf-of-cf-map* $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|)) $\mathfrak{S}$)
),
(
$\lambda\sigma\in_{\circ}$ *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|))(|*Arr*|).
ntcf-arrow ($\mathfrak{K}$ $\circ_{CF-NTCF}$ *ntcf-of-ntcf-arrow* $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|)) σ)
),
cat-FUNCT α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|)),
cat-FUNCT α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomCod*|))
]$_{\circ}$

Components.

lemma *exp-cf-cat-components*:
shows *exp-cf-cat* α $\mathfrak{K}$ $\mathfrak{A}$(|*ObjMap*|) =
(
$\lambda\mathfrak{S}\in_{\circ}$ *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|))(|*Obj*|).
cf-map ($\mathfrak{K}$ $\circ_{CF}$ *cf-of-cf-map* $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|)) $\mathfrak{S}$)
)
and
exp-cf-cat α $\mathfrak{K}$ $\mathfrak{A}$(|*ArrMap*|) =
(
$\lambda\sigma\in_{\circ}$ *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|))(|*Arr*|).
ntcf-arrow ($\mathfrak{K}$ $\circ_{CF-NTCF}$ (*ntcf-of-ntcf-arrow* $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|)) σ))
)
and *exp-cf-cat* α $\mathfrak{K}$ $\mathfrak{A}$(|*HomDom*|) = *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomDom*|))
and *exp-cf-cat* α $\mathfrak{K}$ $\mathfrak{A}$(|*HomCod*|) = *cat-FUNCT* α $\mathfrak{A}$ ($\mathfrak{K}$(|*HomCod*|))
⟨*proof*⟩

25.6.2 Object map

mk-VLambda *exp-cf-cat-components*(*1*)
|*vsv exp-cf-cat-components-ObjMap-vsv*[*cat-FUNCT-cs-intros*]|

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

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

mk-VLambda *exp-cf-cat-components*(*1*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
|*vdomain exp-cf-cat-components-ObjMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-cf-cat-components-ObjMap-app*[*cat-FUNCT-cs-simps*]|

end

25.6.3 Arrow map

mk-VLambda *exp-cf-cat-components*(*2*)
|*vsv exp-cf-cat-components-ArrMap-vsv*[*cat-FUNCT-cs-intros*]|

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

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

mk-VLambda *exp-cf-cat-components*(*2*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
|*vdomain exp-cf-cat-components-ArrMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-cf-cat-components-ArrMap-app*[*cat-FUNCT-cs-simps*]|

end

25.6.4 Domain and codomain

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

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

lemmas *exp-cf-cat-HomDom*[*cat-FUNCT-cs-simps*] =
exp-cf-cat-components(*3*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
and *exp-cf-cat-HomCod*[*cat-FUNCT-cs-simps*] =
exp-cf-cat-components(*4*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]

end

25.6.5 Functor raised to the power of a category is a functor

lemma *exp-cf-cat-is-tiny-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** *category* α $\mathfrak{A}$ **and** $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *exp-cf-cat* α $\mathfrak{K}$ $\mathfrak{A}$: *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$ $\mapsto\mapsto_{C.tiny\beta}$ *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *exp-cf-cat-is-tiny-functor'*[*cat-FUNCT-cs-intros*]:
assumes $\mathcal{Z}\ \beta$
and $\alpha \in_{\circ} \beta$
and *category* $\alpha\ \mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' = \textit{cat-FUNCT}\ \alpha\ \mathfrak{A}\ \mathfrak{B}$
and $\mathfrak{B}' = \textit{cat-FUNCT}\ \alpha\ \mathfrak{A}\ \mathfrak{C}$
shows $\textit{exp-cf-cat}\ \alpha\ \mathfrak{K}\ \mathfrak{A} : \mathfrak{A}' \mapsto\mapsto_{C.tiny\beta} \mathfrak{B}'$
⟨*proof*⟩

25.6.6 Further properties

lemma *exp-cf-cat-cf-comp*:
assumes *category* $\alpha\ \mathfrak{D}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\textit{exp-cf-cat}\ \alpha\ (\mathfrak{G} \circ_{CF} \mathfrak{F})\ \mathfrak{D} = \textit{exp-cf-cat}\ \alpha\ \mathfrak{G}\ \mathfrak{D} \circ_{CF} \textit{exp-cf-cat}\ \alpha\ \mathfrak{F}\ \mathfrak{D}$
⟨*proof*⟩

lemma *exp-cf-cat-cf-id-cat*:
assumes *category* $\alpha\ \mathfrak{C}$ **and** *category* $\alpha\ \mathfrak{D}$
shows $\textit{exp-cf-cat}\ \alpha\ (\textit{cf-id}\ \mathfrak{C})\ \mathfrak{D} = \textit{cf-id}\ (\textit{cat-FUNCT}\ \alpha\ \mathfrak{D}\ \mathfrak{C})$
⟨*proof*⟩

lemma *cf-comp-exp-cf-cat-exp-cf-cat-cf-id*[*cat-FUNCT-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\textit{exp-cf-cat}\ \alpha\ \mathfrak{F}\ \mathfrak{A} \circ_{CF} \textit{exp-cf-cat}\ \alpha\ (\textit{cf-id}\ \mathfrak{B})\ \mathfrak{A} = \textit{exp-cf-cat}\ \alpha\ \mathfrak{F}\ \mathfrak{A}$
⟨*proof*⟩

lemma *cf-comp-exp-cf-cat-cf-id-exp-cf-cat*[*cat-FUNCT-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\textit{exp-cf-cat}\ \alpha\ (\textit{cf-id}\ \mathfrak{C})\ \mathfrak{A} \circ_{CF} \textit{exp-cf-cat}\ \alpha\ \mathfrak{F}\ \mathfrak{A} = \textit{exp-cf-cat}\ \alpha\ \mathfrak{F}\ \mathfrak{A}$
⟨*proof*⟩

25.7 Category raised to the power of a functor

25.7.1 Definition and elementary properties

definition $\textit{exp-cat-cf} :: V \Rightarrow V \Rightarrow V \Rightarrow V$
where $\textit{exp-cat-cf}\ \alpha\ \mathfrak{A}\ \mathfrak{K} =$
[
(
$\lambda \mathfrak{S} \in_{\circ} \textit{cat-FUNCT}\ \alpha\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}(\!|\textit{Obj}|\!).$
$\textit{cf-map}\ (\textit{cf-of-cf-map}\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}\ \mathfrak{S} \circ_{CF} \mathfrak{K})$
),
(
$\lambda \sigma \in_{\circ} \textit{cat-FUNCT}\ \alpha\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}(\!|\textit{Arr}|\!).$
$\textit{ntcf-arrow}\ (\textit{ntcf-of-ntcf-arrow}\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}\ \sigma \circ_{NTCF-CF} \mathfrak{K})$
),
$\textit{cat-FUNCT}\ \alpha\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A},$
$\textit{cat-FUNCT}\ \alpha\ (\mathfrak{K}(\!|\textit{HomDom}|\!))\ \mathfrak{A}$
]$_{\circ}$

Components.

lemma *exp-cat-cf-components*:
shows $\textit{exp-cat-cf}\ \alpha\ \mathfrak{A}\ \mathfrak{K}(\!|\textit{ObjMap}|\!) =$
(
$\lambda \mathfrak{S} \in_{\circ} \textit{cat-FUNCT}\ \alpha\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}(\!|\textit{Obj}|\!).$
$\textit{cf-map}\ (\textit{cf-of-cf-map}\ (\mathfrak{K}(\!|\textit{HomCod}|\!))\ \mathfrak{A}\ \mathfrak{S} \circ_{CF} \mathfrak{K})$

)
and *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}(\!|ArrMap|\!)$ =
(
$\lambda\sigma\in_{\circ}$ *cat-FUNCT* α ($\mathfrak{K}(\!|HomCod|\!)$) $\mathfrak{A}(\!|Arr|\!)$.
ntcf-arrow (*ntcf-of-ntcf-arrow* ($\mathfrak{K}(\!|HomCod|\!)$) $\mathfrak{A}$ σ $\circ_{NTCF-CF}$ $\mathfrak{K}$)
)
and *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}(\!|HomDom|\!)$ = *cat-FUNCT* α ($\mathfrak{K}(\!|HomCod|\!)$) $\mathfrak{A}$
and *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}(\!|HomCod|\!)$ = *cat-FUNCT* α ($\mathfrak{K}(\!|HomDom|\!)$) $\mathfrak{A}$
⟨*proof*⟩

25.7.2 Object map

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

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

mk-VLambda *exp-cat-cf-components*(*1*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
|*vsv exp-cat-cf-components-ObjMap-vsv*[*cat-FUNCT-cs-intros*]|
|*vdomain exp-cat-cf-components-ObjMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-cat-cf-components-ObjMap-app*[*cat-FUNCT-cs-simps*]|

end

25.7.3 Arrow map

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

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

mk-VLambda *exp-cat-cf-components*(*2*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
|*vsv exp-cat-cf-components-ArrMap-vsv*[*cat-FUNCT-cs-intros*]|
|*vdomain exp-cat-cf-components-ArrMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-cat-cf-components-ArrMap-app*[*cat-FUNCT-cs-simps*]|

end

25.7.4 Domain and codomain

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

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

lemmas *exp-cat-cf-HomDom*[*cat-FUNCT-cs-simps*] =
exp-cat-cf-components(*3*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]
and *exp-cat-cf-HomCod*[*cat-FUNCT-cs-simps*] =
exp-cat-cf-components(*4*)[**where** $\mathfrak{K}$=$\mathfrak{K}$ **and** α=α, *unfolded cat-cs-simps*]

end

25.7.5 Category raised to the power of a functor is a functor

lemma *exp-cat-cf-is-tiny-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** *category* α $\mathfrak{A}$ **and** $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}$: *cat-FUNCT* α $\mathfrak{C}$ $\mathfrak{A}$ $\mapsto\mapsto_{C.tiny\beta}$ *cat-FUNCT* α $\mathfrak{B}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *exp-cat-cf-is-tiny-functor'*[*cat-FUNCT-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_{\circ} \beta$
and *category* α $\mathfrak{A}$
and $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{A}' =$ *cat-FUNCT* α $\mathfrak{C}$ $\mathfrak{A}$
and $\mathfrak{B}' =$ *cat-FUNCT* α $\mathfrak{B}$ $\mathfrak{A}$
shows *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{K}$: $\mathfrak{A}' \mapsto\mapsto_{C.tiny\beta}$ $\mathfrak{B}'$
⟨*proof*⟩

25.7.6 Further properties

lemma *exp-cat-cf-cat-cf-id*:
assumes *category* α $\mathfrak{A}$ **and** *category* α $\mathfrak{C}$
shows *exp-cat-cf* α $\mathfrak{A}$ (*cf-id* $\mathfrak{C}$) = *cf-id* (*cat-FUNCT* α $\mathfrak{C}$ $\mathfrak{A}$)
⟨*proof*⟩

lemma *exp-cat-cf-cf-comp*:
assumes *category* α $\mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \mathfrak{D}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *exp-cat-cf* α $\mathfrak{A}$ ($\mathfrak{G} \circ_{CF} \mathfrak{F}$) = *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{F} \circ_{CF}$ *exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{G}$
⟨*proof*⟩

25.8 Natural transformation raised to the power of a category

25.8.1 Definition and elementary properties

definition *exp-ntcf-cat* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$ =
[
(
$\lambda \mathfrak{S} \in_{\circ}$ *cat-FUNCT* α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|))(|*Obj*|).
ntcf-arrow ($\mathfrak{N} \circ_{NTCF-CF}$ *cf-of-cf-map* $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|)) $\mathfrak{S}$)
),
exp-cf-cat α ($\mathfrak{N}$(|*NTDom*|)) $\mathfrak{D}$,
exp-cf-cat α ($\mathfrak{N}$(|*NTCod*|)) $\mathfrak{D}$,
cat-FUNCT α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|)),
cat-FUNCT α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGCod*|))
]$_{\circ}$

Components.

lemma *exp-ntcf-cat-components*:
shows *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$(|*NTMap*|) =
(
$\lambda \mathfrak{S} \in_{\circ}$ *cat-FUNCT* α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|))(|*Obj*|).
ntcf-arrow ($\mathfrak{N} \circ_{NTCF-CF}$ *cf-of-cf-map* $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|)) $\mathfrak{S}$)
)
and *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$(|*NTDom*|) = *exp-cf-cat* α ($\mathfrak{N}$(|*NTDom*|)) $\mathfrak{D}$
and *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$(|*NTCod*|) = *exp-cf-cat* α ($\mathfrak{N}$(|*NTCod*|)) $\mathfrak{D}$
and *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$(|*NTDGDom*|) = *cat-FUNCT* α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGDom*|))
and *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$(|*NTDGCod*|) = *cat-FUNCT* α $\mathfrak{D}$ ($\mathfrak{N}$(|*NTDGCod*|))
⟨*proof*⟩

25.8.2 Natural transformation map

mk-VLambda *exp-ntcf-cat-components*(*1*)
|*vsv exp-ntcf-cat-components-NTMap-vsv*[*cat-FUNCT-cs-intros*]|

context *is-ntcf*
begin

lemmas *exp-ntcf-cat-components′* =
exp-ntcf-cat-components[**where** α=α **and** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded cat-cs-simps*]

lemmas [*cat-FUNCT-cs-simps*] = *exp-ntcf-cat-components′*(*2–5*)

mk-VLambda *exp-ntcf-cat-components*(*1*)[**where** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded cat-cs-simps*]
|*vdomain exp-ntcf-cat-components-NTMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-ntcf-cat-components-NTMap-app*[*cat-FUNCT-cs-simps*]|

end

lemmas [*cat-FUNCT-cs-simps*] =
is-ntcf.exp-ntcf-cat-components′(*2–5*)
is-ntcf.exp-ntcf-cat-components-NTMap-vdomain
is-ntcf.exp-ntcf-cat-components-NTMap-app

25.8.3 Natural transformation raised to the power of a category is a natural transformation

lemma *exp-ntcf-cat-is-tiny-ntcf*:
assumes $\mathcal{Z}$ β
and α $\in_\circ$ β
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *category* α $\mathfrak{D}$
shows *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$:
exp-cf-cat α $\mathfrak{F}$ $\mathfrak{D}$ $\mapsto_{CF.tiny}$ *exp-cf-cat* α $\mathfrak{G}$ $\mathfrak{D}$:
cat-FUNCT α $\mathfrak{D}$ $\mathfrak{A}$ $\mapsto\mapsto_{C.tiny\beta}$ *cat-FUNCT* α $\mathfrak{D}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *exp-ntcf-cat-is-tiny-ntcf′*[*cat-FUNCT-cs-intros*]:
assumes $\mathcal{Z}$ β
and α $\in_\circ$ β
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *category* α $\mathfrak{D}$
and $\mathfrak{F}'$ = *exp-cf-cat* α $\mathfrak{F}$ $\mathfrak{D}$
and $\mathfrak{G}'$ = *exp-cf-cat* α $\mathfrak{G}$ $\mathfrak{D}$
and $\mathfrak{A}'$ = *cat-FUNCT* α $\mathfrak{D}$ $\mathfrak{A}$
and $\mathfrak{B}'$ = *cat-FUNCT* α $\mathfrak{D}$ $\mathfrak{B}$
shows *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$: $\mathfrak{F}' \mapsto_{CF.tiny} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C.tiny\beta} \mathfrak{B}'$
⟨*proof*⟩

25.8.4 Further properties

lemma *exp-ntcf-cat-cf-ntcf-comp*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *category* α $\mathfrak{D}$
shows
exp-ntcf-cat α ($\mathfrak{H}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$) $\mathfrak{D}$ =
exp-cf-cat α $\mathfrak{H}$ $\mathfrak{D}$ $\circ_{CF-NTCF}$ *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{D}$
⟨*proof*⟩

lemma *exp-ntcf-cat-ntcf-cf-comp*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
    and *category* α $\mathfrak{D}$
  shows
    exp-ntcf-cat α $(\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H})$ $\mathfrak{D}$ =
      exp-ntcf-cat α $\mathfrak{N}$ $\mathfrak{D}$ $\circ_{NTCF-CF}$ *exp-cf-cat* α $\mathfrak{H}$ $\mathfrak{D}$
⟨*proof*⟩

lemma *exp-ntcf-cat-ntcf-vcomp*:
  assumes *category* α $\mathfrak{A}$
    and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows
    exp-ntcf-cat α $(\mathfrak{M} \cdot_{NTCF} \mathfrak{N})$ $\mathfrak{A}$ =
      exp-ntcf-cat α $\mathfrak{M}$ $\mathfrak{A}$ $\cdot_{NTCF}$ *exp-ntcf-cat* α $\mathfrak{N}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *ntcf-id-exp-cf-cat*:
  assumes *category* α $\mathfrak{A}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
  shows *ntcf-id* (*exp-cf-cat* α $\mathfrak{F}$ $\mathfrak{A}$) = *exp-ntcf-cat* α (*ntcf-id* $\mathfrak{F}$) $\mathfrak{A}$
⟨*proof*⟩

25.9 Category raised to the power of the natural transformation

25.9.1 Definition and elementary properties

definition *exp-cat-ntcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}$ =
    [
      (
        $\lambda\mathfrak{S}\epsilon_{\circ}$*cat-FUNCT* α $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}(\!|Obj|\!)$.
          ntcf-arrow (*cf-of-cf-map* $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}$ $\mathfrak{S}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$)
      ),
      exp-cat-cf α $\mathfrak{C}$ $(\mathfrak{N}(\!|NTDom|\!))$,
      exp-cat-cf α $\mathfrak{C}$ $(\mathfrak{N}(\!|NTCod|\!))$,
      cat-FUNCT α $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}$,
      cat-FUNCT α $(\mathfrak{N}(\!|NTDGDom|\!))$ $\mathfrak{C}$
    ]$_{\circ}$

Components.

lemma *exp-cat-ntcf-components*:
  shows *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}(\!|NTMap|\!)$ =
    (
      $\lambda\mathfrak{S}\epsilon_{\circ}$*cat-FUNCT* α $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}(\!|Obj|\!)$.
        ntcf-arrow (*cf-of-cf-map* $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}$ $\mathfrak{S}$ $\circ_{CF-NTCF}$ $\mathfrak{N}$)
)
    and *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}(\!|NTDom|\!)$ = *exp-cat-cf* α $\mathfrak{C}$ $(\mathfrak{N}(\!|NTDom|\!))$
    and *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}(\!|NTCod|\!)$ = *exp-cat-cf* α $\mathfrak{C}$ $(\mathfrak{N}(\!|NTCod|\!))$
    and *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}(\!|NTDGDom|\!)$ = *cat-FUNCT* α $(\mathfrak{N}(\!|NTDGCod|\!))$ $\mathfrak{C}$
    and *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}(\!|NTDGCod|\!)$ = *cat-FUNCT* α $(\mathfrak{N}(\!|NTDGDom|\!))$ $\mathfrak{C}$
  ⟨*proof*⟩

25.9.2 Natural transformation map

mk-VLambda *exp-cat-ntcf-components*(*1*)
  |*vsv exp-cat-ntcf-components-NTMap-vsv*[*cat-FUNCT-cs-intros*]|

context *is-ntcf*
begin

lemmas *exp-cat-ntcf-components′* =
exp-cat-ntcf-components[**where** α=α **and** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded cat-cs-simps*]

lemmas [*cat-FUNCT-cs-simps*] = *exp-cat-ntcf-components′*(*2–5*)

mk-VLambda *exp-cat-ntcf-components*(*1*)[**where** $\mathfrak{N}$=$\mathfrak{N}$, *unfolded cat-cs-simps*]
|*vdomain exp-cat-ntcf-components-NTMap-vdomain*[*cat-FUNCT-cs-simps*]|
|*app exp-cat-ntcf-components-NTMap-app*[*cat-FUNCT-cs-simps*]|

end

lemmas *exp-cat-ntcf-components′* = *is-ntcf.exp-cat-ntcf-components′*

lemmas [*cat-FUNCT-cs-simps*] =
is-ntcf.exp-cat-ntcf-components′(*2–5*)
is-ntcf.exp-cat-ntcf-components-NTMap-vdomain
is-ntcf.exp-cat-ntcf-components-NTMap-app

25.9.3 Category raised to the power of a natural transformation is a natural transformation

lemma *exp-cat-ntcf-is-tiny-ntcf*:
assumes $\mathcal{Z}$ β
and $\alpha \in_{\circ} \beta$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *category* α $\mathfrak{C}$
shows *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}$:
exp-cat-cf α $\mathfrak{C}$ $\mathfrak{F}$ $\mapsto_{CF.tiny}$ *exp-cat-cf* α $\mathfrak{C}$ $\mathfrak{G}$:
cat-FUNCT α $\mathfrak{B}$ $\mathfrak{C}$ $\mapsto\mapsto_{C.tiny\beta}$ *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *exp-cat-ntcf-is-tiny-ntcf′*[*cat-FUNCT-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_{\circ} \beta$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *category* α $\mathfrak{C}$
and $\mathfrak{F}'$ = *exp-cat-cf* α $\mathfrak{C}$ $\mathfrak{F}$
and $\mathfrak{G}'$ = *exp-cat-cf* α $\mathfrak{C}$ $\mathfrak{G}$
and $\mathfrak{A}'$ = *cat-FUNCT* α $\mathfrak{B}$ $\mathfrak{C}$
and $\mathfrak{B}'$ = *cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{C}$
shows *exp-cat-ntcf* α $\mathfrak{C}$ $\mathfrak{N}$: $\mathfrak{F}' \mapsto_{CF.tiny} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C.tiny\beta} \mathfrak{B}'$
⟨*proof*⟩

25.9.4 Further properties

lemma *ntcf-id-exp-cat-cf*:
assumes *category* α $\mathfrak{A}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *ntcf-id* (*exp-cat-cf* α $\mathfrak{A}$ $\mathfrak{F}$) = *exp-cat-ntcf* α $\mathfrak{A}$ (*ntcf-id* $\mathfrak{F}$)
⟨*proof*⟩

lemma *exp-cat-ntcf-ntcf-cf-comp*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and *category* α $\mathfrak{D}$
shows

exp-cat-ntcf $\alpha\ \mathfrak{D}\ (\mathfrak{N} \circ_{NTCF-CF} \mathfrak{H}) =$
exp-cat-cf $\alpha\ \mathfrak{D}\ \mathfrak{H} \circ_{CF-NTCF}$ *exp-cat-ntcf* $\alpha\ \mathfrak{D}\ \mathfrak{N}$
⟨*proof*⟩

lemma *exp-cat-ntcf-cf-ntcf-comp*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and *category* $\alpha\ \mathfrak{D}$
shows
exp-cat-ntcf $\alpha\ \mathfrak{D}\ (\mathfrak{H} \circ_{CF-NTCF} \mathfrak{N}) =$
exp-cat-ntcf $\alpha\ \mathfrak{D}\ \mathfrak{N} \circ_{NTCF-CF}$ *exp-cat-cf* $\alpha\ \mathfrak{D}\ \mathfrak{H}$
⟨*proof*⟩

lemma *exp-cat-ntcf-ntcf-vcomp*:
assumes *category* $\alpha\ \mathfrak{A}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
exp-cat-ntcf $\alpha\ \mathfrak{A}\ (\mathfrak{M} \bullet_{NTCF} \mathfrak{N}) =$
exp-cat-ntcf $\alpha\ \mathfrak{A}\ \mathfrak{M} \bullet_{NTCF}$ *exp-cat-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{N}$
⟨*proof*⟩

26 *Hom*-functor

26.1 *hom*-function

The *hom*-function is a part of the definition of the *Hom*-functor, as presented in [1][13].

definition *cf-hom* :: $V \Rightarrow V \Rightarrow V$
  where *cf-hom* $\mathfrak{C}$ f =
    [
      (
        $\lambda q \in_\circ$ *Hom* $\mathfrak{C}$ ($\mathfrak{C}(\!|Cod|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Dom|\!)(\!|vpsnd\ f|\!)$).
          vpsnd f $\circ_{A\mathfrak{C}}$ q $\circ_{A\mathfrak{C}}$ *vpfst* f
      ),
      Hom $\mathfrak{C}$ ($\mathfrak{C}(\!|Cod|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Dom|\!)(\!|vpsnd\ f|\!)$),
      Hom $\mathfrak{C}$ ($\mathfrak{C}(\!|Dom|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Cod|\!)(\!|vpsnd\ f|\!)$)
    ]$_\circ$

Components.

lemma *cf-hom-components*:
  shows *cf-hom* $\mathfrak{C}$ $f(\!|ArrVal|\!)$ =
      (
        $\lambda q \in_\circ$ *Hom* $\mathfrak{C}$ ($\mathfrak{C}(\!|Cod|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Dom|\!)(\!|vpsnd\ f|\!)$).
          vpsnd f $\circ_{A\mathfrak{C}}$ q $\circ_{A\mathfrak{C}}$ *vpfst* f
)
    and *cf-hom* $\mathfrak{C}$ $f(\!|ArrDom|\!)$ = *Hom* $\mathfrak{C}$ ($\mathfrak{C}(\!|Cod|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Dom|\!)(\!|vpsnd\ f|\!)$)
    and *cf-hom* $\mathfrak{C}$ $f(\!|ArrCod|\!)$ = *Hom* $\mathfrak{C}$ ($\mathfrak{C}(\!|Dom|\!)(\!|vpfst\ f|\!)$) ($\mathfrak{C}(\!|Cod|\!)(\!|vpsnd\ f|\!)$)
  ⟨*proof*⟩

26.1.1 Arrow value

mk-VLambda *cf-hom-components*(*1*)
  |*vsv cf-hom-ArrVal-vsv*[*cat-cs-intros*]|

lemma *cf-hom-ArrVal-vdomain*[*cat-cs-simps*]:
  assumes $g : a \mapsto_{op\text{-}cat} \mathfrak{C}\ b$ **and** $f : a' \mapsto_{\mathfrak{C}} b'$
  shows $\mathcal{D}_\circ$ (*cf-hom* $\mathfrak{C}$ [g, f]$_\circ(\!|ArrVal|\!)$) = *Hom* $\mathfrak{C}$ a a'
  ⟨*proof*⟩

lemma *cf-hom-ArrVal-app*[*cat-cs-simps*]:
  assumes $g : c \mapsto_{op\text{-}cat} \mathfrak{C}\ d$ **and** $q : c \mapsto_{\mathfrak{C}} c'$ **and** $f : c' \mapsto_{\mathfrak{C}} d'$
  shows *cf-hom* $\mathfrak{C}$ [g, f]$_\circ(\!|ArrVal|\!)(\!|q|\!)$ = f $\circ_{A\mathfrak{C}}$ q $\circ_{A\mathfrak{C}}$ g
  ⟨*proof*⟩

lemma (**in** *category*) *cf-hom-ArrVal-vrange*:
  assumes $g : a \mapsto_{op\text{-}cat} \mathfrak{C}\ b$ **and** $f : a' \mapsto_{\mathfrak{C}} b'$
  shows $\mathcal{R}_\circ$ (*cf-hom* $\mathfrak{C}$ [g, f]$_\circ(\!|ArrVal|\!)$) $\subseteq_\circ$ *Hom* $\mathfrak{C}$ b b'
⟨*proof*⟩

26.1.2 Arrow domain

lemma (**in** *category*) *cf-hom-ArrDom*:
  assumes $gf : [c, c']_\circ \mapsto_{op\text{-}cat} \mathfrak{C} \times_C \mathfrak{C}\ dd'$
  shows *cf-hom* $\mathfrak{C}$ $gf(\!|ArrDom|\!)$ = *Hom* $\mathfrak{C}$ c c'
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cf-hom-ArrDom*

[13] https://ncatlab.org/nlab/show/hom-functor

26.1.3 Arrow codomain

lemma (**in** *category*) *cf-hom-ArrCod*:
assumes $gf : cc' \mapsto_{op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C}} [d,\ d']_\circ$
shows $cf\text{-}hom\ \mathfrak{C}\ gf(\!|ArrCod|\!) = Hom\ \mathfrak{C}\ d\ d'$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cf-hom-ArrCod*

26.1.4 *hom*-function is an arrow in the category *Set*

lemma (**in** *category*) *cat-cf-hom-ArrRel*:
assumes $gf : cc' \mapsto_{op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C}} dd'$
shows $arr\text{-}Set\ \alpha\ (cf\text{-}hom\ \mathfrak{C}\ gf)$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-hom-ArrRel*

lemma (**in** *category*) *cat-cf-hom-cat-Set-is-arr*:
assumes $gf : [a,\ b]_\circ \mapsto_{op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C}} [c,\ d]_\circ$
shows $cf\text{-}hom\ \mathfrak{C}\ gf : Hom\ \mathfrak{C}\ a\ b \mapsto_{cat\text{-}Set\ \alpha} Hom\ \mathfrak{C}\ c\ d$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-hom-cat-Set-is-arr'*:
assumes $gf : [a,\ b]_\circ \mapsto_{op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C}} [c,\ d]_\circ$
and $\mathfrak{A}' = Hom\ \mathfrak{C}\ a\ b$
and $\mathfrak{B}' = Hom\ \mathfrak{C}\ c\ d$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $cf\text{-}hom\ \mathfrak{C}\ gf : \mathfrak{A}' \mapsto_{\mathfrak{C}'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-hom-cat-Set-is-arr'*

26.1.5 Composition

lemma (**in** *category*) *cat-cf-hom-Comp*:
assumes $g : b \mapsto_{op\text{-}cat\ \mathfrak{C}} c$
and $g' : b' \mapsto_{\mathfrak{C}} c'$
and $f : a \mapsto_{op\text{-}cat\ \mathfrak{C}} b$
and $f' : a' \mapsto_{\mathfrak{C}} b'$
shows
$cf\text{-}hom\ \mathfrak{C}\ [g,\ g']_\circ \circ_{A\ cat\text{-}Set\ \alpha}\ cf\text{-}hom\ \mathfrak{C}\ [f,\ f']_\circ =$
$cf\text{-}hom\ \mathfrak{C}\ [g \circ_{A\ op\text{-}cat\ \mathfrak{C}}\ f,\ g' \circ_{A\ \mathfrak{C}}\ f']_\circ$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-hom-Comp*

26.1.6 Identity

lemma (**in** *category*) *cat-cf-hom-CId*:
assumes $[c,\ c']_\circ \in_\circ (op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C})(\!|Obj|\!)$
shows $cf\text{-}hom\ \mathfrak{C}\ [\mathfrak{C}(\!|CId|\!)(\!|c|\!),\ \mathfrak{C}(\!|CId|\!)(\!|c'|\!)]_\circ = cat\text{-}Set\ \alpha(\!|CId|\!)(\!|Hom\ \mathfrak{C}\ c\ c'|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-hom-CId*

26.1.7 Opposite *hom*-function

lemma (**in** *category*) *cat-op-cat-cf-hom*:

assumes $g : a \mapsto_{\mathfrak{C}} b$ **and** $g' : a' \mapsto_{op\text{-}cat\ \mathfrak{C}} b'$
shows *cf-hom* (*op-cat* $\mathfrak{C}$) $[g, g']_\circ$ = *cf-hom* $\mathfrak{C}$ $[g', g]_\circ$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-op-cat-cf-hom*

26.2 *Hom*-functor

26.2.1 Definition and elementary properties

See [1][14].

definition *cf-Hom* :: $V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C}$1-′(/–,–/′)›)
where $Hom_{O.C\alpha}\mathfrak{C}(-,-)$ =
[
($\lambda a \in_\circ$(*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Obj*|). *Hom* $\mathfrak{C}$ (*vpfst a*) (*vpsnd a*)),
($\lambda f \in_\circ$(*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Arr*|). *cf-hom* $\mathfrak{C}$ *f*),
op-cat $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$,
cat-Set α
]$_\circ$

Components.

lemma *cf-Hom-components*:
shows $Hom_{O.C\alpha}\mathfrak{C}(-,-)(|ObjMap|)$ =
($\lambda a \in_\circ$(*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Obj*|). *Hom* $\mathfrak{C}$ (*vpfst a*) (*vpsnd a*))
and $Hom_{O.C\alpha}\mathfrak{C}(-,-)(|ArrMap|)$ = ($\lambda f \in_\circ$(*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Arr*|). *cf-hom* $\mathfrak{C}$ *f*)
and $Hom_{O.C\alpha}\mathfrak{C}(-,-)(|HomDom|)$ = *op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$
and $Hom_{O.C\alpha}\mathfrak{C}(-,-)(|HomCod|)$ = *cat-Set* α
⟨*proof*⟩

26.2.2 Object map

mk-VLambda *cf-Hom-components*(*1*)
|*vsv cf-Hom-ObjMap-vsv*|

lemma *cf-Hom-ObjMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ ($Hom_{O.C\alpha}\mathfrak{C}(-,-)(|ObjMap|)$) = (*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Obj*|)
⟨*proof*⟩

lemma *cf-Hom-ObjMap-app*[*cat-cs-simps*]:
assumes $[a, b]_\circ \in_\circ$ (*op-cat* $\mathfrak{C}$ ×$_C$ $\mathfrak{C}$)(|*Obj*|)
shows $Hom_{O.C\alpha}\mathfrak{C}(-,-)(|ObjMap|)(|a, b|)_\bullet$ = *Hom* $\mathfrak{C}$ *a b*
⟨*proof*⟩

lemma (**in** *category*) *cf-Hom-ObjMap-vrange*:
$\mathcal{R}_\circ$ ($Hom_{O.C\alpha}\mathfrak{C}(-,-)(|ObjMap|)$) $\subseteq_\circ$ *cat-Set* α(|*Obj*|)
⟨*proof*⟩

26.2.3 Arrow map

mk-VLambda *cf-Hom-components*(*2*)
|*vsv cf-Hom-ArrMap-vsv*|
|*vdomain cf-Hom-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-Hom-ArrMap-app*[*cat-cs-simps*]|

[14] https://ncatlab.org/nlab/show/hom-functor

26.2.4 *Hom*-functor is a functor

lemma (in *category*) *cat-Hom-is-functor*:
$Hom_{O.C\alpha}\mathfrak{C}(-,-) : op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-Hom-is-functor'*:
assumes $\beta = \alpha$ **and** $\mathfrak{A}' = op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{C}$ **and** $\mathfrak{B}' = cat\text{-}Set\ \alpha$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,-) : \mathfrak{A}' \mapsto\mapsto_{C\beta} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-Hom-is-functor'*

26.3 Composition of a *Hom*-functor and two functors

26.3.1 Definition and elementary properties

definition *cf-bcomp-Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C^1}$-'(/-−,-−/')›)
— The following definition may seem redundant, but it will help to avoid proof duplication later.
where $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-) = cf\text{-}cn\text{-}cov\text{-}bcomp\ (Hom_{O.C\alpha}\mathfrak{C}(-,-))\ \mathfrak{F}\ \mathfrak{G}$

26.3.2 Object map

lemma *cf-bcomp-Hom-ObjMap-vsv*: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-bcomp-Hom-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ObjMap|\!)) = (op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-bcomp-Hom-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[a, b]_\circ \in_\circ (op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B})(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ObjMap|\!)(\!|a, b|\!)_\bullet =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!), \mathfrak{G}(\!|ObjMap|\!)(\!|b|\!)|\!)_\bullet$
⟨*proof*⟩

lemma (in *category*) *cf-bcomp-Hom-ObjMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ObjMap|\!)) \subseteq_\circ cat\text{-}Set\ \alpha(\!|Obj|\!)$
⟨*proof*⟩

26.3.3 Arrow map

lemma *cf-bcomp-Hom-ArrMap-vsv*: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ArrMap|\!))$
⟨*proof*⟩

lemma *cf-bcomp-Hom-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ArrMap|\!)) = (op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B})(\!|Arr|\!)$
⟨*proof*⟩

lemma *cf-bcomp-Hom-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $[f, g]_\circ \in_\circ (op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B})(\!|Arr|\!)$

shows
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ArrMap|\!)(\!|f,\ g|\!)_{\bullet} =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!),\ \mathfrak{G}(\!|ArrMap|\!)(\!|g|\!)|\!)_{\bullet}$
⟨*proof*⟩

lemma (**in** *category*) *cf-bcomp-Hom-ArrMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-)(\!|ArrMap|\!)) \subseteq_{\circ} cat\text{-}Set\ \alpha(\!|Arr|\!)$
⟨*proof*⟩

26.3.4 Composition of a *Hom*-functor and two functors is a functor

lemma (**in** *category*) *cat-cf-bcomp-Hom-is-functor*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-) : op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-bcomp-Hom-is-functor′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{B}$
and $\mathfrak{B}' = cat\text{-}Set\ \alpha$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}-) : \mathfrak{A}' \mapsto\mapsto_{C\beta} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-bcomp-Hom-is-functor′*

26.4 Composition of a *Hom*-functor and a functor

26.4.1 Definition and elementary properties

See subsection 1.15 in [3].

definition *cf-lcomp-Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C^1}$-′(/--,-/′)›)
where $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) = cf\text{-}cn\text{-}cov\text{-}lcomp\ \mathfrak{C}\ (Hom_{O.C\alpha}\mathfrak{C}(-,-))\ \mathfrak{F}$

definition *cf-rcomp-Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C^1}$-′(/-,--/′)›)
where $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-) = cf\text{-}cn\text{-}cov\text{-}rcomp\ \mathfrak{C}\ (Hom_{O.C\alpha}\mathfrak{C}(-,-))\ \mathfrak{G}$

26.4.2 Object map

lemma *cf-lcomp-Hom-ObjMap-vsv*[*cat-cs-intros*]: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ObjMap-vsv*[*cat-cs-intros*]: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\!|ObjMap|\!))$
⟨*proof*⟩

lemma *cf-lcomp-Hom-ObjMap-vdomain*[*cat-cs-simps*]:
assumes *category* $\alpha\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\!|ObjMap|\!)) = (op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\!|ObjMap|\!)) = (op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

lemma *cf-lcomp-Hom-ObjMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ$ *op-cat* $\mathfrak{B}(\lfloor Obj \rfloor)$
and $c \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ObjMap \rfloor)(\lfloor b,\ c \rfloor)_\bullet =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\lfloor ObjMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor),\ c \rfloor)_\bullet$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $c \in_\circ$ *op-cat* $\mathfrak{C}(\lfloor Obj \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ObjMap \rfloor)(\lfloor c,\ b \rfloor)_\bullet =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\lfloor ObjMap \rfloor)(\lfloor c,\ \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor) \rfloor)_\bullet$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-lcomp-Hom-ObjMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ObjMap \rfloor)) \subseteq_\circ$ *cat-Set* $\alpha(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-rcomp-Hom-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ObjMap \rfloor)) \subseteq_\circ$ *cat-Set* $\alpha(\lfloor Obj \rfloor)$
⟨*proof*⟩

26.4.3 Arrow map

lemma *cf-lcomp-Hom-ArrMap-vsv*[*cat-cs-intros*]: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ArrMap \rfloor))$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ArrMap-vsv*[*cat-cs-intros*]: *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ArrMap \rfloor))$
⟨*proof*⟩

lemma *cf-lcomp-Hom-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ArrMap \rfloor)) = (op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C})(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ArrMap-vdomain*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ArrMap \rfloor)) = (op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B})(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *cf-lcomp-Hom-ArrMap-app*[*cat-cs-simps*]:
assumes *category* α $\mathfrak{C}$
and $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $g : a \mapsto_{op\text{-}cat\ \mathfrak{B}} b$
and $f : a' \mapsto_{\mathfrak{C}} b'$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ArrMap \rfloor)(\lfloor g,\ f \rfloor)_\bullet =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor),\ f \rfloor)_\bullet$
⟨*proof*⟩

lemma *cf-rcomp-Hom-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $g : a \mapsto_{op\text{-}cat\ \mathfrak{C}} b$
and $f : a' \mapsto_{\mathfrak{B}} b'$

shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ArrMap \rfloor)(\lfloor g,\ f \rfloor)_{\bullet} =$
$Hom_{O.C\alpha}\mathfrak{C}(-,-)(\lfloor ArrMap \rfloor)(\lfloor g,\ \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \rfloor)_{\bullet}$
⟨*proof*⟩

lemma (**in** *category*) *cf-lcomp-Hom-ArrMap-vrange*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)(\lfloor ArrMap \rfloor)) \subseteq_{\circ} cat\text{-}Set\ \alpha(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma (**in** *category*) *cf-rcomp-Hom-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)(\lfloor ArrMap \rfloor)) \subseteq_{\circ} cat\text{-}Set\ \alpha(\lfloor Arr \rfloor)$
⟨*proof*⟩

26.4.4 Further properties

lemma *cf-bcomp-Hom-cf-lcomp-Hom*[*cat-cs-simps*]:
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,cf\text{-}id\ \mathfrak{C}-) = Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$
⟨*proof*⟩

lemma *cf-bcomp-Hom-cf-rcomp-Hom*[*cat-cs-simps*]:
$Hom_{O.C\alpha}\mathfrak{C}(cf\text{-}id\ \mathfrak{C}-,\mathfrak{G}-) = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$
⟨*proof*⟩

26.4.5 Composition of a *Hom*-functor and a functor is a functor

lemma (**in** *category*) *cat-cf-lcomp-Hom-is-functor*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) : op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-lcomp-Hom-is-functor′*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C}$
and $\mathfrak{B}' = cat\text{-}Set\ \alpha$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) : \mathfrak{A}' \mapsto\mapsto_{C\beta} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-lcomp-Hom-is-functor′*

lemma (**in** *category*) *cat-cf-rcomp-Hom-is-functor*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-) : op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-rcomp-Hom-is-functor′*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\beta = \alpha$
and $\mathfrak{A}' = op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B}$
and $\mathfrak{B}' = cat\text{-}Set\ \alpha$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-) : \mathfrak{A}' \mapsto\mapsto_{C\beta} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-rcomp-Hom-is-functor′*

26.4.6 Flip of a projections of a *Hom*-functor

lemma (**in** *category*) *cat-bifunctor-flip-cf-rcomp-Hom*:

assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
bifunctor-flip (*op-cat* $\mathfrak{C}$) $\mathfrak{B}$ ($Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$) =
$Hom_{O.C\alpha}$*op-cat* $\mathfrak{C}$(*op-cf* $\mathfrak{G}-$,$-$)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-bifunctor-flip-cf-rcomp-Hom*

lemma (**in** *category*) *cat-bifunctor-flip-cf-lcomp-Hom*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
bifunctor-flip (*op-cat* $\mathfrak{B}$) $\mathfrak{C}$ ($Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$) =
$Hom_{O.C\alpha}$*op-cat* $\mathfrak{C}$($-$,*op-cf* $\mathfrak{F}-$)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-bifunctor-flip-cf-lcomp-Hom*

26.5 Projections of the *Hom*-functor

The projections of the *Hom*-functor coincide with the definitions of the *Hom*-functor given in Chapter II-2 in [7]. They are also exposed in the aforementioned article in nLab [1][15].

26.5.1 Definitions and elementary properties

definition *cf-Hom-snd* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C^{1}}$-′(/-,-/′)›)
where $Hom_{O.C\alpha}\mathfrak{C}(a,-) = Hom_{O.C\alpha}\mathfrak{C}(-,-)_{op\text{-}cat\ \mathfrak{C},\mathfrak{C}}(a,-)_{CF}$
definition *cf-Hom-fst* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{O.C^{1}}$-′(/-,-/′)›)
where $Hom_{O.C\alpha}\mathfrak{C}(-,b) = Hom_{O.C\alpha}\mathfrak{C}(-,-)_{op\text{-}cat\ \mathfrak{C},\mathfrak{C}}(-,b)_{CF}$

26.5.2 Projections of the *Hom*-functor are functors

lemma (**in** *category*) *cat-cf-Hom-snd-is-functor*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(a,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-Hom-snd-is-functor′*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\beta = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$ **and** $\mathfrak{D}' =$ *cat-Set* α
shows $Hom_{O.C\alpha}\mathfrak{C}(a,-) : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-snd-is-functor′*

lemma (**in** *category*) *cat-cf-Hom-fst-is-functor*:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,b)$: *op-cat* $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-Hom-fst-is-functor′*:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\beta = \alpha$ **and** $\mathfrak{C}' =$ *op-cat* $\mathfrak{C}$ **and** $\mathfrak{D}' =$ *cat-Set* α
shows $Hom_{O.C\alpha}\mathfrak{C}(-,b) : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-fst-is-functor′*

[15] https://ncatlab.org/nlab/show/hom-functor

26.5.3 Object maps

lemma (**in** *category*) *cat-cf-Hom-snd-ObjMap-vsv*[*cat-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ObjMap|\!))$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-snd-ObjMap-vsv*

lemma (**in** *category*) *cat-cf-Hom-fst-ObjMap-vsv*[*cat-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ObjMap|\!))$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-fst-ObjMap-vsv*

lemma (**in** *category*) *cat-cf-Hom-snd-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}$ $(Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ObjMap|\!)) = \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-snd-ObjMap-vdomain*

lemma (**in** *category*) *cat-cf-Hom-fst-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}$ $(Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ObjMap|\!)) = op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-fst-ObjMap-vdomain*

lemma (**in** *category*) *cat-cf-Hom-snd-ObjMap-app*[*cat-cs-simps*]:
assumes $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ObjMap|\!)(\!|b|\!) = Hom\ \mathfrak{C}\ a\ b$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-snd-ObjMap-app*

lemma (**in** *category*) *cat-cf-Hom-fst-ObjMap-app*[*cat-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ObjMap|\!)(\!|a|\!) = Hom\ \mathfrak{C}\ a\ b$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-fst-ObjMap-app*

26.5.4 Arrow maps

lemma (**in** *category*) *cat-cf-Hom-snd-ArrMap-vsv*[*cat-cs-intros*]:
assumes $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
shows *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ArrMap|\!))$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-snd-ArrMap-vsv*

lemma (**in** *category*) *cat-cf-Hom-fst-ArrMap-vsv*[*cat-cs-intros*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *vsv* $(Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ArrMap|\!))$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-Hom-fst-ArrMap-vsv*

lemma (in *category*) *cat-cf-Hom-snd-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ArrMap|\!)) = \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-snd-ArrMap-vdomain*

lemma (in *category*) *cat-cf-Hom-fst-ArrMap-vdomain*[*cat-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}\ (Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ArrMap|\!)) = op\text{-}cat\ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-fst-ArrMap-vdomain*

lemma (in *category*) *cat-cf-Hom-snd-ArrMap-app*[*cat-cs-simps*]:
assumes $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$ **and** $f : b \mapsto_{\mathfrak{C}} b'$
shows $Hom_{O.C\alpha}\mathfrak{C}(a,-)(\!|ArrMap|\!)(\!|f|\!) = cf\text{-}hom\ \mathfrak{C}\ [op\text{-}cat\ \mathfrak{C}(\!|CId|\!)(\!|a|\!),\ f]_{\circ}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-snd-ArrMap-app*

lemma (in *category*) *cat-cf-Hom-fst-ArrMap-app*[*cat-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $f : a \mapsto_{op\text{-}cat\ \mathfrak{C}} a'$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,b)(\!|ArrMap|\!)(\!|f|\!) = cf\text{-}hom\ \mathfrak{C}\ [f,\ \mathfrak{C}(\!|CId|\!)(\!|b|\!)]_{\circ}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-Hom-fst-ArrMap-app*

26.5.5 Opposite *Hom*-functor projections

lemma (in *category*) *cat-op-cat-cf-Hom-snd*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}op\text{-}cat\ \mathfrak{C}(a,-) = Hom_{O.C\alpha}\mathfrak{C}(-,a)$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.cat-op-cat-cf-Hom-snd*

lemma (in *category*) *cat-op-cat-cf-Hom-fst*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}op\text{-}cat\ \mathfrak{C}(-,a) = Hom_{O.C\alpha}\mathfrak{C}(a,-)$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.cat-op-cat-cf-Hom-fst*

26.5.6 *Hom*-functors are injections on objects

lemma (in *category*) *cat-cf-Hom-snd-inj*:
assumes $Hom_{O.C\alpha}\mathfrak{C}(a,-) = Hom_{O.C\alpha}\mathfrak{C}(b,-)$
and $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $a = b$
⟨*proof*⟩

lemma (in *category*) *cat-cf-Hom-fst-inj*:
assumes $Hom_{O.C\alpha}\mathfrak{C}(-,a) = Hom_{O.C\alpha}\mathfrak{C}(-,b)$ **and** $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $a = b$
⟨*proof*⟩

26.5.7 *Hom*-functor is an array bifunctor

lemma (in *category*) *cat-cf-Hom-is-cf-array*:
— See Chapter II-3 in [7].
$Hom_{O.C\alpha}\mathfrak{C}(-,-) =$
cf-array (*op-cat* $\mathfrak{C}$) $\mathfrak{C}$ (*cat-Set* α) (*cf-Hom-fst* α $\mathfrak{C}$) (*cf-Hom-snd* α $\mathfrak{C}$)
⟨*proof*⟩

26.5.8 Projections of the compositions of a *Hom*-functor and a functor are projections of the *Hom*-functor

lemma (in *category*) *cat-cf-rcomp-Hom-cf-Hom-snd*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)_{op\text{-}cat\ \mathfrak{C},\mathfrak{B}}(a,-)_{CF} = Hom_{O.C\alpha}\mathfrak{C}(a,-) \circ_{CF} \mathfrak{G}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-rcomp-Hom-cf-Hom-snd*

lemma (in *category*) *cat-cf-lcomp-Hom-cf-Hom-snd*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)_{op\text{-}cat\ \mathfrak{B},\mathfrak{C}}(b,-)_{CF} = Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!),-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-lcomp-Hom-cf-Hom-snd*

lemma (in *category*) *cat-cf-rcomp-Hom-cf-Hom-fst*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{F}-)_{op\text{-}cat\ \mathfrak{C},\mathfrak{B}}(-,b)_{CF} = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-rcomp-Hom-cf-Hom-fst*

27 Cones and cocones

27.1 Cone and cocone

27.1.1 Definition and elementary properties

In the context of this work, the concept of a cone corresponds to that of a cone to the base of a functor from a vertex, as defined in Chapter III-4 in [7]; the concept of a cocone corresponds to that of a cone from the base of a functor to a vertex, as defined in Chapter III-3 in [7].

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

syntax *-is-cat-cone* :: *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *bool*
  (‹(- :/ - <$_{CF.cone}$ - :/ - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-cone* ⇌ *is-cat-cone*
translations $\mathfrak{N}$: *c* <$_{CF.cone}$ $\mathfrak{F}$: $\mathfrak{I}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-cone α *c* $\mathfrak{I}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

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

syntax *-is-cat-cocone* :: *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *V* ⇒ *bool*
  (‹(- :/ - >$_{CF.cocone}$ - :/ - ↦↦$_{C1}$ -)› [*51*, *51*, *51*, *51*, *51*] *51*)
syntax-consts *-is-cat-cocone* ⇌ *is-cat-cocone*
translations $\mathfrak{N}$: $\mathfrak{F}$ >$_{CF.cocone}$ *c* : $\mathfrak{I}$ ↦↦$_{C\alpha}$ $\mathfrak{C}$ ⇌
  CONST is-cat-cocone α *c* $\mathfrak{I}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

Rules.

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

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

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

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

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

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

Duality.

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

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

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

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

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

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

Elementary properties.

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

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

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

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

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

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

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

thm *is-cat-cone.cat-cone-Comp-commute*

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

⟨*proof*⟩

thm *is-cat-cocone.cat-cocone-Comp-commute*

Utilities/helper lemmas.

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

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

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

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

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

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

27.1.2 Vertical composition of a natural transformation and a cone

lemma *ntcf-vcomp-is-cat-cone*[*cat-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{N} : a <_{CF.cone} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : a <_{CF.cone} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

27.1.3 Composition of a functor and a cone, composition of a functor and a cocone

lemma *cf-ntcf-comp-cf-cat-cone*:

assumes $\mathfrak{N} : c <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G}(\!|ObjMap|\!)(\!|c|\!) <_{CF.cone} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-cf-cat-cone'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : c <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $c' = \mathfrak{G}(\!|ObjMap|\!)(\!|c|\!)$
and $\mathfrak{G}\mathfrak{F} = \mathfrak{G} \circ_{CF} \mathfrak{F}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : c' <_{CF.cone} \mathfrak{G}\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-cf-cat-cocone*:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.cocone} c : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G} \circ_{CF} \mathfrak{F} >_{CF.cocone} \mathfrak{G}(\!|ObjMap|\!)(\!|c|\!) : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-cf-cat-cocone'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.cocone} c : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $c' = \mathfrak{G}(\!|ObjMap|\!)(\!|c|\!)$
and $\mathfrak{G}\mathfrak{F} = \mathfrak{G} \circ_{CF} \mathfrak{F}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G}\mathfrak{F} >_{CF.cocone} c' : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

27.1.4 Cones, cocones and constant natural transformations

lemma *ntcf-vcomp-ntcf-const-is-cat-cone*:
assumes $\mathfrak{N} : b <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{N} \bullet_{NTCF} \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f : a <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-cat-cone'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : b <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{M} = \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f$
and $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{N} \bullet_{NTCF} \mathfrak{M} : a <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-cat-cocone*:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.cocone} a : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $f : a \mapsto_{\mathfrak{B}} b$
shows $\textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} >_{CF.cocone} b : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-cat-cocone'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.cocone} a : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
and $\mathfrak{M} = \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f$
and $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} >_{CF.cocone} b : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-CId*:
assumes $\mathfrak{N} : b <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$
shows $\mathfrak{N} \bullet_{NTCF} \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ (\mathfrak{B}(\!|CId|\!)(\!|b|\!)) = \mathfrak{N}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-CId'*[*cat-cs-simps*]:
assumes $\mathfrak{N} : b <_{CF.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{B}$ **and** $\mathfrak{B}' = \mathfrak{B}$

shows $\mathfrak{N} \cdot_{NTCF}$ *ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ ($\mathfrak{B}'(\!|CId|\!)(\!|b|\!)$) = $\mathfrak{N}$
⟨*proof*⟩

27.2 Cone and cocone functors

27.2.1 Definition and elementary properties

See Chapter V-1 in [7].

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

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

An alternative form of the definition.

context *is-functor*
begin

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

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

end

27.2.2 Object map

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

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

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

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

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

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

lemma (**in** *is-functor*) *cf-Cocone-ObjMap-vdomain*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$ **and** $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$

shows $\mathcal{D}_\circ$ (*cf-Cocone* α β $\mathfrak{F}(\lfloor ObjMap \rfloor)$) = $\mathfrak{B}(\lfloor Obj \rfloor)$
⟨*proof*⟩

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

lemma (**in** *is-functor*) *cf-Cone-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$ **and** $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows *cf-Cone* α β $\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$ =
Hom (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* (*cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b)) (*cf-map* $\mathfrak{F}$)
⟨*proof*⟩

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

lemma (**in** *is-functor*) *cf-Cocone-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$ **and** $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows *cf-Cocone* α β $\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$ =
Hom (*cat-FUNCT* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* $\mathfrak{F}$) (*cf-map* (*cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b))
⟨*proof*⟩

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

27.2.3 Arrow map

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

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

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

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

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

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

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

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

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

⟨*proof*⟩

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

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

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

27.2.4 The cone functor is a functor

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

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

28 Smallness for cones and cocones

28.1 Cone with tiny maps and cocone with tiny maps

28.1.1 Definition and elementary properties

locale *is-tm-cat-cone* =
  is-ntcf α $\mathfrak{J}$ $\mathfrak{C}$ ‹*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ c› $\mathfrak{F}$ $\mathfrak{N}$ + *NTCod*: *is-tm-functor* α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$
  for α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$ +
  assumes *tm-cat-cone-obj*[*cat-cs-intros*, *cat-small-cs-intros*]: $c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$

syntax *-is-tm-cat-cone* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (‹(- :/ - $<_{CF.tm.cone}$ - :/ - $\mapsto\mapsto_{C.tm1}$ -)› [51, 51, 51, 51, 51] 51)
syntax-consts *-is-tm-cat-cone* $\rightleftharpoons$ *is-tm-cat-cone*
translations $\mathfrak{N}$: $c <_{CF.tm.cone} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \rightleftharpoons$
  CONST is-tm-cat-cone α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

locale *is-tm-cat-cocone* =
  is-ntcf α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ ‹*cf-const* $\mathfrak{J}$ $\mathfrak{C}$ c› $\mathfrak{N}$ + *NTDom*: *is-tm-functor* α $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$
  for α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$ +
  assumes *tm-cat-cocone-obj*[*cat-cs-intros*, *cat-small-cs-intros*]: $c \in_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$

syntax *-is-tm-cat-cocone* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (‹(- :/ - $>_{CF.tm.cocone}$ - :/ - $\mapsto\mapsto_{C.tm1}$ -)› [51, 51, 51, 51, 51] 51)
syntax-consts *-is-tm-cat-cocone* $\rightleftharpoons$ *is-tm-cat-cocone*
translations $\mathfrak{N}$: $\mathfrak{F} >_{CF.tm.cocone} c : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C} \rightleftharpoons$
  CONST is-tm-cat-cocone α c $\mathfrak{J}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{N}$

Rules.

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

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

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

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

Duality.

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

lemma (**in** *is-tm-cat-cone*) *is-tm-cat-cocone-op'*[*cat-op-intros*]:

assumes $\alpha' = \alpha$ **and** $\mathfrak{J}' = \textit{op-cat}\ \mathfrak{J}$ **and** $\mathfrak{C}' = \textit{op-cat}\ \mathfrak{C}$ **and** $\mathfrak{F}' = \textit{op-cf}\ \mathfrak{F}$
shows $\textit{op-ntcf}\ \mathfrak{N} : \mathfrak{F}' >_{CF.tm.cocone} c : \mathfrak{J}' \mapsto\mapsto_{C.tm_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

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

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

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

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

Elementary properties.

lemma (**in** *is-tm-cat-cone*) *tm-cat-cone-is-tm-ntcf'*[
cat-cs-intros, *cat-small-cs-intros*
]:
assumes $c' = \textit{cf-const}\ \mathfrak{J}\ \mathfrak{C}\ c$
shows $\mathfrak{N} : c' \mapsto_{CF.tm} \mathfrak{F} : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-cat-cone.tm-cat-cone-is-tm-ntcf'*

sublocale *is-tm-cat-cone* ⊆ *is-tm-ntcf* $\alpha\ \mathfrak{J}\ \mathfrak{C}$ ‹*cf-const* $\mathfrak{J}\ \mathfrak{C}\ c$› $\mathfrak{F}\ \mathfrak{N}$
⟨*proof*⟩

lemma (**in** *is-tm-cat-cocone*) *tm-cat-cocone-is-tm-ntcf'*[
cat-cs-intros, *cat-small-cs-intros*
]:
assumes $c' = \textit{cf-const}\ \mathfrak{J}\ \mathfrak{C}\ c$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{CF.tm} c' : \mathfrak{J} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*, *cat-cs-intros*] =
is-tm-cat-cocone.tm-cat-cocone-is-tm-ntcf'

sublocale *is-tm-cat-cocone* ⊆ *is-tm-ntcf* $\alpha\ \mathfrak{J}\ \mathfrak{C}\ \mathfrak{F}$ ‹*cf-const* $\mathfrak{J}\ \mathfrak{C}\ c$› $\mathfrak{N}$
⟨*proof*⟩

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

lemmas (**in** *is-tm-cat-cone*) *tm-cat-cone-is-cat-cone* = *is-cat-cone-axioms*
lemmas [*cat-small-cs-intros*] = *is-tm-cat-cone.tm-cat-cone-is-cat-cone*

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

lemmas (**in** *is-tm-cat-cocone*) *tm-cat-cocone-is-cat-cocone* = *is-cat-cocone-axioms*
lemmas [*cat-small-cs-intros*] = *is-tm-cat-cocone.tm-cat-cocone-is-cat-cocone*

28.1.2 Vertical composition of a natural transformation with tiny maps and a cone with tiny maps

lemma *ntcf-vcomp-is-tm-cat-cone*[*cat-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF.tm} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{N} : a <_{CF.tm.cone} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : a <_{CF.tm.cone} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

28.1.3 Composition of a functor and a cone with tiny maps, composition of a functor and a cocone with tiny maps

lemma *cf-ntcf-comp-tm-cf-tm-cat-cone*:
assumes $\mathfrak{N} : c <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor) <_{CF.tm.cone} \mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-tm-cf-tm-cat-cone'*[*cat-small-cs-intros*]:
assumes $\mathfrak{N} : c <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $c' = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor)$
and $\mathfrak{G}\mathfrak{F} = \mathfrak{G} \circ_{CF} \mathfrak{F}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : c' <_{CF.tm.cone} \mathfrak{G}\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-tm-cf-tm-cat-cocone*:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} c : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G} \circ_{CF} \mathfrak{F} >_{CF.tm.cocone} \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor) : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *cf-ntcf-comp-tm-cf-tm-cat-cocone'*[*cat-small-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} c : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
and $c' = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor c \rfloor)$
and $\mathfrak{G}\mathfrak{F} = \mathfrak{G} \circ_{CF} \mathfrak{F}$
shows $\mathfrak{G} \circ_{CF-NTCF} \mathfrak{N} : \mathfrak{G}\mathfrak{F} >_{CF.tm.cocone} c' : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

28.1.4 Cones and cocones with tiny maps and constant natural transformations

lemma *ntcf-vcomp-ntcf-const-is-tm-cat-cone*:
assumes $\mathfrak{N} : b <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$ **and** $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{N} \bullet_{NTCF} \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f : a <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-tm-cat-cone'*[*cat-small-cs-intros*]:
assumes $\mathfrak{N} : b <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{M} = \textit{ntcf-const}\ \mathfrak{A}\ \mathfrak{B}\ f$
and $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{N} \bullet_{NTCF} \mathfrak{M} : a <_{CF.tm.cone} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-tm-cat-cocone*:

assumes $\mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} a : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$ **and** $f : a \mapsto_{\mathfrak{B}} b$
shows *ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ $f \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} b : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntcf-vcomp-ntcf-const-is-tm-cat-cocone'*[*cat-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} a : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
and $\mathfrak{M} =$ *ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ f
and $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{M} \bullet_{NTCF} \mathfrak{N} : \mathfrak{F} >_{CF.tm.cocone} b : \mathfrak{A} \mapsto\mapsto_{C.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

28.2 Small cone and small cocone functors

28.2.1 Definition and elementary properties

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

definition *tm-cf-Cocone* :: $V \Rightarrow V \Rightarrow V$
where *tm-cf-Cocone* α $\mathfrak{F}$ =
$Hom_{O.C\alpha}$*cat-Funct* α ($\mathfrak{F}$(|*HomDom*|)) ($\mathfrak{F}$(|*HomCod*|))(*cf-map* $\mathfrak{F}$,−) $\circ_{CF}$
($\Delta_{CF.tm}$ α ($\mathfrak{F}$(|*HomDom*|)) ($\mathfrak{F}$(|*HomCod*|)))

Alternative definitions.

context *is-tm-functor*
begin

lemma *tm-cf-Cone-def'*:
tm-cf-Cone α $\mathfrak{F}$ =
$Hom_{O.C\alpha}$*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$(−,*cf-map* $\mathfrak{F}$) $\circ_{CF}$ *op-cf* ($\Delta_{CF.tm}$ α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *tm-cf-Cocone-def'*:
tm-cf-Cocone α $\mathfrak{F}$ =
$Hom_{O.C\alpha}$*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$(*cf-map* $\mathfrak{F}$,−) $\circ_{CF}$ ($\Delta_{CF.tm}$ α $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

end

28.2.2 Object map

lemma (**in** *is-tm-functor*) *tm-cf-Cone-ObjMap-vsv*[*cat-small-cs-intros*]:
vsv (*tm-cf-Cone* α $\mathfrak{F}$(|*ObjMap*|))
⟨*proof*⟩

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

lemma (**in** *is-tm-functor*) *tm-cf-Cocone-ObjMap-vsv*[*cat-small-cs-intros*]:
vsv (*tm-cf-Cocone* α $\mathfrak{F}$(|*ObjMap*|))
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functor.tm-cf-Cocone-ObjMap-vsv*

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

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

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

lemmas [*cat-small-cs-simps*] = *is-tm-functor.tm-cf-Cocone-ObjMap-vdomain*

lemma (**in** *is-tm-functor*) *tm-cf-Cone-ObjMap-app*[*cat-small-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *tm-cf-Cone* α $\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$ =
Hom (*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* (*cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b)) (*cf-map* $\mathfrak{F}$)
⟨*proof*⟩

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

lemma (**in** *is-tm-functor*) *tm-cf-Cocone-ObjMap-app*[*cat-small-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *tm-cf-Cocone* α $\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$ =
Hom (*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$) (*cf-map* $\mathfrak{F}$) (*cf-map* (*cf-const* $\mathfrak{A}$ $\mathfrak{B}$ b))
⟨*proof*⟩

lemmas [*cat-small-cs-simps*] = *is-tm-functor.tm-cf-Cocone-ObjMap-app*

28.2.3 Arrow map

lemma (**in** *is-tm-functor*) *tm-cf-Cone-ArrMap-vsv*[*cat-small-cs-intros*]:
vsv (*tm-cf-Cone* α $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

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

lemma (**in** *is-tm-functor*) *tm-cf-Cocone-ArrMap-vsv*[*cat-small-cs-intros*]:
vsv (*tm-cf-Cocone* α $\mathfrak{F}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functor.tm-cf-Cocone-ArrMap-vsv*

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

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

lemma (**in** *is-tm-functor*) *tm-cf-Cocone-ArrMap-vdomain*[*cat-small-cs-simps*]:
assumes $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}$ (*tm-cf-Cocone* α $\mathfrak{F}(\!|ArrMap|\!)$) = $\mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*cat-small-cs-simps*] = *is-tm-functor.tm-cf-Cocone-ArrMap-vdomain*

lemma (**in** *is-tm-functor*) *tm-cf-Cone-ArrMap-app*[*cat-small-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows *tm-cf-Cone* α $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)$ = *cf-hom*
(*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$)

[*ntcf-arrow* (*ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ f), *cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$(|*CId*|)(|*cf-map* $\mathfrak{F}$|)]$_\circ$
⟨*proof*⟩

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

lemma (**in** *is-tm-functor*) *tm-cf-Cocone-ArrMap-app*[*cat-small-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows *tm-cf-Cocone* α $\mathfrak{F}$(|*ArrMap*|)(|f|) = *cf-hom*
(*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$)
[*cat-Funct* α $\mathfrak{A}$ $\mathfrak{B}$(|*CId*|)(|*cf-map* $\mathfrak{F}$|), *ntcf-arrow* (*ntcf-const* $\mathfrak{A}$ $\mathfrak{B}$ f)]$_\circ$
⟨*proof*⟩

lemmas [*cat-small-cs-simps*] = *is-tm-functor.tm-cf-Cocone-ArrMap-app*

28.2.4 Small cone functor and small cocone functor are functors

lemma (**in** *is-tm-functor*) *tm-cf-cf-Cone-is-functor*:
tm-cf-Cone α $\mathfrak{F}$: *op-cat* $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-cf-Cone-is-functor'*[*cat-small-cs-intros*]:
assumes $\mathfrak{A}'$ = *op-cat* $\mathfrak{B}$ **and** $\mathfrak{B}'$ = *cat-Set* α **and** α' = α
shows *tm-cf-Cone* α $\mathfrak{F}$: $\mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functor.tm-cf-cf-Cone-is-functor'*

lemma (**in** *is-tm-functor*) *tm-cf-cf-Cocone-is-functor*:
tm-cf-Cocone α $\mathfrak{F}$: $\mathfrak{B}$ $\mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

lemma (**in** *is-tm-functor*) *tm-cf-cf-Cocone-is-functor'*[*cat-small-cs-intros*]:
assumes $\mathfrak{B}'$ = *cat-Set* α **and** α' = α
shows *tm-cf-Cocone* α $\mathfrak{F}$: $\mathfrak{B} \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-small-cs-intros*] = *is-tm-functor.tm-cf-cf-Cocone-is-functor'*

29 Yoneda Lemma

29.1 Yoneda map

The Yoneda map is the bijection that is used in the statement of the Yoneda Lemma, as presented, for example, in Chapter III-2 in [7] or in subsection 1.15 in [3].

definition *Yoneda-map* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *Yoneda-map* α $\mathfrak{K}$ r =
    (
      $\lambda\psi\in_\circ$*these-ntcfs* α ($\mathfrak{K}$(|*HomDom*|)) (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{K}$(|*HomDom*|)($r$,−) $\mathfrak{K}$.
        ψ(|*NTMap*|)(|r|)(|*ArrVal*|)(|$\mathfrak{K}$(|*HomDom*|)(|*CId*|)(|r|)|)
)

Elementary properties.

mk-VLambda *Yoneda-map-def*
  |*vsv Yoneda-map-vsv*[*cat-cs-intros*]|

mk-VLambda (**in** *is-functor*) *Yoneda-map-def*[**where** α=α **and** $\mathfrak{K}$=$\mathfrak{F}$, *unfolded cf-HomDom*]
  |*vdomain Yoneda-map-vdomain*|
  |*app Yoneda-map-app*[*unfolded these-ntcfs-iff*]|

lemmas [*cat-cs-simps*] = *is-functor.Yoneda-map-vdomain*

lemmas *Yoneda-map-app*[*cat-cs-simps*] =
  is-functor.Yoneda-map-app[*unfolded these-ntcfs-iff*]

29.2 Yoneda component

29.2.1 Definition and elementary properties

The Yoneda components are the components of the natural transformations that appear in the statement of the Yoneda Lemma (e.g., see Chapter III-2 in [7] or subsection 1.15 in [3]).

definition *Yoneda-component* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *Yoneda-component* $\mathfrak{K}$ r u d =
    [
      ($\lambda f\in_\circ Hom$ ($\mathfrak{K}$(|*HomDom*|)) r d. $\mathfrak{K}$(|*ArrMap*|)(|f|)(|*ArrVal*|)(|u|)),
      Hom ($\mathfrak{K}$(|*HomDom*|)) r d,
      $\mathfrak{K}$(|*ObjMap*|)(|d|)
    ]$_\circ$

Components.

lemma (**in** *is-functor*) *Yoneda-component-components*:
  shows *Yoneda-component* $\mathfrak{F}$ r u d(|*ArrVal*|) =
    ($\lambda f\in_\circ Hom$ $\mathfrak{A}$ r d. $\mathfrak{F}$(|*ArrMap*|)(|f|)(|*ArrVal*|)(|u|))
    and *Yoneda-component* $\mathfrak{F}$ r u d(|*ArrDom*|) = *Hom* $\mathfrak{A}$ r d
    and *Yoneda-component* $\mathfrak{F}$ r u d(|*ArrCod*|) = $\mathfrak{F}$(|*ObjMap*|)(|d|)
  ⟨*proof*⟩

29.2.2 Arrow value

mk-VLambda (**in** *is-functor*) *Yoneda-component-components*(*1*)
  |*vsv Yoneda-component-ArrVal-vsv*|
  |*vdomain Yoneda-component-ArrVal-vdomain*|
  |*app Yoneda-component-ArrVal-app*[*unfolded in-Hom-iff*]|

lemmas [*cat-cs-simps*] = *is-functor.Yoneda-component-ArrVal-vdomain*

lemmas *Yoneda-component-ArrVal-app*[*cat-cs-simps*] =
is-functor.Yoneda-component-ArrVal-app[*unfolded in-Hom-iff*]

29.2.3 Yoneda component is an arrow in the category *Set*

lemma (**in** *category*) *cat-Yoneda-component-is-arr*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $r \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $u \in_\circ \mathfrak{K}(\!|ObjMap|\!)(\!|r|\!)$
and $d \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *Yoneda-component* $\mathfrak{K}$ r u d : *Hom* $\mathfrak{C}$ r $d \mapsto_{cat\text{-}Set\ \alpha} \mathfrak{K}(\!|ObjMap|\!)(\!|d|\!)$
⟨*proof*⟩

lemma (**in** *category*) *cat-Yoneda-component-is-arr′*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $r \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $u \in_\circ \mathfrak{K}(\!|ObjMap|\!)(\!|r|\!)$
and $d \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $s = Hom\ \mathfrak{C}\ r\ d$
and $t = \mathfrak{K}(\!|ObjMap|\!)(\!|d|\!)$
and $\mathfrak{D} = cat\text{-}Set\ \alpha$
shows *Yoneda-component* $\mathfrak{K}$ r u d : $s \mapsto_{\mathfrak{D}} t$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-Yoneda-component-is-arr′*[*rotated 1*]

29.3 Yoneda arrow

29.3.1 Definition and elementary properties

The Yoneda arrows are the natural transformations that appear in the statement of the Yoneda Lemma in Chapter III-2 in [7] and subsection 1.15 in [3].

definition *Yoneda-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *Yoneda-arrow* α $\mathfrak{K}$ r u =
[
$(\lambda d \in_\circ \mathfrak{K}(\!|HomDom|\!)(\!|Obj|\!).\ Yoneda\text{-}component\ \mathfrak{K}\ r\ u\ d)$,
$Hom_{O.C\alpha}\mathfrak{K}(\!|HomDom|\!)(r,-)$,
$\mathfrak{K}$,
$\mathfrak{K}(\!|HomDom|\!)$,
cat-Set α
]$_\circ$

Components.

lemma (**in** *is-functor*) *Yoneda-arrow-components*:
shows *Yoneda-arrow* α $\mathfrak{F}$ r $u(\!|NTMap|\!)$ =
$(\lambda d \in_\circ \mathfrak{A}(\!|Obj|\!).\ Yoneda\text{-}component\ \mathfrak{F}\ r\ u\ d)$
and *Yoneda-arrow* α $\mathfrak{F}$ r $u(\!|NTDom|\!) = Hom_{O.C\alpha}\mathfrak{A}(r,-)$
and *Yoneda-arrow* α $\mathfrak{F}$ r $u(\!|NTCod|\!) = \mathfrak{F}$
and *Yoneda-arrow* α $\mathfrak{F}$ r $u(\!|NTDGDom|\!) = \mathfrak{A}$
and *Yoneda-arrow* α $\mathfrak{F}$ r $u(\!|NTDGCod|\!) = cat\text{-}Set\ \alpha$
⟨*proof*⟩

29.3.2 Natural transformation map

mk-VLambda (**in** *is-functor*) *Yoneda-arrow-components*(*1*)
|*vsv Yoneda-arrow-NTMap-vsv*|
|*vdomain Yoneda-arrow-NTMap-vdomain*|
|*app Yoneda-arrow-NTMap-app*|

lemmas [*cat-cs-simps*] = *is-functor.Yoneda-arrow-NTMap-vdomain*

lemmas *Yoneda-arrow-NTMap-app*[*cat-cs-simps*] =
is-functor.Yoneda-arrow-NTMap-app

29.3.3 Yoneda arrow is a natural transformation

lemma (**in** *category*) *cat-Yoneda-arrow-is-ntcf*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $r \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
and $u \in_\circ \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor)$
shows *Yoneda-arrow* α $\mathfrak{K}$ r u : $Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF} \mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
⟨*proof*⟩

29.4 Yoneda Lemma

The following lemma is approximately equivalent to the Yoneda Lemma stated in subsection 1.15 in [3] (the first two conclusions correspond to the statement of the Yoneda lemma in Chapter III-2 in [7]).

lemma (**in** *category*) *cat-Yoneda-Lemma*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α **and** $r \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
shows *v11* (*Yoneda-map* α $\mathfrak{K}$ r)
and $\mathcal{R}_\circ$ (*Yoneda-map* α $\mathfrak{K}$ r) = $\mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor)$
and (*Yoneda-map* α $\mathfrak{K}$ r)$^{-1}{}_\circ$ = $(\lambda u \in_\circ \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor).$ *Yoneda-arrow* α $\mathfrak{K}$ r $u)$
⟨*proof*⟩

29.5 Inverse of the Yoneda map

lemma (**in** *category*) *inv-Yoneda-map-v11*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α **and** $r \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
shows *v11* ((*Yoneda-map* α $\mathfrak{K}$ r)$^{-1}{}_\circ$)
⟨*proof*⟩

lemma (**in** *category*) *inv-Yoneda-map-vdomain*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α **and** $r \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$
shows $\mathcal{D}_\circ$ ((*Yoneda-map* α $\mathfrak{K}$ r)$^{-1}{}_\circ$) = $\mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-Yoneda-map-vdomain*

lemma (**in** *category*) *inv-Yoneda-map-app*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α **and** $r \in_\circ \mathfrak{C}(\lfloor Obj \rfloor)$ **and** $u \in_\circ \mathfrak{K}(\lfloor ObjMap \rfloor)(\lfloor r \rfloor)$
shows (*Yoneda-map* α $\mathfrak{K}$ r)$^{-1}{}_\circ(\lfloor u \rfloor)$ = *Yoneda-arrow* α $\mathfrak{K}$ r u
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-Yoneda-map-app*

lemma (**in** *category*) *inv-Yoneda-map-vrange*:
assumes $\mathfrak{K} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows $\mathcal{R}_\circ$ ((*Yoneda-map* α $\mathfrak{K}$ r)$^{-1}{}_\circ$) =
these-ntcfs α $\mathfrak{C}$ (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(r,-)$ $\mathfrak{K}$
⟨*proof*⟩

29.6 Component of a composition of a *Hom*-natural transformation with natural transformations

29.6.1 Definition and elementary properties

The following definition is merely a technical generalization that is used in the context of the description of the composition of a *Hom*-natural transformation with a natural transformation later in this section (also see subsection 1.15 in [3]).

definition *ntcf-Hom-component* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-Hom-component* $\varphi\ \psi\ a\ b$ =
    [
      (
        $\lambda f \in_\circ Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|b|\!)).$
          $\psi(\!|NTMap|\!)(\!|b|\!) \circ_{A\psi(\!|NTDGCod|\!)} f \circ_{A\psi(\!|NTDGCod|\!)} \varphi(\!|NTMap|\!)(\!|a|\!)$
      ),
      $Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|b|\!)),$
      $Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|b|\!))$
    ]$_\circ$

Components.

lemma *ntcf-Hom-component-components*:
  shows *ntcf-Hom-component* $\varphi\ \psi\ a\ b(\!|ArrVal|\!)$ =
    (
      $\lambda f \in_\circ Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|b|\!)).$
        $\psi(\!|NTMap|\!)(\!|b|\!) \circ_{A\psi(\!|NTDGCod|\!)} f \circ_{A\psi(\!|NTDGCod|\!)} \varphi(\!|NTMap|\!)(\!|a|\!)$
)
    and *ntcf-Hom-component* $\varphi\ \psi\ a\ b(\!|ArrDom|\!)$ =
      $Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|b|\!))$
    and *ntcf-Hom-component* $\varphi\ \psi\ a\ b(\!|ArrCod|\!)$ =
      $Hom\ (\varphi(\!|NTDGCod|\!))\ (\varphi(\!|NTDom|\!)(\!|ObjMap|\!)(\!|a|\!))\ (\psi(\!|NTCod|\!)(\!|ObjMap|\!)(\!|b|\!))$
  ⟨*proof*⟩

29.6.2 Arrow value

mk-VLambda *ntcf-Hom-component-components*(*1*)
  |*vsv ntcf-Hom-component-ArrVal-vsv*[*intro*]|

context
  fixes $\alpha\ \varphi\ \psi\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{F}'\ \mathfrak{G}'\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}$
  assumes φ: $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and ψ: $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
begin

interpretation φ: *is-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{C}\ \mathfrak{F}\ \mathfrak{G}\ \varphi$ ⟨*proof*⟩
interpretation ψ: *is-ntcf* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{F}'\ \mathfrak{G}'\ \psi$ ⟨*proof*⟩

mk-VLambda
  ntcf-Hom-component-components(*1*)
    [
      of $\varphi\ \psi$,
      unfolded
        φ.ntcf-NTDom ψ.ntcf-NTDom
        φ.ntcf-NTCod ψ.ntcf-NTCod
        φ.ntcf-NTDGDom ψ.ntcf-NTDGDom
        φ.ntcf-NTDGCod ψ.ntcf-NTDGCod
    ]
  |*vdomain ntcf-Hom-component-ArrVal-vdomain*|

|*app ntcf-Hom-component-ArrVal-app*[*unfolded in-Hom-iff*]|

lemmas [*cat-cs-simps*] =
ntcf-Hom-component-ArrVal-vdomain
ntcf-Hom-component-ArrVal-app

lemma *ntcf-Hom-component-ArrVal-vrange*:
assumes $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows
$\mathcal{R}_{\circ}$ (*ntcf-Hom-component* φ ψ a $b(\!|ArrVal|\!)$) $\subseteq_{\circ}$
Hom $\mathfrak{C}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{G}'(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

end

29.6.3 Arrow domain and codomain

context
fixes α φ ψ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{F}'$ $\mathfrak{G}'$ $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes φ: $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and ψ: $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
begin

interpretation φ: *is-ntcf* α $\mathfrak{A}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{G}$ φ ⟨*proof*⟩
interpretation ψ: *is-ntcf* α $\mathfrak{B}$ $\mathfrak{C}$ $\mathfrak{F}'$ $\mathfrak{G}'$ ψ ⟨*proof*⟩

lemma *ntcf-Hom-component-ArrDom*[*cat-cs-simps*]:
ntcf-Hom-component φ ψ a $b(\!|ArrDom|\!)$ = *Hom* $\mathfrak{C}$ ($\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{F}'(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

lemma *ntcf-Hom-component-ArrCod*[*cat-cs-simps*]:
ntcf-Hom-component φ ψ a $b(\!|ArrCod|\!)$ = *Hom* $\mathfrak{C}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{G}'(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

end

29.6.4 Component of a composition of a *Hom*-natural transformation with natural transformations is an arrow in the category *Set*

lemma (**in** *category*) *cat-ntcf-Hom-component-is-arr*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ}$ *op-cat* $\mathfrak{A}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows
ntcf-Hom-component φ ψ a b :
Hom $\mathfrak{C}$ ($\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{F}'(\!|ObjMap|\!)(\!|b|\!)$) $\mapsto_{cat\text{-}Set\ \alpha}$
Hom $\mathfrak{C}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{G}'(\!|ObjMap|\!)(\!|b|\!)$)
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Hom-component-is-arr'*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ}$ *op-cat* $\mathfrak{A}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}'$ = *Hom* $\mathfrak{C}$ ($\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{F}'(\!|ObjMap|\!)(\!|b|\!)$)
and $\mathfrak{B}'$ = *Hom* $\mathfrak{C}$ ($\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$) ($\mathfrak{G}'(\!|ObjMap|\!)(\!|b|\!)$)
and $\mathfrak{C}'$ = *cat-Set* α

shows *ntcf-Hom-component* φ ψ a b : $\mathfrak{A}' \mapsto_{\mathfrak{C}'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-Hom-component-is-arr'*

29.6.5 Naturality of the components of a composition of a *Hom*-natural transformation with natural transformations

lemma (**in** *category*) *cat-ntcf-Hom-component-nat*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $g : a \mapsto_{op\text{-}cat} \mathfrak{A}\ a'$
and $f : b \mapsto_{\mathfrak{B}} b'$
shows
ntcf-Hom-component φ ψ a' b' $\circ_{A\,cat\text{-}Set\ \alpha}$
cf-hom $\mathfrak{C}$ [$\mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor)$, $\mathfrak{F}'(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$]$_\circ$ =
cf-hom $\mathfrak{C}$ [$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor)$, $\mathfrak{G}'(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$]$_\circ$ $\circ_{A\,cat\text{-}Set\ \alpha}$
ntcf-Hom-component φ ψ a b
⟨*proof*⟩

29.6.6 Composition of the components of a composition of a *Hom*-natural transformation with natural transformations

lemma (**in** *category*) *cat-ntcf-Hom-component-Comp*:
assumes $\varphi' : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi' : \mathfrak{G}' \mapsto_{CF} \mathfrak{H}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows
ntcf-Hom-component φ ψ' a b $\circ_{A\,cat\text{-}Set\ \alpha}$ *ntcf-Hom-component* φ' ψ a b =
ntcf-Hom-component ($\varphi' \cdot_{NTCF} \varphi$) ($\psi' \cdot_{NTCF} \psi$) a b
(**is** ‹*?φψ'* $\circ_{A\,cat\text{-}Set\ \alpha}$ *?φ'ψ* = *?φ'φψ'ψ*›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-component-Comp*

29.6.7 Component of a composition of *Hom*-natural transformation with the identity natural transformations

lemma (**in** *category*) *cat-ntcf-Hom-component-ntcf-id*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ \mathfrak{A}(\lfloor Obj \rfloor)$
and $b \in_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
shows
ntcf-Hom-component (*ntcf-id* $\mathfrak{F}$) (*ntcf-id* $\mathfrak{F}'$) a b =
cat-Set $\alpha(\lfloor CId \rfloor)(\lfloor Hom\ \mathfrak{C}\ (\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor))\ (\mathfrak{F}'(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)) \rfloor)$
(**is** ‹*?𝔉𝔉'* = *cat-Set* $\alpha(\lfloor CId \rfloor)(\lfloor ?\mathfrak{F}a\mathfrak{F}'b \rfloor)$›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-component-ntcf-id*

29.7 Component of a composition of a *Hom*-natural transformation with a natural transformation

29.7.1 Definition and elementary properties

definition *ntcf-lcomp-Hom-component* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-lcomp-Hom-component* $\varphi\ a\ b =$
    ntcf-Hom-component φ (*ntcf-id* (*cf-id* (φ(|*NTDGCod*|)))) $a\ b$

definition *ntcf-rcomp-Hom-component* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-rcomp-Hom-component* $\psi\ a\ b =$
    ntcf-Hom-component (*ntcf-id* (*cf-id* (ψ(|*NTDGCod*|)))) $\psi\ a\ b$

29.7.2 Arrow value

lemma *ntcf-lcomp-Hom-component-ArrVal-vsv*:
  vsv (*ntcf-lcomp-Hom-component* $\varphi\ a\ b$(|*ArrVal*|))
  ⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrVal-vsv*:
  vsv (*ntcf-rcomp-Hom-component* $\psi\ a\ b$(|*ArrVal*|))
  ⟨*proof*⟩

lemma *ntcf-lcomp-Hom-component-ArrVal-vdomain*[*cat-cs-simps*]:
  assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $b \in_\circ \mathfrak{C}$(|*Obj*|)
  shows $\mathcal{D}_\circ$ (*ntcf-lcomp-Hom-component* $\varphi\ a\ b$(|*ArrVal*|)) = *Hom* $\mathfrak{C}$ ($\mathfrak{G}$(|*ObjMap*|)(|a|)) b
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrVal-vdomain*[*cat-cs-simps*]:
  assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $a \in_\circ$ *op-cat* $\mathfrak{C}$(|*Obj*|)
  shows $\mathcal{D}_\circ$ (*ntcf-rcomp-Hom-component* $\psi\ a\ b$(|*ArrVal*|)) = *Hom* $\mathfrak{C}\ a$ ($\mathfrak{F}$(|*ObjMap*|)(|b|))
⟨*proof*⟩

lemma *ntcf-lcomp-Hom-component-ArrVal-app*[*cat-cs-simps*]:
  assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ$ *op-cat* $\mathfrak{A}$(|*Obj*|)
    and $b \in_\circ \mathfrak{C}$(|*Obj*|)
    and $h : \mathfrak{G}$(|*ObjMap*|)(|a|) $\mapsto_{\mathfrak{C}} b$
  shows *ntcf-lcomp-Hom-component* $\varphi\ a\ b$(|*ArrVal*|)(|h|) = $h \circ_{A\mathfrak{C}}$ φ(|*NTMap*|)(|a|)
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrVal-app*[*cat-cs-simps*]:
  assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ$ *op-cat* $\mathfrak{C}$(|*Obj*|)
    and $b \in_\circ \mathfrak{B}$(|*Obj*|)
    and $h : a \mapsto_{\mathfrak{C}} \mathfrak{F}$(|*ObjMap*|)(|$b$|)
  shows *ntcf-rcomp-Hom-component* $\psi\ a\ b$(|*ArrVal*|)(|h|) = ψ(|*NTMap*|)(|b|) $\circ_{A\mathfrak{C}}$ h
⟨*proof*⟩

lemma *ntcf-lcomp-Hom-component-ArrVal-vrange*:
  assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ$ *op-cat* $\mathfrak{A}$(|*Obj*|)
    and $b \in_\circ \mathfrak{C}$(|*Obj*|)
  shows $\mathcal{R}_\circ$ (*ntcf-lcomp-Hom-component* $\varphi\ a\ b$(|*ArrVal*|)) $\subseteq_\circ$ *Hom* $\mathfrak{C}$ ($\mathfrak{F}$(|*ObjMap*|)(|a|)) b
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrVal-vrange*:
  assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $a \in_\circ$ *op-cat* $\mathfrak{C}$(|*Obj*|)

and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $\mathcal{R}_{\circ}$ (*ntcf-rcomp-Hom-component* ψ a $b(\!|ArrVal|\!)$) $\subseteq_{\circ}$ *Hom* $\mathfrak{C}$ a $(\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

29.7.3 Arrow domain and codomain

lemma *ntcf-lcomp-Hom-component-ArrDom*[*cat-cs-simps*]:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-lcomp-Hom-component* φ a $b(\!|ArrDom|\!)$ = *Hom* $\mathfrak{C}$ $(\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!))$ b
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrDom*[*cat-cs-simps*]:
assumes $\psi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $a \in_{\circ}$ *op-cat* $\mathfrak{C}(\!|Obj|\!)$
shows *ntcf-rcomp-Hom-component* ψ a $b(\!|ArrDom|\!)$ = *Hom* $\mathfrak{C}$ a $(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

lemma *ntcf-lcomp-Hom-component-ArrCod*[*cat-cs-simps*]:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-lcomp-Hom-component* φ a $b(\!|ArrCod|\!)$ = *Hom* $\mathfrak{C}$ $(\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))$ b
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-component-ArrCod*[*cat-cs-simps*]:
assumes $\psi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $a \in_{\circ}$ *op-cat* $\mathfrak{C}(\!|Obj|\!)$
shows *ntcf-rcomp-Hom-component* ψ a $b(\!|ArrCod|\!)$ = *Hom* $\mathfrak{C}$ a $(\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

29.7.4 Component of a composition of a *Hom*-natural transformation with a natural transformation is an arrow in the category *Set*

lemma (**in** *category*) *cat-ntcf-lcomp-Hom-component-is-arr*:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ}$ *op-cat* $\mathfrak{A}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-lcomp-Hom-component* φ a b :
Hom $\mathfrak{C}$ $(\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!))$ $b \mapsto_{cat\text{-}Set\ \alpha}$ *Hom* $\mathfrak{C}$ $(\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))$ b
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-lcomp-Hom-component-is-arr′*:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ}$ *op-cat* $\mathfrak{A}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{A}'$ = *Hom* $\mathfrak{C}$ $(\mathfrak{G}(\!|ObjMap|\!)(\!|a|\!))$ b
and $\mathfrak{B}'$ = *Hom* $\mathfrak{C}$ $(\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))$ b
and $\mathfrak{C}'$ = *cat-Set* α
shows *ntcf-lcomp-Hom-component* φ a b : $\mathfrak{A}' \mapsto_{\mathfrak{C}'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-lcomp-Hom-component-is-arr′*

lemma (**in** *category*) *cat-ntcf-rcomp-Hom-component-is-arr*:
assumes $\psi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ}$ *op-cat* $\mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows *ntcf-rcomp-Hom-component* ψ a b :
Hom $\mathfrak{C}$ a $(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)) \mapsto_{cat\text{-}Set\ \alpha}$ *Hom* $\mathfrak{C}$ a $(\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-rcomp-Hom-component-is-arr′*:

assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_{\circ} op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
and $\mathfrak{A}' = Hom\ \mathfrak{C}\ a\ (\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
and $\mathfrak{B}' = Hom\ \mathfrak{C}\ a\ (\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!))$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $ntcf\text{-}rcomp\text{-}Hom\text{-}component\ \psi\ a\ b : \mathfrak{A}' \mapsto_{\mathfrak{C}'} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-rcomp-Hom-component-is-arr'*

29.8 Composition of a *Hom*-natural transformation with two natural transformations

29.8.1 Definition and elementary properties

See subsection 1.15 in [3].

definition *ntcf-Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹$Hom_{A.C^1}'(/--,--/')$›)
where $Hom_{A.C\alpha}(\varphi{-},\psi{-}) =$
[
(
$\lambda ab{\in_{\circ}}(op\text{-}cat\ (\varphi(\!|NTDGDom|\!))\ \times_C\ \psi(\!|NTDGDom|\!))(\!|Obj|\!).$
$ntcf\text{-}Hom\text{-}component\ \varphi\ \psi\ (vpfst\ ab)\ (vpsnd\ ab)$
),
$Hom_{O.C\alpha}\psi(\!|NTDGCod|\!)(\varphi(\!|NTCod|\!)-,\psi(\!|NTDom|\!)-)$,
$Hom_{O.C\alpha}\psi(\!|NTDGCod|\!)(\varphi(\!|NTDom|\!)-,\psi(\!|NTCod|\!)-)$,
$op\text{-}cat\ (\varphi(\!|NTDGDom|\!))\ \times_C\ \psi(\!|NTDGDom|\!)$,
$cat\text{-}Set\ \alpha$
]$_{\circ}$

Components.

lemma *ntcf-Hom-components*:
shows $Hom_{A.C\alpha}(\varphi{-},\psi{-})(\!|NTMap|\!) =$
(
$\lambda ab{\in_{\circ}}(op\text{-}cat\ (\varphi(\!|NTDGDom|\!))\ \times_C\ \psi(\!|NTDGDom|\!))(\!|Obj|\!).$
$ntcf\text{-}Hom\text{-}component\ \varphi\ \psi\ (vpfst\ ab)\ (vpsnd\ ab)$
)
and $Hom_{A.C\alpha}(\varphi{-},\psi{-})(\!|NTDom|\!) =$
$Hom_{O.C\alpha}\psi(\!|NTDGCod|\!)(\varphi(\!|NTCod|\!)-,\psi(\!|NTDom|\!)-)$
and $Hom_{A.C\alpha}(\varphi{-},\psi{-})(\!|NTCod|\!) =$
$Hom_{O.C\alpha}\psi(\!|NTDGCod|\!)(\varphi(\!|NTDom|\!)-,\psi(\!|NTCod|\!)-)$
and $Hom_{A.C\alpha}(\varphi{-},\psi{-})(\!|NTDGDom|\!) = op\text{-}cat\ (\varphi(\!|NTDGDom|\!))\ \times_C\ \psi(\!|NTDGDom|\!)$
and $Hom_{A.C\alpha}(\varphi{-},\psi{-})(\!|NTDGCod|\!) = cat\text{-}Set\ \alpha$
⟨*proof*⟩

29.8.2 Natural transformation map

mk-VLambda *ntcf-Hom-components*(*1*)
|*vsv ntcf-Hom-NTMap-vsv*|

context
fixes $\alpha\ \varphi\ \psi\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{F}'\ \mathfrak{G}'\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}$
assumes φ: $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and ψ: $\psi : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
begin

interpretation φ: *is-ntcf* $\alpha\ \mathfrak{A}\ \mathfrak{C}\ \mathfrak{F}\ \mathfrak{G}\ \varphi$ ⟨*proof*⟩
interpretation ψ: *is-ntcf* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{F}'\ \mathfrak{G}'\ \psi$ ⟨*proof*⟩

mk-VLambda *ntcf-Hom-components(1)*[*of - φ ψ, simplified*]
|*vdomain ntcf-Hom-NTMap-vdomain*[*unfolded in-Hom-iff*]|

lemmas [*cat-cs-simps*] = *ntcf-Hom-NTMap-vdomain*

lemma *ntcf-Hom-NTMap-app*[*cat-cs-simps*]:
assumes $[a,\ b]_{\circ} \in_{\circ} (\textit{op-cat}\ \mathfrak{A} \times_C \mathfrak{B})(\!|\textit{Obj}|\!)$
shows $\textit{Hom}_{A.C\alpha}(\varphi-,\psi-)(\!|\textit{NTMap}|\!)(\!|a,\ b|\!)_{\bullet} = \textit{ntcf-Hom-component}\ \varphi\ \psi\ a\ b$
⟨*proof*⟩

end

lemma (**in** *category*) *ntcf-Hom-NTMap-vrange*:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\psi : \mathfrak{F}' \mapsto\mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_{\circ}\ (\textit{Hom}_{A.C\alpha}(\varphi-,\psi-)(\!|\textit{NTMap}|\!)) \subseteq_{\circ} \textit{cat-Set}\ \alpha(\!|\textit{Arr}|\!)$
⟨*proof*⟩

29.8.3 Composition of a *Hom*-natural transformation with two natural transformations is a natural transformation

lemma (**in** *category*) *cat-ntcf-Hom-is-ntcf*:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\psi : \mathfrak{F}' \mapsto\mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\textit{Hom}_{A.C\alpha}(\varphi-,\psi-) :$
$\textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,\mathfrak{F}'-) \mapsto\mapsto_{CF} \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}'-) :$
$\textit{op-cat}\ \mathfrak{A} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Hom-is-ntcf'*:
assumes $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto\mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{A}' = \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,\mathfrak{F}'-)$
and $\mathfrak{B}' = \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{G}'-)$
and $\mathfrak{C}' = \textit{op-cat}\ \mathfrak{A} \times_C \mathfrak{B}$
and $\mathfrak{D}' = \textit{cat-Set}\ \alpha$
shows $\textit{Hom}_{A.C\alpha}(\varphi-,\psi-) : \mathfrak{A}' \mapsto\mapsto_{CF} \mathfrak{B}' : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-Hom-is-ntcf'*

29.8.4 Composition of a *Hom*-natural transformation with two vertical compositions of natural transformations

lemma (**in** *category*) *cat-ntcf-Hom-vcomp*:
assumes $\varphi' : \mathfrak{G} \mapsto\mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\varphi : \mathfrak{F} \mapsto\mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi' : \mathfrak{G}' \mapsto\mapsto_{CF} \mathfrak{H}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\psi : \mathfrak{F}' \mapsto\mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows
$\textit{Hom}_{A.C\alpha}(\varphi' \bullet_{NTCF} \varphi-,\psi' \bullet_{NTCF} \psi-) =$
$\textit{Hom}_{A.C\alpha}(\varphi-,\psi'-) \bullet_{NTCF} \textit{Hom}_{A.C\alpha}(\varphi'-,\psi-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-vcomp*

lemma (**in** *category*) *cat-ntcf-Hom-ntcf-id*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{F}'$: $\mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

shows $Hom_{A.C\alpha}(ntcf\text{-}id\ \mathfrak{F}-,ntcf\text{-}id\ \mathfrak{F}'-) = ntcf\text{-}id\ Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,\mathfrak{F}'-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-ntcf-id*

29.9 Composition of a *Hom*-natural transformation with a natural transformation

29.9.1 Definition and elementary properties

See subsection 1.15 in [3].

definition *ntcf-lcomp-Hom* :: $V \Rightarrow V \Rightarrow V$ (‹$Hom_{A.C}1'(/--,-/')$›)
where $Hom_{A.C\alpha}(\varphi-,-) = Hom_{A.C\alpha}(\varphi-,ntcf\text{-}id\ (cf\text{-}id\ (\varphi(\!|NTDGCod|\!)))-)$

definition *ntcf-rcomp-Hom* :: $V \Rightarrow V \Rightarrow V$ (‹$Hom_{A.C}1'(/-,--/')$›)
where $Hom_{A.C\alpha}(-,\psi-) = Hom_{A.C\alpha}(ntcf\text{-}id\ (cf\text{-}id\ (\psi(\!|NTDGCod|\!)))-,\psi-)$

29.9.2 Natural transformation map

lemma *ntcf-lcomp-Hom-NTMap-vsv*: *vsv* $(Hom_{A.C\alpha}(\varphi-,-)(\!|NTMap|\!))$
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-NTMap-vsv*: *vsv* $(Hom_{A.C\alpha}(-,\psi-)(\!|NTMap|\!))$
⟨*proof*⟩

lemma *ntcf-lcomp-Hom-NTMap-vdomain*[*cat-cs-simps*]:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{A.C\alpha}(\varphi-,-)(\!|NTMap|\!)) = (op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-NTMap-vdomain*[*cat-cs-simps*]:
assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ\ (Hom_{A.C\alpha}(-,\psi-)(\!|NTMap|\!)) = (op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

lemma *ntcf-lcomp-Hom-NTMap-app*[*cat-cs-simps*]:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ op\text{-}cat\ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $Hom_{A.C\alpha}(\varphi-,-)(\!|NTMap|\!)(\!|a, b|\!)_\bullet = ntcf\text{-}lcomp\text{-}Hom\text{-}component\ \varphi\ a\ b$
⟨*proof*⟩

lemma *ntcf-rcomp-Hom-NTMap-app*[*cat-cs-simps*]:
assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $a \in_\circ op\text{-}cat\ \mathfrak{C}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $Hom_{A.C\alpha}(-,\psi-)(\!|NTMap|\!)(\!|a, b|\!)_\bullet = ntcf\text{-}rcomp\text{-}Hom\text{-}component\ \psi\ a\ b$
⟨*proof*⟩

lemma (**in** *category*) *ntcf-lcomp-Hom-NTMap-vrange*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (Hom_{A.C\alpha}(\varphi-,-)(\!|NTMap|\!)) \subseteq_\circ cat\text{-}Set\ \alpha(\!|Arr|\!)$
⟨*proof*⟩

lemma (**in** *category*) *ntcf-rcomp-Hom-NTMap-vrange*:
assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ (Hom_{A.C\alpha}(-,\psi-)(\!|NTMap|\!)) \subseteq_\circ cat\text{-}Set\ \alpha(\!|Arr|\!)$
⟨*proof*⟩

29.9.3 Composition of a *Hom*-natural transformation with a natural transformation is a natural transformation

lemma (in *category*) *cat-ntcf-lcomp-Hom-is-ntcf*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(\varphi-,-) :$
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) : op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-lcomp-Hom-is-ntcf'*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{A}' = Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$
and $\mathfrak{B}' = Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$
and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{A} \times_C \mathfrak{C}$
and $\mathfrak{D}' = cat\text{-}Set\ \alpha$
shows $Hom_{A.C\alpha}(\varphi-,-) : \mathfrak{A}' \mapsto_{CF} \mathfrak{B}' : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-lcomp-Hom-is-ntcf'*

lemma (in *category*) *cat-ntcf-rcomp-Hom-is-ntcf*:
assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(-,\psi-) :$
$Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{F}-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-) : op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-rcomp-Hom-is-ntcf'*:
assumes $\psi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{A}' = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{F}-)$
and $\mathfrak{B}' = Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{G}-)$
and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C} \times_C \mathfrak{B}$
and $\mathfrak{D}' = cat\text{-}Set\ \alpha$
shows $Hom_{A.C\alpha}(-,\psi-) : \mathfrak{A}' \mapsto_{CF} \mathfrak{B}' : \mathfrak{C}' \mapsto\mapsto_{C\alpha} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-rcomp-Hom-is-ntcf'*

29.9.4 Component of a composition of a *Hom*-natural transformation with a natural transformation and the Yoneda component

lemma (in *category*) *cat-ntcf-lcomp-Hom-component-is-Yoneda-component*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $b \in_\circ op\text{-}cat\ \mathfrak{B}(|Obj|)$
and $c \in_\circ \mathfrak{C}(|Obj|)$
shows
$ntcf\text{-}lcomp\text{-}Hom\text{-}component\ \varphi\ b\ c =$
$Yoneda\text{-}component\ Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}(|ObjMap|)(|b|),-)\ (\mathfrak{G}(|ObjMap|)(|b|))\ (\varphi(|NTMap|)(|b|))\ c$
(**is** ‹*?lcomp* = *?Yc*›)
⟨*proof*⟩

29.9.5 Composition of a *Hom*-natural transformation with a vertical composition of natural transformations

lemma (in *category*) *cat-ntcf-lcomp-Hom-vcomp*:
assumes $\varphi' : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(\varphi' \bullet_{NTCF} \varphi-,-) = Hom_{A.C\alpha}(\varphi-,-) \bullet_{NTCF} Hom_{A.C\alpha}(\varphi'-,-)$

⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-lcomp-Hom-vcomp*

lemma (**in** *category*) *cat-ntcf-rcomp-Hom-vcomp*:
assumes $\varphi' : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(-,\varphi' \bullet_{NTCF} \varphi-) = Hom_{A.C\alpha}(-,\varphi'-) \bullet_{NTCF} Hom_{A.C\alpha}(-,\varphi-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-rcomp-Hom-vcomp*

29.9.6 Composition of a *Hom*-natural transformation with an identity natural transformation

lemma (**in** *category*) *cat-ntcf-lcomp-Hom-ntcf-id*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(ntcf\text{-}id\ \mathfrak{F}-,-) = ntcf\text{-}id\ Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-lcomp-Hom-ntcf-id*

lemma (**in** *category*) *cat-ntcf-rcomp-Hom-ntcf-id*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $Hom_{A.C\alpha}(-,ntcf\text{-}id\ \mathfrak{F}-) = ntcf\text{-}id\ Hom_{O.C\alpha}\mathfrak{C}(-,\mathfrak{F}-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-rcomp-Hom-ntcf-id*

29.10 Projections of a *Hom*-natural transformation

The concept of a projection of a *Hom*-natural transformation appears in the corollary to the Yoneda Lemma in Chapter III-2 in [7] (although the concept has not been given any specific name in the aforementioned reference).

29.10.1 Definition and elementary properties

definition *ntcf-Hom-snd* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ ($\langle Hom_{A.C^1}\text{-}'(/\text{-},\text{-}/')\rangle$)
where $Hom_{A.C\alpha}\mathfrak{C}(f,-) =$
Yoneda-arrow α ($Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{C}(\lfloor Dom\rfloor)(\lfloor f\rfloor),-)$) ($\mathfrak{C}(\lfloor Cod\rfloor)(\lfloor f\rfloor)$) f

definition *ntcf-Hom-fst* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ ($\langle Hom_{A.C^1}\text{-}'(/-,\text{-}/')\rangle$)
where $Hom_{A.C\alpha}\mathfrak{C}(-,f) = Hom_{A.C\alpha}$*op-cat* $\mathfrak{C}(f,-)$

Components.

lemma (**in** *category*) *cat-ntcf-Hom-snd-components*:
assumes $f : s \mapsto_{\mathfrak{C}} r$
shows $Hom_{A.C\alpha}\mathfrak{C}(f,-)(\lfloor NTMap\rfloor) =$
$(\lambda d \in_{\circ} \mathfrak{C}(\lfloor Obj\rfloor).\ Yoneda\text{-}component\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r\ f\ d)$
and $Hom_{A.C\alpha}\mathfrak{C}(f,-)(\lfloor NTDom\rfloor) = Hom_{O.C\alpha}\mathfrak{C}(r,-)$
and $Hom_{A.C\alpha}\mathfrak{C}(f,-)(\lfloor NTCod\rfloor) = Hom_{O.C\alpha}\mathfrak{C}(s,-)$
and $Hom_{A.C\alpha}\mathfrak{C}(f,-)(\lfloor NTDGDom\rfloor) = \mathfrak{C}$
and $Hom_{A.C\alpha}\mathfrak{C}(f,-)(\lfloor NTDGCod\rfloor) = cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Hom-fst-components*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
shows $Hom_{A.C\alpha}\mathfrak{C}(-,f)(\lfloor NTMap\rfloor) =$

$(\lambda d \in_\circ op\text{-}cat\ \mathfrak{C}(|Obj|).\ Yoneda\text{-}component\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r\ f\ d)$
and $Hom_{A.C\alpha}\mathfrak{C}(-,f)(|NTDom|) = Hom_{O.C\alpha}\mathfrak{C}(-,r)$
and $Hom_{A.C\alpha}\mathfrak{C}(-,f)(|NTCod|) = Hom_{O.C\alpha}\mathfrak{C}(-,s)$
and $Hom_{A.C\alpha}\mathfrak{C}(-,f)(|NTDGDom|) = op\text{-}cat\ \mathfrak{C}$
and $Hom_{A.C\alpha}\mathfrak{C}(-,f)(|NTDGCod|) = cat\text{-}Set\ \alpha$
⟨*proof*⟩

Alternative definition.

lemma (in *category*) *ntcf-Hom-snd-def′*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
shows $Hom_{A.C\alpha}\mathfrak{C}(f,-) = Yoneda\text{-}arrow\ \alpha\ (Hom_{O.C\alpha}\mathfrak{C}(r,-))\ s\ f$
⟨*proof*⟩

lemma (in *category*) *ntcf-Hom-fst-def′*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
shows $Hom_{A.C\alpha}\mathfrak{C}(-,f) = Yoneda\text{-}arrow\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r\ f$
⟨*proof*⟩

29.10.2 Natural transformation map

context *category*
begin

context
fixes $s\ r\ f$
assumes *f*: $f : s \mapsto_{\mathfrak{C}} r$
begin

mk-VLambda *cat-ntcf-Hom-snd-components*(*1*)[*OF f*]
|*vsv ntcf-Hom-snd-NTMap-vsv*[*intro*]|
|*vdomain ntcf-Hom-snd-NTMap-vdomain*|
|*app ntcf-Hom-snd-NTMap-app*|

end

context
fixes $s\ r\ f$
assumes *f*: $f : r \mapsto_{\mathfrak{C}} s$
begin

mk-VLambda *cat-ntcf-Hom-fst-components*(*1*)[*OF f*]
|*vsv ntcf-Hom-fst-NTMap-vsv*[*intro*]|
|*vdomain ntcf-Hom-fst-NTMap-vdomain*|
|*app ntcf-Hom-fst-NTMap-app*|

end

end

lemmas [*cat-cs-simps*] =
category.ntcf-Hom-snd-NTMap-vdomain
category.ntcf-Hom-fst-NTMap-vdomain

lemmas *ntcf-Hom-snd-NTMap-app*[*cat-cs-simps*] =
category.ntcf-Hom-snd-NTMap-app
category.ntcf-Hom-fst-NTMap-app

29.10.3 *Hom*-natural transformation projections are natural transformations

lemma (in *category*) *cat-ntcf-Hom-snd-is-ntcf*:
assumes $f : s \mapsto_{\mathfrak{C}} r$
shows $Hom_{A.C\alpha}\mathfrak{C}(f,-) :$
$Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-snd-is-ntcf′*:
assumes $f : s \mapsto_{\mathfrak{C}} r$
and $\beta = \alpha$
and $\mathfrak{A}' = Hom_{O.C\alpha}\mathfrak{C}(r,-)$
and $\mathfrak{B}' = Hom_{O.C\alpha}\mathfrak{C}(s,-)$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = cat\text{-}Set\ \alpha$
shows $Hom_{A.C\alpha}\mathfrak{C}(f,-) : \mathfrak{A}' \mapsto_{CF} \mathfrak{B}' : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-Hom-snd-is-ntcf′*

lemma (in *category*) *cat-ntcf-Hom-fst-is-ntcf*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
shows $Hom_{A.C\alpha}\mathfrak{C}(-,f) :$
$Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-fst-is-ntcf′*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
and $\beta = \alpha$
and $\mathfrak{A}' = Hom_{O.C\alpha}\mathfrak{C}(-,r)$
and $\mathfrak{B}' = Hom_{O.C\alpha}\mathfrak{C}(-,s)$
and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{C}$
and $\mathfrak{D}' = cat\text{-}Set\ \alpha$
shows $Hom_{A.C\alpha}\mathfrak{C}(-,f) : \mathfrak{A}' \mapsto_{CF} \mathfrak{B}' : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-ntcf-Hom-fst-is-ntcf′*

29.10.4 Opposite *Hom*-natural transformation projections

lemma (in *category*) *cat-op-cat-ntcf-Hom-snd*:
$Hom_{A.C\alpha}op\text{-}cat\ \mathfrak{C}(f,-) = Hom_{A.C\alpha}\mathfrak{C}(-,f)$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.cat-op-cat-ntcf-Hom-snd*

lemma (in *category*) *cat-op-cat-ntcf-Hom-fst*:
$Hom_{A.C\alpha}op\text{-}cat\ \mathfrak{C}(-,f) = Hom_{A.C\alpha}\mathfrak{C}(f,-)$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *category.cat-op-cat-ntcf-Hom-fst*

29.10.5 *Hom*-natural transformation projections and the Yoneda component

lemma (in *category*) *cat-Yoneda-component-cf-Hom-snd-Comp*:
assumes $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$ **and** $d \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows
$Yoneda\text{-}component\ Hom_{O.C\alpha}\mathfrak{C}(a,-)\ b\ f\ d \circ_{A} cat\text{-}Set\ \alpha$
$Yoneda\text{-}component\ Hom_{O.C\alpha}\mathfrak{C}(b,-)\ c\ g\ d =$

Yoneda-component Hom$_{O.C\alpha}\mathfrak{C}(a,-)$ c $(g \circ_{A\mathfrak{C}} f)$ d
(**is** ‹*?Ya b f d* $\circ_{A\,cat\text{-}Set\ \alpha}$ *?Yb c g d* = *?Ya c* $(g \circ_{A\mathfrak{C}} f)$ *d*›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] =
category.cat-Yoneda-component-cf-Hom-snd-Comp[*symmetric*]

lemma (**in** *category*) *cat-Yoneda-component-cf-Hom-snd-CId*:
assumes $c \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $d \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows
Yoneda-component Hom$_{O.C\alpha}\mathfrak{C}(c,-)$ c $(\mathfrak{C}(\!|CId|\!)(\!|c|\!))$ d =
cat-Set $\alpha(\!|CId|\!)(\!|Hom\ \mathfrak{C}\ c\ d|\!)$
(**is** ‹*?Ycd* = *cat-Set* $\alpha(\!|CId|\!)(\!|Hom\ \mathfrak{C}\ c\ d|\!)$›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-Yoneda-component-cf-Hom-snd-CId*

29.10.6 *Hom*-natural transformation projection of a composition

lemma (**in** *category*) *cat-ntcf-Hom-snd-Comp*:
assumes $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows *Hom*$_{A.C\alpha}\mathfrak{C}(g \circ_{A\mathfrak{C}} f,-)$ = *Hom*$_{A.C\alpha}\mathfrak{C}(f,-)$ $\cdot_{NTCF}$ *Hom*$_{A.C\alpha}\mathfrak{C}(g,-)$
(**is** ‹*?H-gf* = *?H-f* $\cdot_{NTCF}$ *?H-g*›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-snd-Comp*

lemma (**in** *category*) *cat-ntcf-Hom-fst-Comp*:
assumes $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows *Hom*$_{A.C\alpha}\mathfrak{C}(-,g \circ_{A\mathfrak{C}} f)$ = *Hom*$_{A.C\alpha}\mathfrak{C}(-,g)$ $\cdot_{NTCF}$ *Hom*$_{A.C\alpha}\mathfrak{C}(-,f)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-fst-Comp*

29.10.7 *Hom*-natural transformation projection of an identity

lemma (**in** *category*) *cat-ntcf-Hom-snd-CId*:
assumes $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *Hom*$_{A.C\alpha}\mathfrak{C}(\mathfrak{C}(\!|CId|\!)(\!|c|\!),-)$ = *ntcf-id Hom*$_{O.C\alpha}\mathfrak{C}(c,-)$
(**is** ‹*?H-c* = *?id-H-c*›)
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-snd-CId*

lemma (**in** *category*) *cat-ntcf-Hom-fst-CId*:
assumes $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *Hom*$_{A.C\alpha}\mathfrak{C}(-,\mathfrak{C}(\!|CId|\!)(\!|c|\!))$ = *ntcf-id Hom*$_{O.C\alpha}\mathfrak{C}(-,c)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-Hom-fst-CId*

29.10.8 *Hom*-natural transformation and the Yoneda map

lemma (**in** *category*) *cat-Yoneda-map-of-ntcf-Hom-snd*:
assumes $f : s \mapsto_{\mathfrak{C}} r$
shows *Yoneda-map* α (*Hom*$_{O.C\alpha}\mathfrak{C}(s,-)$) $r(\!|Hom_{A.C\alpha}\mathfrak{C}(f,-)|\!)$ = f
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-Yoneda-map-of-ntcf-Hom-snd*

lemma (in *category*) *cat-Yoneda-map-of-ntcf-Hom-fst*:
assumes $f : r \mapsto_{\mathfrak{C}} s$
shows *Yoneda-map* α $(Hom_{O.C\alpha}\mathfrak{C}(-,s))$ $r(\!|Hom_{A.C\alpha}\mathfrak{C}(-,f)|\!) = f$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-Yoneda-map-of-ntcf-Hom-fst*

29.11 Evaluation arrow

29.11.1 Definition and elementary properties

The evaluation arrow is a part of the definition of the evaluation functor. The evaluation functor appears in Chapter III-2 in [7].

definition *cf-eval-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-eval-arrow* $\mathfrak{C}$ $\mathfrak{N}$ f =
[
(
$\lambda x \in_{\circ} \mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!).$
$\mathfrak{N}(\!|NTCod|\!)(\!|ArrMap|\!)(\!|f|\!)(\!|ArrVal|\!)(\!|\mathfrak{N}(\!|NTMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!)(\!|ArrVal|\!)(\!|x|\!)|\!)$
),
$\mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!)$,
$\mathfrak{N}(\!|NTCod|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Cod|\!)(\!|f|\!)|\!)$
]$_{\circ}$

Components.

lemma *cf-eval-arrow-components*:
shows *cf-eval-arrow* $\mathfrak{C}$ $\mathfrak{N}$ $f(\!|ArrVal|\!)$ =
(
$\lambda x \in_{\circ} \mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!).$
$\mathfrak{N}(\!|NTCod|\!)(\!|ArrMap|\!)(\!|f|\!)(\!|ArrVal|\!)(\!|\mathfrak{N}(\!|NTMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!)(\!|ArrVal|\!)(\!|x|\!)|\!)$
)
and *cf-eval-arrow* $\mathfrak{C}$ $\mathfrak{N}$ $f(\!|ArrDom|\!) = \mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Dom|\!)(\!|f|\!)|\!)$
and *cf-eval-arrow* $\mathfrak{C}$ $\mathfrak{N}$ $f(\!|ArrCod|\!) = \mathfrak{N}(\!|NTCod|\!)(\!|ObjMap|\!)(\!|\mathfrak{C}(\!|Cod|\!)(\!|f|\!)|\!)$
⟨*proof*⟩

context
fixes α $\mathfrak{N}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{G}$ a b f
assumes $\mathfrak{N}$: $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and f: $f : a \mapsto_{\mathfrak{C}} b$
begin

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

lemmas *cf-eval-arrow-components'* = *cf-eval-arrow-components*[
where $\mathfrak{C}$=$\mathfrak{C}$ **and** $\mathfrak{N}$=‹*ntcf-arrow* $\mathfrak{N}$› **and** f=f,
unfolded
ntcf-arrow-components
cf-map-components
$\mathfrak{N}$.*NTDom.HomDom.cat-is-arrD*[*OF f*]
cat-cs-simps
]

lemmas [*cat-cs-simps*] = *cf-eval-arrow-components'*(*2*,*3*)

end

29.11.2 Arrow value

context
  fixes α $\mathfrak{N}$ $\mathfrak{C}$ $\mathfrak{F}$ $\mathfrak{G}$ a b f
  assumes $\mathfrak{N}$: $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
    and f: $f : a \mapsto_{\mathfrak{C}} b$
begin

mk-VLambda *cf-eval-arrow-components'(1)*[*OF* $\mathfrak{N}$ f]
  |*vsv cf-eval-arrow-ArrVal-vsv*[*cat-cs-intros*]|
  |*vdomain cf-eval-arrow-ArrVal-vdomain*[*cat-cs-simps*]|
  |*app cf-eval-arrow-ArrVal-app*[*cat-cs-simps*]|

end

29.11.3 Evaluation arrow is an arrow in the category *Set*

lemma *cf-eval-arrow-is-arr*:
  assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$ **and** $f : a \mapsto_{\mathfrak{C}} b$
  shows *cf-eval-arrow* $\mathfrak{C}$ (*ntcf-arrow* $\mathfrak{N}$) f :
    $\mathfrak{F}(\!|\textit{ObjMap}|\!)(\!|a|\!) \mapsto_{\textit{cat-Set}\ \alpha} \mathfrak{G}(\!|\textit{ObjMap}|\!)(\!|b|\!)$
⟨*proof*⟩

lemma *cf-eval-arrow-is-arr'*[*cat-cs-intros*]:
  assumes $\mathfrak{N}' =$ *ntcf-arrow* $\mathfrak{N}$
    and $\mathfrak{F}a = \mathfrak{F}(\!|\textit{ObjMap}|\!)(\!|a|\!)$
    and $\mathfrak{G}b = \mathfrak{G}(\!|\textit{ObjMap}|\!)(\!|b|\!)$
    and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
    and $f : a \mapsto_{\mathfrak{C}} b$
  shows *cf-eval-arrow* $\mathfrak{C}$ $\mathfrak{N}'$ $f : \mathfrak{F}a \mapsto_{\textit{cat-Set}\ \alpha} \mathfrak{G}b$
  ⟨*proof*⟩

lemma (**in** *category*) *cat-cf-eval-arrow-ntcf-vcomp*[*cat-cs-simps*]:
  assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
    and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
    and $g : b \mapsto_{\mathfrak{C}} c$
    and $f : a \mapsto_{\mathfrak{C}} b$
  shows
    cf-eval-arrow $\mathfrak{C}$ (*ntcf-arrow* ($\mathfrak{M} \bullet_{NTCF} \mathfrak{N}$)) ($g \circ_{A\mathfrak{C}} f$) =
      cf-eval-arrow $\mathfrak{C}$ (*ntcf-arrow* $\mathfrak{M}$) g $\circ_{A}$ *cat-Set* α
      cf-eval-arrow $\mathfrak{C}$ (*ntcf-arrow* $\mathfrak{N}$) f
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-eval-arrow-ntcf-vcomp*

lemma (**in** *category*) *cat-cf-eval-arrow-ntcf-id*[*cat-cs-simps*]:
  assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$ **and** $c \in_{\circ} \mathfrak{C}(\!|\textit{Obj}|\!)$
  shows
    cf-eval-arrow $\mathfrak{C}$ (*ntcf-arrow* (*ntcf-id* $\mathfrak{F}$)) ($\mathfrak{C}(\!|\textit{CId}|\!)(\!|c|\!)$) =
      cat-Set $\alpha(\!|\textit{CId}|\!)(\!|\mathfrak{F}(\!|\textit{ObjMap}|\!)(\!|c|\!)|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-cf-eval-arrow-ntcf-id*

29.12 *HOM*-functor

29.12.1 Definition and elementary properties

The following definition is a technical generalization that is used later in this section.

definition *cf-HOM-snd* :: $V \Rightarrow V \Rightarrow V$ ($\langle HOM_{C1}'(/,--/')\rangle$)
where $HOM_{C\alpha}(,\mathfrak{F}-) =$
$[$
$(\lambda a \in_\circ op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(a,-) \circ_{CF} \mathfrak{F}))$,
(
$\lambda f \in_\circ op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))(\!|Arr|\!).$
$ntcf\text{-}arrow\ (Hom_{A.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(f,-) \circ_{NTCF\text{-}CF} \mathfrak{F})$
),
$op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))$,
$cat\text{-}FUNCT\ \alpha\ (\mathfrak{F}(\!|HomDom|\!))\ (cat\text{-}Set\ \alpha)$
$]_\circ$

definition *cf-HOM-fst* :: $V \Rightarrow V \Rightarrow V$ ($\langle HOM_{C1}'(/-,/')\rangle$)
where $HOM_{C\alpha}(\mathfrak{F}-,) =$
$[$
$(\lambda a \in_\circ (\mathfrak{F}(\!|HomCod|\!))(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(-,a) \circ_{CF} op\text{-}cf\ \mathfrak{F}))$,
(
$\lambda f \in_\circ (\mathfrak{F}(\!|HomCod|\!))(\!|Arr|\!).$
$ntcf\text{-}arrow\ (Hom_{A.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(-,f) \circ_{NTCF\text{-}CF} op\text{-}cf\ \mathfrak{F})$
),
$\mathfrak{F}(\!|HomCod|\!)$,
$cat\text{-}FUNCT\ \alpha\ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!)))\ (cat\text{-}Set\ \alpha)$
$]_\circ$

Components.

lemma *cf-HOM-snd-components*:
shows $HOM_{C\alpha}(,\mathfrak{F}-)(\!|ObjMap|\!) =$
$(\lambda a \in_\circ op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(a,-) \circ_{CF} \mathfrak{F}))$
and $HOM_{C\alpha}(,\mathfrak{F}-)(\!|ArrMap|\!) =$
(
$\lambda f \in_\circ op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))(\!|Arr|\!).$
$ntcf\text{-}arrow\ (Hom_{A.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(f,-) \circ_{NTCF\text{-}CF} \mathfrak{F})$
)
and [*cat-cs-simps*]: $HOM_{C\alpha}(,\mathfrak{F}-)(\!|HomDom|\!) = op\text{-}cat\ (\mathfrak{F}(\!|HomCod|\!))$
and [*cat-cs-simps*]:
$HOM_{C\alpha}(,\mathfrak{F}-)(\!|HomCod|\!) = cat\text{-}FUNCT\ \alpha\ (\mathfrak{F}(\!|HomDom|\!))\ (cat\text{-}Set\ \alpha)$
⟨*proof*⟩

lemma *cf-HOM-fst-components*:
shows $HOM_{C\alpha}(\mathfrak{F}-,)(\!|ObjMap|\!) =$
$(\lambda a \in_\circ (\mathfrak{F}(\!|HomCod|\!))(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(-,a) \circ_{CF} op\text{-}cf\ \mathfrak{F}))$
and $HOM_{C\alpha}(\mathfrak{F}-,)(\!|ArrMap|\!) =$
(
$\lambda f \in_\circ (\mathfrak{F}(\!|HomCod|\!))(\!|Arr|\!).$
$ntcf\text{-}arrow\ (Hom_{A.C\alpha}(\mathfrak{F}(\!|HomCod|\!))(-,f) \circ_{NTCF\text{-}CF} op\text{-}cf\ \mathfrak{F})$
)
and $HOM_{C\alpha}(\mathfrak{F}-,)(\!|HomDom|\!) = \mathfrak{F}(\!|HomCod|\!)$
and $HOM_{C\alpha}(\mathfrak{F}-,)(\!|HomCod|\!) = cat\text{-}FUNCT\ \alpha\ (op\text{-}cat\ (\mathfrak{F}(\!|HomDom|\!)))\ (cat\text{-}Set\ \alpha)$
⟨*proof*⟩

context *is-functor*
begin

lemmas *cf-HOM-snd-components'* =
cf-HOM-snd-components[**where** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cf-HomDom cf-HomCod*]

lemmas [*cat-cs-simps*] = *cf-HOM-snd-components'*(*3*,*4*)

lemmas *cf-HOM-fst-components′* =
cf-HOM-fst-components[**where** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded cf-HomDom cf-HomCod*]

lemmas [*cat-cs-simps*] = *cf-HOM-snd-components′*(*3*,*4*)

end

29.12.2 Object map

mk-VLambda *cf-HOM-snd-components*(*1*)
|*vsv cf-HOM-snd-ObjMap-vsv*[*cat-cs-intros*]|

mk-VLambda (**in** *is-functor*) *cf-HOM-snd-components′*(*1*)[*unfolded cat-op-simps*]
|*vdomain cf-HOM-snd-ObjMap-vdomain*[*cat-cs-simps*]|
|*app cf-HOM-snd-ObjMap-app*[*cat-cs-simps*]|

mk-VLambda *cf-HOM-snd-components*(*1*)
|*vsv cf-HOM-fst-ObjMap-vsv*[*cat-cs-intros*]|

mk-VLambda (**in** *is-functor*) *cf-HOM-fst-components′*(*1*)[*unfolded cat-op-simps*]
|*vdomain cf-HOM-fst-ObjMap-vdomain*[*cat-cs-simps*]|
|*app cf-HOM-fst-ObjMap-app*[*cat-cs-simps*]|

29.12.3 Arrow map

mk-VLambda *cf-HOM-snd-components*(*2*)
|*vsv cf-HOM-snd-ArrMap-vsv*[*cat-cs-intros*]|

mk-VLambda (**in** *is-functor*) *cf-HOM-snd-components′*(*2*)[*unfolded cat-op-simps*]
|*vdomain cf-HOM-snd-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-HOM-snd-ArrMap-app*[*cat-cs-simps*]|

mk-VLambda *cf-HOM-fst-components*(*2*)
|*vsv cf-HOM-fst-ArrMap-vsv*[*cat-cs-intros*]|

mk-VLambda (**in** *is-functor*) *cf-HOM-fst-components′*(*2*)[*unfolded cat-op-simps*]
|*vdomain cf-HOM-fst-ArrMap-vdomain*[*cat-cs-simps*]|
|*app cf-HOM-fst-ArrMap-app*[*cat-cs-simps*]|

29.12.4 Opposite *HOM*-functor

lemma (**in** *is-functor*) *cf-HOM-snd-op*[*cat-op-simps*]:
$HOM_{C\alpha}(,op\text{-}cf\ \mathfrak{F}-) = HOM_{C\alpha}(\mathfrak{F}-,)$
⟨*proof*⟩

lemmas [*cat-op-simps*] = *is-functor.cf-HOM-snd-op*

context *is-functor*
begin

lemmas *cf-HOM-fst-op*[*cat-op-simps*] =
is-functor.cf-HOM-snd-op[*OF is-functor-op*, *unfolded cat-op-simps*, *symmetric*]

end

lemmas [*cat-op-simps*] = *is-functor.cf-HOM-fst-op*

29.12.5 *HOM*-functor is a functor

lemma (in *is-functor*) *cf-HOM-snd-is-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows $HOM_{C\alpha}(,\mathfrak{F}-) : op\text{-}cat\ \mathfrak{B} \mapsto\mapsto_{C\beta} cat\text{-}FUNCT\ \alpha\ \mathfrak{A}\ (cat\text{-}Set\ \alpha)$
⟨*proof*⟩

lemma (in *is-functor*) *cf-HOM-snd-is-functor'*[*cat-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{C}' = op\text{-}cat\ \mathfrak{B}$
and $\mathfrak{D} = cat\text{-}FUNCT\ \alpha\ \mathfrak{A}\ (cat\text{-}Set\ \alpha)$
shows $HOM_{C\alpha}(,\mathfrak{F}-) : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-HOM-snd-is-functor'*

lemma (in *is-functor*) *cf-HOM-fst-is-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows $HOM_{C\alpha}(\mathfrak{F}-,) : \mathfrak{B} \mapsto\mapsto_{C\beta} cat\text{-}FUNCT\ \alpha\ (op\text{-}cat\ \mathfrak{A})\ (cat\text{-}Set\ \alpha)$
⟨*proof*⟩

lemma (in *is-functor*) *cf-HOM-fst-is-functor'*[*cat-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{C}' = \mathfrak{B}$
and $\mathfrak{D} = cat\text{-}FUNCT\ \alpha\ (op\text{-}cat\ \mathfrak{A})\ (cat\text{-}Set\ \alpha)$
shows $HOM_{C\alpha}(\mathfrak{F}-,) : \mathfrak{C}' \mapsto\mapsto_{C\beta} \mathfrak{D}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-HOM-fst-is-functor'*

29.13 Evaluation functor

29.13.1 Definition and elementary properties

See Chapter III-2 in [7].

definition *cf-eval* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *cf-eval* $\alpha\ \beta\ \mathfrak{C}$ =
[
$(\lambda\mathfrak{F} d\in_\circ (cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ (cat\text{-}Set\ \alpha) \times_C \mathfrak{C})(\!|Obj|\!).\ \mathfrak{F}d(\!|0|\!)(\!|ObjMap|\!)(\!|\mathfrak{F}d(\!|1_{\mathbb{N}}|\!)|\!))$,
(
$\lambda\mathfrak{N} f\in_\circ (cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ (cat\text{-}Set\ \alpha) \times_C \mathfrak{C})(\!|Arr|\!).$
$cf\text{-}eval\text{-}arrow\ \mathfrak{C}\ (\mathfrak{N}f(\!|0|\!))\ (\mathfrak{N}f(\!|1_{\mathbb{N}}|\!))$
),
$cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ (cat\text{-}Set\ \alpha) \times_C \mathfrak{C}$,
$cat\text{-}Set\ \beta$
]$_\circ$

Components.

lemma *cf-eval-components*:
shows $cf\text{-}eval\ \alpha\ \beta\ \mathfrak{C}(\!|ObjMap|\!) =$
$(\lambda\mathfrak{F} d\in_\circ (cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ (cat\text{-}Set\ \alpha) \times_C \mathfrak{C})(\!|Obj|\!).\ \mathfrak{F}d(\!|0|\!)(\!|ObjMap|\!)(\!|\mathfrak{F}d(\!|1_{\mathbb{N}}|\!)|\!))$
and $cf\text{-}eval\ \alpha\ \beta\ \mathfrak{C}(\!|ArrMap|\!) =$
(
$\lambda\mathfrak{N} f\in_\circ (cat\text{-}FUNCT\ \alpha\ \mathfrak{C}\ (cat\text{-}Set\ \alpha) \times_C \mathfrak{C})(\!|Arr|\!).$
$cf\text{-}eval\text{-}arrow\ \mathfrak{C}\ (\mathfrak{N}f(\!|0|\!))\ (\mathfrak{N}f(\!|1_{\mathbb{N}}|\!))$
)

and [*cat-cs-simps*]:
cf-eval α β $\mathfrak{C}(\!|HomDom|\!)$ = *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$
and [*cat-cs-simps*]: *cf-eval* α β $\mathfrak{C}(\!|HomCod|\!)$ = *cat-Set* β
⟨*proof*⟩

29.13.2 Object map

lemma *cf-eval-ObjMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-eval* α β $\mathfrak{C}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemma *cf-eval-ObjMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-eval* α β $\mathfrak{C}(\!|ObjMap|\!)$) = (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$)$(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *category*) *cf-eval-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{F}c$ = [*cf-map* $\mathfrak{F}$, c]$_\circ$
and $\mathfrak{F}$: $\mathfrak{C}$ $\mapsto\!\!\mapsto_{C\alpha}$ *cat-Set* α
and c $\in_\circ$ $\mathfrak{C}(\!|Obj|\!)$
shows *cf-eval* α β $\mathfrak{C}(\!|ObjMap|\!)(\!|\mathfrak{F}c|\!)$ = $\mathfrak{F}(\!|ObjMap|\!)(\!|c|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cf-eval-ObjMap-app*

29.13.3 Arrow map

lemma *cf-eval-ArrMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-eval* α β $\mathfrak{C}(\!|ArrMap|\!)$)
⟨*proof*⟩

lemma *cf-eval-ArrMap-vdomain*[*cat-cs-simps*]:
$\mathcal{D}_\circ$ (*cf-eval* α β $\mathfrak{C}(\!|ArrMap|\!)$) = (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$)$(\!|Arr|\!)$
⟨*proof*⟩

lemma (**in** *category*) *cf-eval-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathfrak{N}f$ = [*ntcf-arrow* $\mathfrak{N}$, f]$_\circ$
and $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{CF}$ $\mathfrak{G}$: $\mathfrak{C}$ $\mapsto\!\!\mapsto_{C\alpha}$ *cat-Set* α
and f : a $\mapsto_{\mathfrak{C}}$ b
shows *cf-eval* α β $\mathfrak{C}(\!|ArrMap|\!)(\!|\mathfrak{N}f|\!)$ = *cf-eval-arrow* $\mathfrak{C}$ (*ntcf-arrow* $\mathfrak{N}$) f
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cf-eval-ArrMap-app*

29.13.4 Evaluation functor is a functor

lemma (**in** *category*) *cat-cf-eval-is-functor*:
assumes $\mathcal{Z}$ β **and** α $\in_\circ$ β
shows *cf-eval* α β $\mathfrak{C}$: *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$ $\mapsto\!\!\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-eval-is-functor′*:
assumes $\mathcal{Z}$ β
and α $\in_\circ$ β
and $\mathfrak{A}'$ = *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$
and $\mathfrak{B}'$ = *cat-Set* β
and β' = β
shows *cf-eval* α β $\mathfrak{C}$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{C\beta'}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-cf-eval-is-functor′*

29.14 N-functor

29.14.1 Definition and elementary properties

See Chapter III-2 in [7].

definition *cf-nt* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *cf-nt* α β $\mathfrak{F}$ =
    bifunctor-flip ($\mathfrak{F}(\!|HomCod|\!)$) (*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α))
      ($Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))

Alternative definition.

lemma (**in** *is-functor*) *cf-nt-def'*:
  cf-nt α β $\mathfrak{F}$ =
    bifunctor-flip $\mathfrak{B}$ (*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α))
      ($Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
  ⟨*proof*⟩

Components.

lemma *cf-nt-components*:
  shows *cf-nt* α β $\mathfrak{F}(\!|ObjMap|\!)$ =
    (
      bifunctor-flip ($\mathfrak{F}(\!|HomCod|\!)$) (*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α))
        ($Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|ObjMap|\!$)
    and *cf-nt* α β $\mathfrak{F}(\!|ArrMap|\!)$ =
      (
        bifunctor-flip ($\mathfrak{F}(\!|HomCod|\!)$) (*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α))
          ($Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|ArrMap|\!$)
    and *cf-nt* α β $\mathfrak{F}(\!|HomDom|\!)$ =
      (
        bifunctor-flip ($\mathfrak{F}(\!|HomCod|\!)$) (*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α))
          ($Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|HomDom|\!$)
    and *cf-nt* α β $\mathfrak{F}(\!|HomCod|\!)$ =
      (
        bifunctor-flip ($\mathfrak{F}(\!|HomCod|\!)$) (*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α))
          ($Hom_{O.C\beta}$*cat-FUNCT* α ($\mathfrak{F}(\!|HomDom|\!)$) (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|HomCod|\!$)
  ⟨*proof*⟩

lemma (**in** *is-functor*) *cf-nt-components'*:
  assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
  shows *cf-nt* α β $\mathfrak{F}(\!|ObjMap|\!)$ =
      (
        bifunctor-flip $\mathfrak{B}$ (*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α))
          ($Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|ObjMap|\!$)
    and *cf-nt* α β $\mathfrak{F}(\!|ArrMap|\!)$ =
      (
        bifunctor-flip $\mathfrak{B}$ (*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α))
          ($Hom_{O.C\beta}$*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α)($HOM_{C\alpha}(,\mathfrak{F}-)-,-$))
)($\!|ArrMap|\!$)
    and [*cat-cs-simps*]:
      cf-nt α β $\mathfrak{F}(\!|HomDom|\!)$ = *cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α) $\times_C$ $\mathfrak{B}$
    and [*cat-cs-simps*]:
      cf-nt α β $\mathfrak{F}(\!|HomCod|\!)$ = *cat-Set* β

⟨*proof*⟩

lemmas [*cat-cs-simps*] = *is-functor.cf-nt-components'*(*3*,*4*)

29.14.2 Object map

lemma *cf-nt-ObjMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-nt* α β $\mathfrak{C}$(|*ObjMap*|))
⟨*proof*⟩

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

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

lemma (**in** *is-functor*) *cf-nt-ObjMap-app*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{G}b$ = [*cf-map* $\mathfrak{G}$, b]$_\circ$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $b \in_\circ \mathfrak{B}$(|*Obj*|)
shows *cf-nt* α β $\mathfrak{F}$(|*ObjMap*|)(|$\mathfrak{G}b$|) = *Hom*
(*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α))
(*cf-map* (*Hom*$_{O.C\alpha}\mathfrak{B}(b,-) \circ_{CF} \mathfrak{F}$))
(*cf-map* $\mathfrak{G}$)
⟨*proof*⟩

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

29.14.3 Arrow map

lemma *cf-nt-ArrMap-vsv*[*cat-cs-intros*]: *vsv* (*cf-nt* α β $\mathfrak{C}$(|*ArrMap*|))
⟨*proof*⟩

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

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

lemma (**in** *is-functor*) *cf-nt-ArrMap-app*[*cat-cs-simps*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{N}f$ = [*ntcf-arrow* $\mathfrak{N}$, f]$_\circ$
and $\mathfrak{N} : \mathfrak{G} \mapsto_{CF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $f : a \mapsto_{\mathfrak{B}} b$
shows *cf-nt* α β $\mathfrak{F}$(|*ArrMap*|)(|$\mathfrak{N}f$|) = *cf-hom*
(*cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α))
[*ntcf-arrow* (*Hom*$_{A.C\alpha}\mathfrak{B}(f,-) \circ_{NTCF-CF} \mathfrak{F}$), *ntcf-arrow* $\mathfrak{N}$]$_\circ$
⟨*proof*⟩

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

29.14.4 *N*-functor is a functor

lemma (**in** *is-functor*) *cf-nt-is-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$

shows *cf-nt* α β $\mathfrak{F}$: *cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α) $\times_C$ $\mathfrak{B}$ $\mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

lemma (**in** *is-functor*) *cf-nt-is-functor'*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{A}'$ = *cat-FUNCT* α $\mathfrak{A}$ (*cat-Set* α) $\times_C$ $\mathfrak{B}$
and $\mathfrak{B}'$ = *cat-Set* β
and β' = β
shows *cf-nt* α β $\mathfrak{F}$: $\mathfrak{A}'$ $\mapsto\mapsto_{C\beta'}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *is-functor.cf-nt-is-functor'*

29.15 Yoneda natural transformation arrow

29.15.1 Definition and elementary properties

The following subsection is based on the elements of the content of Chapter III-2 in [7].

definition *ntcf-Yoneda-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}$ r =
[
(
$\lambda\psi\in_\circ$*Hom* (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)) (*cf-map* (*Hom*$_{O.C\alpha}\mathfrak{C}(r,-)$)) $\mathfrak{F}$.
Yoneda-map α (*cf-of-cf-map* $\mathfrak{C}$ (*cat-Set* α) $\mathfrak{F}$) r⦇
ntcf-of-ntcf-arrow $\mathfrak{C}$ (*cat-Set* α) ψ
⦈
),
Hom (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)) (*cf-map* (*Hom*$_{O.C\alpha}\mathfrak{C}(r,-)$)) $\mathfrak{F}$,
$\mathfrak{F}$⦇*ObjMap*⦈⦇r⦈
]$_\circ$

Components

lemma *ntcf-Yoneda-arrow-components*:
shows *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}$ r⦇*ArrVal*⦈ =
(
$\lambda\psi\in_\circ$*Hom* (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)) (*cf-map* (*Hom*$_{O.C\alpha}\mathfrak{C}(r,-)$)) $\mathfrak{F}$.
Yoneda-map α (*cf-of-cf-map* $\mathfrak{C}$ (*cat-Set* α) $\mathfrak{F}$) r⦇
ntcf-of-ntcf-arrow $\mathfrak{C}$ (*cat-Set* α) ψ
⦈
)
and [*cat-cs-simps*]: *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}$ r⦇*ArrDom*⦈ =
Hom (*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)) (*cf-map* (*Hom*$_{O.C\alpha}\mathfrak{C}(r,-)$)) $\mathfrak{F}$
and [*cat-cs-simps*]: *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}$ r⦇*ArrCod*⦈ = $\mathfrak{F}$⦇*ObjMap*⦈⦇r⦈
⟨*proof*⟩

29.15.2 Arrow map

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

context *category*
begin

context
fixes $\mathfrak{F}$ :: V

begin

mk-VLambda *ntcf-Yoneda-arrow-components*(*1*)[**where** α=α **and** $\mathfrak{C}$=$\mathfrak{C}$ **and** $\mathfrak{F}$=‹*cf-map* $\mathfrak{F}$›]
|*app ntcf-Yoneda-arrow-app′*|

lemmas *ntcf-Yoneda-arrow-app* =
ntcf-Yoneda-arrow-app′[*unfolded in-Hom-iff*, *cat-cs-simps*]

end

end

lemmas [*cat-cs-simps*] = *category.ntcf-Yoneda-arrow-app*

29.15.3 Several technical lemmas

lemma (**in** *vsv*) *vsv-vrange-VLambda-app*:
assumes *g* ‘ *elts A* = *elts* ($\mathcal{D}_\circ$ *r*)
shows $\mathcal{R}_\circ$ ($\lambda x \in_\circ A$. *r*(|*g x*|)) = $\mathcal{R}_\circ$ *r*
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-vrange-VLambda-app′*:
assumes *g* ‘ *elts A* = *elts* ($\mathcal{D}_\circ$ *r*)
and *R* = $\mathcal{R}_\circ$ *r*
shows $\mathcal{R}_\circ$ ($\lambda x \in_\circ A$. *r*(|*g x*|)) = *R*
⟨*proof*⟩

lemma (**in** *v11*) *v11-VLambda-v11-bij-betw-comp*:
assumes *bij-betw g* (*elts A*) (*elts* ($\mathcal{D}_\circ$ *r*))
shows *v11* ($\lambda x \in_\circ A$. *r*(|*g x*|))
⟨*proof*⟩

29.15.4 Yoneda natural transformation arrow is an arrow in the category *Set*

lemma (**in** *category*) *cat-ntcf-Yoneda-arrow-is-arr-isomoprhism*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and *r* $\in_\circ$ $\mathfrak{C}$(|*Obj*|)
shows *ntcf-Yoneda-arrow* α $\mathfrak{C}$ (*cf-map* $\mathfrak{F}$) *r* :
Hom
(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α))
(*cf-map* (*Hom*$_{O.C\alpha}$$\mathfrak{C}$(*r*,−)))
(*cf-map* $\mathfrak{F}$) $\mapsto_{iso}$ *cat-Set* β
$\mathfrak{F}$(|*ObjMap*|)(|*r*|)
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Yoneda-arrow-is-arr-isomoprhism′*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{F}'$ = *cf-map* $\mathfrak{F}$
and *B* = $\mathfrak{F}$(|*ObjMap*|)(|*r*|)
and *A* = *Hom*
(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α))
(*cf-map* (*Hom*$_{O.C\alpha}$$\mathfrak{C}$(*r*,−)))
(*cf-map* $\mathfrak{F}$)
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and *r* $\in_\circ$ $\mathfrak{C}$(|*Obj*|)

shows *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}'$ r : $A \mapsto_{iso cat\text{-}Set\ \beta} B$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] =
category.cat-ntcf-Yoneda-arrow-is-arr-isomoprhism'

lemma (**in** *category*) *cat-ntcf-Yoneda-arrow-is-arr*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
and $r \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-Yoneda-arrow* α $\mathfrak{C}$ (*cf-map* $\mathfrak{F}$) r :
Hom
(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α))
(*cf-map* ($Hom_{O.C\alpha}\mathfrak{C}(r,-)$))
(*cf-map* $\mathfrak{F}$) $\mapsto_{cat\text{-}Set\ \beta}$
$\mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Yoneda-arrow-is-arr'*[*cat-cs-intros*]:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{F}'$ = *cf-map* $\mathfrak{F}$
and $B = \mathfrak{F}(\!|ObjMap|\!)(\!|r|\!)$
and A = *Hom*
(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α))
(*cf-map* ($Hom_{O.C\alpha}\mathfrak{C}(r,-)$))
(*cf-map* $\mathfrak{F}$)
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
and $r \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows *ntcf-Yoneda-arrow* α $\mathfrak{C}$ $\mathfrak{F}'$ r : $A \mapsto_{cat\text{-}Set\ \beta} B$
⟨*proof*⟩

lemmas [*cat-arrow-cs-intros*] = *category.cat-ntcf-Yoneda-arrow-is-arr'*

29.16 Commutativity law for the Yoneda natural transformation arrow

lemma (**in** *category*) *cat-ntcf-Yoneda-arrow-commute*:
assumes $\mathcal{Z}$ β
and $\alpha \in_\circ \beta$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
and $f : a \mapsto_{\mathfrak{C}} b$
shows
ntcf-Yoneda-arrow α $\mathfrak{C}$ (*cf-map* $\mathfrak{G}$) b $\circ_{A\, cat\text{-}Set\ \beta}$
cf-hom
(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α))
[*ntcf-arrow* $Hom_{A.C\alpha}\mathfrak{C}(f,-)$, *ntcf-arrow* $\mathfrak{N}$]$_\circ$ =
cf-eval-arrow $\mathfrak{C}$ (*ntcf-arrow* $\mathfrak{N}$) f $\circ_{A\, cat\text{-}Set\ \beta}$
ntcf-Yoneda-arrow α $\mathfrak{C}$ (*cf-map* $\mathfrak{F}$) a
⟨*proof*⟩

29.17 Yoneda Lemma: naturality

29.17.1 The Yoneda natural transformation: definition and elementary properties

The main result of this subsection corresponds to the corollary to the Yoneda Lemma on page 61 in [7].

definition *ntcf-Yoneda* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-Yoneda* α β $\mathfrak{C}$ =
    [
      (
        $\lambda\mathfrak{F} r\in_\circ$(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$)(⌊*Obj*⌋).
          ntcf-Yoneda-arrow α $\mathfrak{C}$ ($\mathfrak{F} r$(⌊*0*⌋)) ($\mathfrak{F} r$(⌊$1_{\mathbb{N}}$⌋))
      ),
      cf-nt α β (*cf-id* $\mathfrak{C}$),
      cf-eval α β $\mathfrak{C}$,
      cat-FUNCT α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$,
      cat-Set β
    ]$_\circ$

Components.

lemma *ntcf-Yoneda-components*:
  shows *ntcf-Yoneda* α β $\mathfrak{C}$(⌊*NTMap*⌋) =
    (
      $\lambda\mathfrak{F} r\in_\circ$(*cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$)(⌊*Obj*⌋).
        ntcf-Yoneda-arrow α $\mathfrak{C}$ ($\mathfrak{F} r$(⌊*0*⌋)) ($\mathfrak{F} r$(⌊$1_{\mathbb{N}}$⌋))
)
    and [*cat-cs-simps*]: *ntcf-Yoneda* α β $\mathfrak{C}$(⌊*NTDom*⌋) = *cf-nt* α β (*cf-id* $\mathfrak{C}$)
    and [*cat-cs-simps*]: *ntcf-Yoneda* α β $\mathfrak{C}$(⌊*NTCod*⌋) = *cf-eval* α β $\mathfrak{C}$
    and [*cat-cs-simps*]:
      ntcf-Yoneda α β $\mathfrak{C}$(⌊*NTDGDom*⌋) = *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$
    and [*cat-cs-simps*]: *ntcf-Yoneda* α β $\mathfrak{C}$(⌊*NTDGCod*⌋) = *cat-Set* β
  ⟨*proof*⟩

29.17.2 Natural transformation map

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

lemma (**in** *category*) *ntcf-Yoneda-NTMap-app*[*cat-cs-simps*]:
  assumes $\mathcal{Z}$ β
    and $\alpha \in_\circ \beta$
    and $\mathfrak{F} r$ = [*cf-map* $\mathfrak{F}$, r]$_\circ$
    and $\mathfrak{F}$: $\mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
    and $r \in_\circ$ $\mathfrak{C}$(⌊*Obj*⌋)
  shows *ntcf-Yoneda* α β $\mathfrak{C}$(⌊*NTMap*⌋)(⌊$\mathfrak{F} r$⌋) = *ntcf-Yoneda-arrow* α $\mathfrak{C}$ (*cf-map* $\mathfrak{F}$) r
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.ntcf-Yoneda-NTMap-app*

29.17.3 The Yoneda natural transformation is a natural transformation

lemma (**in** *category*) *cat-ntcf-Yoneda-is-ntcf*:
  assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
  shows *ntcf-Yoneda* α β $\mathfrak{C}$:
    cf-nt α β (*cf-id* $\mathfrak{C}$) $\mapsto_{CF.iso}$ *cf-eval* α β $\mathfrak{C}$:
    cat-FUNCT α $\mathfrak{C}$ (*cat-Set* α) $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\beta}$ *cat-Set* β
⟨*proof*⟩

29.18 *Hom*-map

This subsection presents some of the results stated as Corollary 2 in subsection 1.15 in [3] and the corollary following the statement of the Yoneda Lemma on page 61 in [7] in a variety of forms.

29.18.1 Definition and elementary properties

The following function makes an explicit appearance in subsection 1.15 in [3].

definition *ntcf-Hom-map* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntcf-Hom-map* $\alpha\ \mathfrak{C}\ a\ b = (\lambda f \in_{\circ} Hom\ \mathfrak{C}\ a\ b.\ Hom_{A.C\alpha}\mathfrak{C}(f,-))$

Elementary properties.

mk-VLambda *ntcf-Hom-map-def*
  |*vsv ntcf-Hom-map-vsv*|
  |*vdomain ntcf-Hom-map-vdomain*[*cat-cs-simps*]|
  |*app ntcf-Hom-map-app*[*unfolded in-Hom-iff*, *cat-cs-simps*]|

29.18.2 *Hom*-map is a bijection

lemma (in *category*) *cat-ntcf-Hom-snd-is-ntcf-Hom-snd-unique*:
  — The following lemma approximately corresponds to the corollary on page 61 in [7].
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows *Yoneda-map* $\alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!) : s \mapsto_{\mathfrak{C}} r$
    and $\mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!),-)$
    and $\bigwedge f.\ [\![\ f \in_{\circ} \mathfrak{C}(\!|Arr|\!);\ \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(f,-)\ ]\!] \Longrightarrow$
      $f = Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!)$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-fst-is-ntcf-Hom-fst-unique*:
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows *Yoneda-map* $\alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!) : r \mapsto_{\mathfrak{C}} s$
    and $\mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(-, Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!))$
    and $\bigwedge f.\ [\![\ f \in_{\circ} \mathfrak{C}(\!|Arr|\!);\ \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(-,f)\ ]\!] \Longrightarrow$
      $f = Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!)$
  ⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-snd-is-ntcf-Hom-snd-unique′*:
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $\exists!f.\ f \in_{\circ} \mathfrak{C}(\!|Arr|\!) \wedge \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(f,-)$
  ⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-fst-is-ntcf-Hom-fst-unique′*:
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $\exists!f.\ f \in_{\circ} \mathfrak{C}(\!|Arr|\!) \wedge \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(-,f)$
  ⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-snd-inj*:
  assumes $Hom_{A.C\alpha}\mathfrak{C}(g,-) = Hom_{A.C\alpha}\mathfrak{C}(f,-)$
    and $g : a \mapsto_{\mathfrak{C}} b$
    and $f : a \mapsto_{\mathfrak{C}} b$
  shows $g = f$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-fst-inj*:
  assumes $Hom_{A.C\alpha}\mathfrak{C}(-,g) = Hom_{A.C\alpha}\mathfrak{C}(-,f)$

and $g : a \mapsto_{\mathfrak{C}} b$
and $f : a \mapsto_{\mathfrak{C}} b$
shows $g = f$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-Hom-map*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *v11* (*ntcf-Hom-map* α $\mathfrak{C}$ a b)
and $\mathcal{R}_{\circ}$ (*ntcf-Hom-map* α $\mathfrak{C}$ a b) =
these-ntcfs α $\mathfrak{C}$ (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(b,-)$ $Hom_{O.C\alpha}\mathfrak{C}(a,-)$
and (*ntcf-Hom-map* α $\mathfrak{C}$ a b)$^{-1}_{\circ}$ =
$(\lambda\mathfrak{N}\in_{\circ}$*these-ntcfs* α $\mathfrak{C}$ (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(b,-)$ $Hom_{O.C\alpha}\mathfrak{C}(a,-)$.
Yoneda-map α $Hom_{O.C\alpha}\mathfrak{C}(a,-)$ $b(\!|\mathfrak{N}|\!)$)
⟨*proof*⟩

29.18.3 Inverse of a *Hom*-map

lemma (**in** *category*) *inv-ntcf-Hom-map-v11*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *v11* ((*ntcf-Hom-map* α $\mathfrak{C}$ a b)$^{-1}_{\circ}$)
⟨*proof*⟩

lemma (**in** *category*) *inv-ntcf-Hom-map-vdomain*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_{\circ}$ ((*ntcf-Hom-map* α $\mathfrak{C}$ a b)$^{-1}_{\circ}$) =
these-ntcfs α $\mathfrak{C}$ (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(b,-)$ $Hom_{O.C\alpha}\mathfrak{C}(a,-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-ntcf-Hom-map-vdomain*

lemma (**in** *category*) *inv-ntcf-Hom-map-app*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(b,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(a,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
shows (*ntcf-Hom-map* α $\mathfrak{C}$ a b)$^{-1}_{\circ}(\!|\mathfrak{N}|\!)$ = *Yoneda-map* α $Hom_{O.C\alpha}\mathfrak{C}(a,-)$ $b(\!|\mathfrak{N}|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-ntcf-Hom-map-app*

lemma *inv-ntcf-Hom-map-vrange*: $\mathcal{R}_{\circ}$ ((*ntcf-Hom-map* α $\mathfrak{C}$ a b)$^{-1}_{\circ}$) = *Hom* $\mathfrak{C}$ a b
⟨*proof*⟩

29.18.4 *Hom*-natural transformation and isomorphisms

This subsection presents further results that were stated as Corollary 2 in subsection 1.15 in [3].

lemma (**in** *category*) *cat-is-iso-arr-ntcf-Hom-snd-is-iso-ntcf*:
assumes $f : s \mapsto_{iso\mathfrak{C}} r$
shows $Hom_{A.C\alpha}\mathfrak{C}(f,-)$:
$Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-is-iso-arr-ntcf-Hom-fst-is-iso-ntcf*:
assumes $f : r \mapsto_{iso\mathfrak{C}} s$
shows $Hom_{A.C\alpha}\mathfrak{C}(-,f)$:
$Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-snd-is-iso-ntcf-Hom-snd-unique*:
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} : Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!) : s \mapsto_{iso\mathfrak{C}} r$
    and $\mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!),-)$
    and $\bigwedge f.\ [\![f \in_{\circ} \mathfrak{C}(\!|Arr|\!);\ \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(f,-)]\!] \Longrightarrow$
      $f = Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(s,-)\ r(\!|\mathfrak{N}|\!)$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-fst-is-iso-ntcf-Hom-fst-unique*:
  assumes $r \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $s \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
    and $\mathfrak{N} :$
    $Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!) : r \mapsto_{iso\mathfrak{C}} s$
    and $\mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(-, Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!))$
    and $\bigwedge f.\ [\![f \in_{\circ} \mathfrak{C}(\!|Arr|\!);\ \mathfrak{N} = Hom_{A.C\alpha}\mathfrak{C}(-,f)]\!] \Longrightarrow$
      $f = Yoneda\text{-}map\ \alpha\ Hom_{O.C\alpha}\mathfrak{C}(-,s)\ r(\!|\mathfrak{N}|\!)$
  ⟨*proof*⟩

lemma (in *category*) *cat-is-iso-arr-if-ntcf-Hom-snd-is-iso-ntcf*:
  assumes $f : s \mapsto_{\mathfrak{C}} r$
    and $Hom_{A.C\alpha}\mathfrak{C}(f,-) :$
      $Hom_{O.C\alpha}\mathfrak{C}(r,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(s,-) : \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $f : s \mapsto_{iso\mathfrak{C}} r$
⟨*proof*⟩

lemma (in *category*) *cat-is-iso-arr-if-ntcf-Hom-fst-is-iso-ntcf*:
  assumes $f : r \mapsto_{\mathfrak{C}} s$
    and $Hom_{A.C\alpha}\mathfrak{C}(-,f) :$
      $Hom_{O.C\alpha}\mathfrak{C}(-,r) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(-,s) : op\text{-}cat\ \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
  shows $f : r \mapsto_{iso\mathfrak{C}} s$
⟨*proof*⟩

29.18.5 The relationship between a *Hom*-natural transformation and the compositions of a *Hom*-natural transformation and a natural transformation

lemma (in *category*) *cat-ntcf-lcomp-Hom-ntcf-Hom-snd-NTMap-app*:
  assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
    and $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
  shows $Hom_{A.C\alpha}(\varphi-,-)(\!|NTMap|\!)(\!|b,\ c|\!)_{\bullet} = Hom_{A.C\alpha}\mathfrak{C}(\varphi(\!|NTMap|\!)(\!|b|\!),-)(\!|NTMap|\!)(\!|c|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-ntcf-lcomp-Hom-ntcf-Hom-snd-NTMap-app*

lemma (in *category*) *cat-bnt-proj-snd-tcf-lcomp-Hom-ntcf-Hom-snd*:
  assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
    and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
  shows $Hom_{A.C\alpha}(\varphi-,-)_{op\text{-}cat\ \mathfrak{B},\mathfrak{C}}(b,-)_{NTCF} = Hom_{A.C\alpha}\mathfrak{C}(\varphi(\!|NTMap|\!)(\!|b|\!),-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.cat-bnt-proj-snd-tcf-lcomp-Hom-ntcf-Hom-snd*

29.18.6 The relationship between the *Hom*-natural isomorphisms and the compositions of a *Hom*-natural isomorphism and a natural transformation

lemma (in *category*) *cat-ntcf-lcomp-Hom-if-ntcf-Hom-snd-is-iso-ntcf*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\bigwedge b.\ b \in_{\circ} \mathfrak{B}(\!|Obj|\!) \Longrightarrow Hom_{A.C\alpha}\mathfrak{C}(\varphi(\!|NTMap|\!)(\!|b|\!),-) :$
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!),-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!),-) :$
$\mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
shows $Hom_{A.C\alpha}(\varphi-,-) :$
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) :$
$op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

lemma (in *category*) *cat-ntcf-Hom-snd-if-ntcf-lcomp-Hom-is-iso-ntcf*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $Hom_{A.C\alpha}(\varphi-,-) :$
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) :$
$op\text{-}cat\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
and $b \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $Hom_{A.C\alpha}\mathfrak{C}(\varphi(\!|NTMap|\!)(\!|b|\!),-) :$
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}(\!|ObjMap|\!)(\!|b|\!),-) \mapsto_{CF.iso} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!),-) :$
$\mathfrak{C} \mapsto\mapsto_{C\alpha} cat\text{-}Set\ \alpha$
⟨*proof*⟩

29.19 Yoneda map for arbitrary functors

The concept of the Yoneda map for arbitrary functors was developed based on the function that was used in the statement of Lemma 3 in subsection 1.15 in [3].

definition *af-Yoneda-map* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G} =$
$(\lambda\varphi\in_{\circ} these\text{-}ntcfs\ \alpha\ (\mathfrak{F}(\!|HomDom|\!))\ (\mathfrak{F}(\!|HomCod|\!))\ \mathfrak{F}\ \mathfrak{G}.\ Hom_{A.C\alpha}(\varphi-,-))$

Elementary properties.

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

interpretation $\mathfrak{F}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{F}$ ⟨*proof*⟩
interpretation $\mathfrak{G}$: *is-functor* $\alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{G}$ ⟨*proof*⟩

mk-VLambda
af-Yoneda-map-def[**where** $\mathfrak{F}$=$\mathfrak{F}$ **and** $\mathfrak{G}$=$\mathfrak{G}$, *unfolded* $\mathfrak{F}$.*cf-HomDom* $\mathfrak{F}$.*cf-HomCod*]
|*vsv af-Yoneda-map-vsv*|
|*vdomain af-Yoneda-map-vdomain*[*cat-cs-simps*]|
|*app af-Yoneda-map-app*[*unfolded these-ntcfs-iff*, *cat-cs-simps*]|

end

29.20 Yoneda arrow for arbitrary functors

29.20.1 Definition and elementary properties

The following natural transformation is used in the proof of Lemma 3 in subsection 1.15 in [3].

definition *af-Yoneda-arrow* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *af-Yoneda-arrow* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N} =$

```
  [
    (
      λb∈₀(𝔉(⁅HomDom⁆))(⁅Obj⁆).
        Yoneda-map α Hom_{O.Cα}𝔉(⁅HomCod⁆)(𝔉(⁅ObjMap⁆)(⁅b⁆),−) (𝔊(⁅ObjMap⁆)(⁅b⁆))(⁅
          𝔑_{op-cat (𝔉(⁅HomDom⁆)),𝔉(⁅HomCod⁆)}^{(b,−)_{NTCF}}
        ⁆)
    ),
    𝔉,
    𝔊,
    𝔉(⁅HomDom⁆),
    𝔉(⁅HomCod⁆)
  ]₀
```

Components.

lemma *af-Yoneda-arrow-components*:
shows *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$(⁅*NTMap*⁆) =
(
λb∈₀𝔉(⁅HomDom⁆)(⁅Obj⁆).
Yoneda-map α $Hom_{O.C\alpha}$𝔉(⁅HomCod⁆)(𝔉(⁅ObjMap⁆)(⁅b⁆),−) (𝔊(⁅ObjMap⁆)(⁅b⁆))(⁅
$\mathfrak{N}_{op\text{-}cat\ (\mathfrak{F}(⁅HomDom⁆)),\mathfrak{F}(⁅HomCod⁆)}^{(b,-)_{NTCF}}$
⁆)
)
and *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$(⁅*NTDom*⁆) = $\mathfrak{F}$
and *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$(⁅*NTCod*⁆) = $\mathfrak{G}$
and *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$(⁅*NTDGDom*⁆) = $\mathfrak{F}$(⁅*HomDom*⁆)
and *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$(⁅*NTDGCod*⁆) = $\mathfrak{F}$(⁅*HomCod*⁆)
⟨*proof*⟩

29.20.2 Natural transformation map

mk-VLambda *af-Yoneda-arrow-components*(*1*)
|*vsv af-Yoneda-arrow-NTMap-vsv*|

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

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

mk-VLambda
af-Yoneda-arrow-components(*1*)[**where** $\mathfrak{F}$=$\mathfrak{F}$, *unfolded* $\mathfrak{F}$.*cf-HomDom* $\mathfrak{F}$.*cf-HomCod*]
|*vdomain af-Yoneda-arrow-NTMap-vdomain*[*cat-cs-simps*]|
|*app af-Yoneda-arrow-NTMap-app*[*cat-cs-simps*]|

end

lemma (**in** *category*) *cat-af-Yoneda-arrow-is-ntcf*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N}$:
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$:
op-cat $\mathfrak{B}$ $\times_C$ $\mathfrak{C}$ $\mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows *af-Yoneda-arrow* α $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$: $\mathfrak{F} \mapsto_{CF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-af-Yoneda-arrow-is-ntcf'*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$

and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N}$:
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$:
op-cat $\mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
and $\beta = \alpha$
and $\mathfrak{F}' = \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{G}$
shows *af-Yoneda-arrow* $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N} : \mathfrak{F}' \mapsto_{CF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\beta} \mathfrak{C}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] = *category.cat-af-Yoneda-arrow-is-ntcf'*

29.20.3 Yoneda Lemma for arbitrary functors

The following lemmas correspond to variants of the elements of Lemma 3 in subsection 1.15 in [3].

lemma (**in** *category*) *cat-af-Yoneda-map-af-Yoneda-arrow-app*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N}$:
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$:
op-cat $\mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows $\mathfrak{N} = Hom_{A.C\alpha}(af\text{-}Yoneda\text{-}arrow\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}-,-)$
⟨*proof*⟩

lemma (**in** *category*) *cat-af-Yoneda-Lemma*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *v11* (*af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G}$)
and $\mathcal{R}_\circ$ (*af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G}$) =
these-ntcfs α (*op-cat* $\mathfrak{B} \times_C \mathfrak{C}$) (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$ $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$
and (*af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G}$)$^{-1}{}_\circ$ =
(
$\lambda\mathfrak{N}\epsilon_\circ$ *these-ntcfs*
α (*op-cat* $\mathfrak{B} \times_C \mathfrak{C}$) (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$ $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$.
af-Yoneda-arrow $\alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$
)
⟨*proof*⟩

29.20.4 Inverse of the Yoneda map for arbitrary functors

lemma (**in** *category*) *inv-af-Yoneda-map-v11*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows *v11* ((*af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G}$)$^{-1}{}_\circ$)
⟨*proof*⟩

lemma (**in** *category*) *inv-af-Yoneda-map-vdomain*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{D}_\circ$ ((*af-Yoneda-map* $\alpha\ \mathfrak{F}\ \mathfrak{G}$)$^{-1}{}_\circ$) =
these-ntcfs α (*op-cat* $\mathfrak{B} \times_C \mathfrak{C}$) (*cat-Set* α) $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$ $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-af-Yoneda-map-vdomain*

lemma (**in** *category*) *inv-af-Yoneda-map-app*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N}$:
$Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-) \mapsto_{CF} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-)$:

op-cat $\mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha}$ *cat-Set* α
shows $(\textit{af-Yoneda-map}\ \alpha\ \mathfrak{F}\ \mathfrak{G})^{-1}{}_\circ(\lfloor\mathfrak{N}\rfloor) = \textit{af-Yoneda-arrow}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = *category.inv-af-Yoneda-map-app*

lemma (**in** *category*) *inv-af-Yoneda-map-vrange*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$ **and** $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\mathcal{R}_\circ\ ((\textit{af-Yoneda-map}\ \alpha\ \mathfrak{F}\ \mathfrak{G})^{-1}{}_\circ) = \textit{these-ntcfs}\ \alpha\ \mathfrak{B}\ \mathfrak{C}\ \mathfrak{F}\ \mathfrak{G}$
⟨*proof*⟩

29.20.5 Yoneda map for arbitrary functors and natural isomorphisms

The following lemmas correspond to variants of the elements of Lemma 3 in subsection 1.15 in [3].

lemma (**in** *category*) *cat-ntcf-lcomp-Hom-is-iso-ntcf-if-is-iso-ntcf*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
shows $\textit{Hom}_{A.C\alpha}(\varphi -,-) :$
$\textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G} -,-) \mapsto_{CF.iso} \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F} -,-) :$
$\textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
⟨*proof*⟩

lemma (**in** *category*) *cat-ntcf-lcomp-Hom-is-iso-ntcf-if-is-iso-ntcf′*:
assumes $\varphi : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\beta = \alpha$
and $\mathfrak{G}' = \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G} -,-)$
and $\mathfrak{F}' = \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F} -,-)$
and $\mathfrak{B}' = \textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C}$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows $\textit{Hom}_{A.C\alpha}(\varphi -,-) : \mathfrak{G}' \mapsto_{CF.iso} \mathfrak{F}' : \mathfrak{B}' \mapsto\mapsto_{C\beta} \mathfrak{C}'$
⟨*proof*⟩

lemmas [*cat-cs-intros*] =
category.cat-ntcf-lcomp-Hom-is-iso-ntcf-if-is-iso-ntcf′

lemma (**in** *category*) *cat-aYa-is-iso-ntcf-if-ntcf-lcomp-Hom-is-iso-ntcf*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} :$
$\textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G} -,-) \mapsto_{CF.iso} \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F} -,-) :$
$\textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
shows $\textit{af-Yoneda-arrow}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N} : \mathfrak{F} \mapsto_{CF.iso} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *category*) *cat-aYa-is-iso-ntcf-if-ntcf-lcomp-Hom-is-iso-ntcf′*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{N} :$
$\textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{G} -,-) \mapsto_{CF.iso} \textit{Hom}_{O.C\alpha}\mathfrak{C}(\mathfrak{F} -,-) :$
$\textit{op-cat}\ \mathfrak{B} \times_C \mathfrak{C} \mapsto\mapsto_{C\alpha} \textit{cat-Set}\ \alpha$
and $\beta = \alpha$
and $\mathfrak{F}' = \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{G}$
shows $\textit{af-Yoneda-arrow}\ \alpha\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N} : \mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
⟨*proof*⟩

lemmas [*cat-cs-intros*] =

category.cat-aYa-is-iso-ntcf-if-ntcf-lcomp-Hom-is-iso-ntcf′

lemma (**in** *category*) *cat-iso-functor-if-cf-lcomp-Hom-iso-functor*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) \approx_{CF\alpha} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$
shows $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-lcomp-Hom-iso-functor-if-iso-functor*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) \approx_{CF\alpha} Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{G}-,-)$
⟨*proof*⟩

lemma (**in** *category*) *cat-cf-lcomp-Hom-iso-functor-if-iso-functor′*:
assumes $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{C\alpha} \mathfrak{C}$
and $\mathfrak{F} \approx_{CF\alpha} \mathfrak{G}$
and $\alpha' = \alpha$
and $\mathfrak{C}' = \mathfrak{C}$
shows $Hom_{O.C\alpha}\mathfrak{C}(\mathfrak{F}-,-) \approx_{CF\alpha} Hom_{O.C_{\alpha'}}\mathfrak{C}'(\mathfrak{G}-,-)$
⟨*proof*⟩

lemmas [*cat-cs-intros*] =
category.cat-cf-lcomp-Hom-iso-functor-if-iso-functor′

29.21 The Yoneda Functor

29.21.1 Definition and elementary properties

See Chapter III-2 in [7].

definition *Yoneda-functor* :: $V \Rightarrow V \Rightarrow V$
where *Yoneda-functor* $\alpha\ \mathfrak{D}$ =
[
$(\lambda r \in_\circ op\text{-}cat\ \mathfrak{D}(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}\mathfrak{D}(r,-)))$,
$(\lambda f \in_\circ op\text{-}cat\ \mathfrak{D}(\!|Arr|\!).\ ntcf\text{-}arrow\ (Hom_{A.C\alpha}\mathfrak{D}(f,-)))$,
op-cat $\mathfrak{D}$,
cat-FUNCT $\alpha\ \mathfrak{D}$ (*cat-Set* α)
]$_\circ$

Components.

lemma *Yoneda-functor-components*:
shows *Yoneda-functor* $\alpha\ \mathfrak{D}(\!|ObjMap|\!)$ =
$(\lambda r \in_\circ op\text{-}cat\ \mathfrak{D}(\!|Obj|\!).\ cf\text{-}map\ (Hom_{O.C\alpha}\mathfrak{D}(r,-)))$
and *Yoneda-functor* $\alpha\ \mathfrak{D}(\!|ArrMap|\!)$ =
$(\lambda f \in_\circ op\text{-}cat\ \mathfrak{D}(\!|Arr|\!).\ ntcf\text{-}arrow\ (Hom_{A.C\alpha}\mathfrak{D}(f,-)))$
and *Yoneda-functor* $\alpha\ \mathfrak{D}(\!|HomDom|\!)$ = *op-cat* $\mathfrak{D}$
and *Yoneda-functor* $\alpha\ \mathfrak{D}(\!|HomCod|\!)$ = *cat-FUNCT* $\alpha\ \mathfrak{D}$ (*cat-Set* α)
⟨*proof*⟩

29.21.2 Object map

mk-VLambda *Yoneda-functor-components*(*1*)
|*vsv Yoneda-functor-ObjMap-vsv*[*cat-cs-intros*]|
|*vdomain Yoneda-functor-ObjMap-vdomain*[*cat-cs-simps*]|
|*app Yoneda-functor-ObjMap-app*[*cat-cs-simps*]|

lemma (in *category*) *Yoneda-functor-ObjMap-vrange*:
$\mathcal{R}_\circ$ (*Yoneda-functor* α $\mathfrak{C}$(|*ObjMap*|)) $\subseteq_\circ$ *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)(|*Obj*|)
⟨*proof*⟩

29.21.3 Arrow map

mk-VLambda *Yoneda-functor-components*(*2*)
|*vsv Yoneda-functor-ArrMap-vsv*[*cat-cs-intros*]|
|*vdomain Yoneda-functor-ArrMap-vdomain*[*cat-cs-simps*]|
|*app Yoneda-functor-ArrMap-app*[*cat-cs-simps*]|

lemma (in *category*) *Yoneda-functor-ArrMap-vrange*:
$\mathcal{R}_\circ$ (*Yoneda-functor* α $\mathfrak{C}$(|*ArrMap*|)) $\subseteq_\circ$ *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)(|*Arr*|)
⟨*proof*⟩

29.21.4 The Yoneda Functor is a fully faithful functor

lemma (in *category*) *cat-Yoneda-functor-is-functor*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *Yoneda-functor* α $\mathfrak{C}$: *op-cat* $\mathfrak{C}$ $\mapsto\mapsto_{C.ff\beta}$ *cat-FUNCT* α $\mathfrak{C}$ (*cat-Set* α)
⟨*proof*⟩

30 Orders

30.1 Background

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

30.2 Preorder category

See Chapter I-2 in [7].

locale *cat-preorder* = *category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  assumes *cat-peo*:
    $[\![\ a \in_\circ \mathfrak{C}(\!|Obj|\!);\ b \in_\circ \mathfrak{C}(\!|Obj|\!)\ ]\!] \Longrightarrow$
      $(\exists f.\ Hom\ \mathfrak{C}\ a\ b = set\ \{f\}) \vee (Hom\ \mathfrak{C}\ a\ b = 0)$

Rules.

lemma (**in** *cat-preorder*) *cat-preorder-axioms'*[*cat-order-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *cat-preorder* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *cat-preorder-def*[*unfolded cat-preorder-axioms-def*]
  |*intro cat-preorderI*|
  |*dest cat-preorderD*[*dest*]|
  |*elim cat-preorderE*[*elim*]|

lemmas [*cat-order-cs-intros*] = *cat-preorderD*(*1*)

Elementary properties.

lemma (**in** *cat-preorder*) *cat-peo-HomE*:
  assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
  obtains f **where** ‹$Hom\ \mathfrak{C}\ a\ b = set\ \{f\}$› | ‹$Hom\ \mathfrak{C}\ a\ b = 0$›
  ⟨*proof*⟩

lemma (**in** *cat-preorder*) *cat-peo-is-thin-category*:
  — The statement of the lemma appears in nLab [1][16].
  assumes $f : a \mapsto_{\mathfrak{C}} b$ **and** $g : a \mapsto_{\mathfrak{C}} b$
  shows $f = g$
⟨*proof*⟩

30.3 Order relation

30.3.1 Definition and elementary properties

definition *is-le* :: $V \Rightarrow V \Rightarrow V \Rightarrow$ *bool* (**infix** ‹$\leq_{O^1}$› *50*)
  where $a \leq_{O\mathfrak{C}} b \longleftrightarrow Hom\ \mathfrak{C}\ a\ b \neq 0$

Rules.

mk-ide *is-le-def*
  |*intro is-leI*|
  |*dest is-leD*[*dest*]|
  |*elim is-leE*[*elim*]|

Elementary properties.

lemma (**in** *cat-preorder*) *cat-peo-is-le*[*cat-order-cs-intros*]:
  assumes $f : a \mapsto_{\mathfrak{C}} b$

[16] https://ncatlab.org/nlab/show/preorder

shows $a \le_{O\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*cat-order-cs-intros*] = *cat-preorder.cat-peo-is-le*

lemma (**in** *cat-preorder*) *cat-peo-is-le-ex1*:
assumes $a \le_{O\mathfrak{C}} b$ **and** $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $\exists!f.\ f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *cat-preorder*) *cat-peo-is-le-ex*[*elim*]:
assumes $a \le_{O\mathfrak{C}} b$ **and** $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
obtains f **where** $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

30.3.2 Order relation on a preorder category is a preorder

lemma (**in** *cat-preorder*) *is-le-refl*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows $a \le_{O\mathfrak{C}} a$
⟨*proof*⟩

lemma (**in** *cat-preorder*) *is-le-trans*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $c \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $a \le_{O\mathfrak{C}} b$
and $b \le_{O\mathfrak{C}} c$
shows $a \le_{O\mathfrak{C}} c$
⟨*proof*⟩

30.4 Partial order category

See Chapter I-2 in [7].

locale *cat-partial-order* = *cat-preorder* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
assumes *cat-po*: $[\![\ a \in_{\circ} \mathfrak{C}(\!|Obj|\!);\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!);\ a \le_{O\mathfrak{C}} b;\ b \le_{O\mathfrak{C}} a\ ]\!] \Longrightarrow a = b$

Rules.

lemma (**in** *cat-partial-order*) *cat-partial-order-axioms'*[*cat-order-cs-intros*]:
assumes $\alpha' = \alpha$
shows *cat-partial-order* α' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cat-partial-order-def*[*unfolded cat-partial-order-axioms-def*]
|*intro cat-partial-orderI*|
|*dest cat-partial-orderD*[*dest*]|
|*elim cat-partial-orderE*[*elim*]|

lemmas [*cat-order-cs-intros*] = *cat-partial-orderD*(*1*)

30.5 Linear order category

See Chapter I-2 in [7].

locale *cat-linear-order* = *cat-partial-order* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
assumes *cat-lo*: $[\![\ a \in_{\circ} \mathfrak{C}(\!|Obj|\!);\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!)\ ]\!] \Longrightarrow a \le_{O\mathfrak{C}} b \vee b \le_{O\mathfrak{C}} a$

Rules.

lemma (**in** *cat-linear-order*) *cat-linear-order-axioms'*[*cat-order-cs-intros*]:
assumes $\alpha' = \alpha$
shows *cat-linear-order* α' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cat-linear-order-def*[*unfolded cat-linear-order-axioms-def*]
|*intro cat-linear-orderI*|
|*dest cat-linear-orderD*[*dest*]|
|*elim cat-linear-orderE*[*elim*]|

lemmas [*cat-order-cs-intros*] = *cat-linear-orderD*(*1*)

30.6 Preorder functor

30.6.1 Definition and elementary properties

See [1][17].

locale *is-preorder-functor* =
is-functor α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ + *HomDom*: *cat-preorder* α $\mathfrak{A}$ + *HomCod*: *cat-preorder* α $\mathfrak{B}$
for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

syntax *-is-preorder-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - $\leq_{C.PEO1}$ -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-preorder-functor* ⇌ *is-preorder-functor*
translations $\mathfrak{F} : \mathfrak{A} \leq_{C.PEO\alpha} \mathfrak{B}$ ⇌ *CONST is-preorder-functor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

Rules.

lemma (**in** *is-preorder-functor*) *is-preorder-functor-axioms'*[*cat-order-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \leq_{C.PEO\alpha'} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-preorder-functor-def*
|*intro is-preorder-functorI*|
|*dest is-preorder-functorD*[*dest*]|
|*elim is-preorder-functorE*[*elim*]|

lemmas [*cat-order-cs-intros*] = *is-preorder-functorD*

30.6.2 A preorder functor is a faithful functor

sublocale *is-preorder-functor* ⊆ *is-ft-functor*
⟨*proof*⟩

lemmas (**in** *is-preorder-functor*) *is-preorder-functor-is-ft-functor* =
is-ft-functor-axioms

lemmas [*cat-order-cs-intros*] =
is-preorder-functor.is-preorder-functor-is-ft-functor

30.6.3 A preorder functor is a monotone function

lemma (**in** *is-preorder-functor*) *cat-peo*:
— Based on [1][18]

[17] https://ncatlab.org/nlab/show/monotone+function
[18] https://ncatlab.org/nlab/show/monotone+function

assumes $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$ **and** $b \in_{\circ} \mathfrak{A}(\!|Obj|\!)$ **and** $a \leq_{O\mathfrak{A}} b$
shows $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \leq_{O\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
⟨*proof*⟩

30.6.4 Composition of preorder functors

lemma *cf-comp-is-preorder-functor*[*cat-order-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \leq_{C.PEO\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \leq_{C.PEO\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{CF} \mathfrak{F} : \mathfrak{A} \leq_{C.PEO\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *cat-preorder*) *cat-peo-cf-is-preorder-functor*:
cf-id $\mathfrak{C} : \mathfrak{C} \leq_{C.PEO\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *cat-preorder*) *cat-peo-cf-is-preorder-functor'*[*cat-order-cs-intros*]:
assumes $\mathfrak{A}' = \mathfrak{C}$ **and** $\mathfrak{B}' = \mathfrak{C}$
shows *cf-id* $\mathfrak{C} : \mathfrak{A}' \leq_{C.PEO\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*cat-order-cs-intros*] = *cat-preorder.cat-peo-cf-is-preorder-functor'*

31 Smallness for orders

31.1 Background

named-theorems *cat-small-order-cs-simps*
named-theorems *cat-small-order-cs-intros*

31.2 Tiny preorder category

locale *cat-tiny-preorder* = *tiny-category* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
assumes *cat-tiny-peo*:
$[\![a \in_{\circ} \mathfrak{C}(\!|Obj|\!);\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!)]\!] \Longrightarrow$
$(\exists f.\ Hom\ \mathfrak{C}\ a\ b = set\ \{f\}) \vee (Hom\ \mathfrak{C}\ a\ b = 0)$

Rules.

lemma (**in** *cat-tiny-preorder*) *cat-tiny-preorder-axioms'*[*cat-order-cs-intros*]:
assumes $\alpha' = \alpha$
shows *cat-tiny-preorder* α' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *cat-tiny-preorder-def*[*unfolded cat-tiny-preorder-axioms-def*]
|*intro cat-tiny-preorderI*|
|*dest cat-tiny-preorderD*[*dest*]|
|*elim cat-tiny-preorderE*[*elim*]|

lemmas [*cat-small-order-cs-intros*] = *cat-tiny-preorderD*(*1*)

Tiny preorder is a preorder.

sublocale *cat-tiny-preorder* ⊆ *cat-preorder*
⟨*proof*⟩

lemmas (**in** *cat-tiny-preorder*) *cat-tiny-peo-is-cat-preoder* = *cat-preorder-axioms*

lemmas [*cat-small-order-cs-intros*] =
cat-tiny-preorder.cat-tiny-peo-is-cat-preoder

31.3 Tiny partial order category

locale *cat-tiny-partial-order* = *cat-tiny-preorder* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  assumes *cat-tiny-po*:
    $[\![\ a \in_\circ \mathfrak{C}(\!|Obj|\!);\ b \in_\circ \mathfrak{C}(\!|Obj|\!);\ a \le_{O\mathfrak{C}} b;\ b \le_{O\mathfrak{C}} a\ ]\!] \Longrightarrow a = b$

Rules.

lemma (**in** *cat-tiny-partial-order*)
  cat-tiny-partial-order-axioms'[*cat-order-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *cat-tiny-partial-order* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *cat-tiny-partial-order-def*[*unfolded cat-tiny-partial-order-axioms-def*]
  |*intro cat-tiny-partial-orderI*|
  |*dest cat-tiny-partial-orderD*[*dest*]|
  |*elim cat-tiny-partial-orderE*[*elim*]|

lemmas [*cat-small-order-cs-intros*] = *cat-tiny-partial-orderD*(*1*)

Tiny partial order is a partial order.

sublocale *cat-tiny-partial-order* ⊆ *cat-partial-order*
  ⟨*proof*⟩

lemmas (**in** *cat-tiny-preorder*) *cat-tiny-po-is-cat-preoder* = *cat-preorder-axioms*

lemmas [*cat-small-order-cs-intros*] =
  cat-tiny-preorder.cat-tiny-peo-is-cat-preoder

lemma *cat-tiny-partial-orderI'*:
  assumes *tiny-category* α $\mathfrak{C}$
    and *cat-partial-order* α $\mathfrak{C}$
  shows *cat-tiny-partial-order* α $\mathfrak{C}$
⟨*proof*⟩

31.4 Tiny linear order category

locale *cat-tiny-linear-order* = *cat-tiny-partial-order* α $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  assumes *cat-tiny-lo*: $[\![\ a \in_\circ \mathfrak{C}(\!|Obj|\!);\ b \in_\circ \mathfrak{C}(\!|Obj|\!)\ ]\!] \Longrightarrow a \le_{O\mathfrak{C}} b \vee b \le_{O\mathfrak{C}} a$

Rules.

lemma (**in** *cat-tiny-linear-order*)
  cat-tiny-linear-order-axioms'[*cat-order-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *cat-tiny-linear-order* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *cat-tiny-linear-order-def*[*unfolded cat-tiny-linear-order-axioms-def*]
  |*intro cat-tiny-linear-orderI*|
  |*dest cat-tiny-linear-orderD*[*dest*]|
  |*elim cat-tiny-linear-orderE*[*elim*]|

lemmas [*cat-small-order-cs-intros*] = *cat-tiny-linear-orderD*(*1*)

Tiny linear order is a partial order.

sublocale *cat-tiny-linear-order* ⊆ *cat-linear-order*
  ⟨*proof*⟩

lemmas (**in** *cat-tiny-linear-order*) *cat-tiny-lo-is-cat-partial-order* =
cat-linear-order-axioms

lemmas [*cat-small-order-cs-intros*] =
cat-tiny-linear-order.cat-tiny-lo-is-cat-partial-order

lemma *cat-tiny-linear-orderI'*:
assumes *tiny-category* α $\mathfrak{C}$ **and** *cat-linear-order* α $\mathfrak{C}$
shows *cat-tiny-linear-order* α $\mathfrak{C}$
⟨*proof*⟩

31.5 Tiny preorder functor

locale *is-tiny-preorder-functor* =
is-functor α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
HomDom: *cat-tiny-preorder* α $\mathfrak{A}$ +
HomCod: *cat-tiny-preorder* α $\mathfrak{B}$
for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

syntax *-is-tiny-preorder-functor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $\leq_{C.PEO.tiny^1}$ -)› [*51*, *51*, *51*] *51*)
syntax-consts *-is-tiny-preorder-functor* $\rightleftharpoons$ *is-tiny-preorder-functor*
translations $\mathfrak{F} : \mathfrak{A} \leq_{C.PEO.tiny\alpha} \mathfrak{B} \rightleftharpoons$
CONST is-tiny-preorder-functor α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

Rules.

lemma (**in** *is-tiny-preorder-functor*)
is-tiny-preorder-functor-axioms'[*cat-order-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{A}' = \mathfrak{A}$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A}' \leq_{C.PEO.tiny_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-tiny-preorder-functor-def*
|*intro is-tiny-preorder-functorI*|
|*dest is-tiny-preorder-functorD*[*dest*]|
|*elim is-tiny-preorder-functorE*[*elim*]|

lemmas [*cat-small-order-cs-intros*] = *is-tiny-preorder-functorD*(*1*)

Tiny preorder functor is a tiny functor

sublocale *is-tiny-preorder-functor* ⊆ *is-tiny-functor*
⟨*proof*⟩

32 Ordinal numbers

32.1 Background

The content of this section is based on the treatment of the ordinal numbers from the perspective of category theory as exposed, for example, in Chapter I-2 in [7].

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

32.2 Arrows associated with an ordinal number

definition *ordinal-arrs* :: $V \Rightarrow V$
where *ordinal-arrs* $A \equiv$ *set* $\{[a, b]_\circ \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge a \leq b\}$

lemma *small-ordinal-arrs*[*simp*]:
small {[*a*, *b*]$_\circ$ | *a b*. *a* $\in_\circ$ *A* $\wedge$ *b* $\in_\circ$ *A* $\wedge$ *a* $\leq$ *b*}
⟨*proof*⟩

Rules.

lemma *ordinal-arrsI*[*cat-ordinal-cs-intros*]:
assumes *x* = [*a*, *b*]$_\circ$ **and** *a* $\in_\circ$ *A* **and** *b* $\in_\circ$ *A* **and** *a* $\leq$ *b*
shows *x* $\in_\circ$ *ordinal-arrs A*
⟨*proof*⟩

lemma *ordinal-arrsD*[*dest*]:
assumes [*a*, *b*]$_\circ$ $\in_\circ$ *ordinal-arrs A*
shows *a* $\in_\circ$ *A* **and** *b* $\in_\circ$ *A* **and** *a* $\leq$ *b*
⟨*proof*⟩

lemma *ordinal-arrsE*[*elim*]:
assumes *x* $\in_\circ$ *ordinal-arrs A*
obtains *a b* **where** *a* $\in_\circ$ *A* **and** *b* $\in_\circ$ *A* **and** *a* $\leq$ *b* **and** *x* = [*a*, *b*]$_\circ$
⟨*proof*⟩

32.3 Composable arrows

abbreviation *ordinal-composable* :: *V* $\Rightarrow$ *V*
where *ordinal-composable A* $\equiv$ *set*
{
[[*b*, *c*]$_\circ$, [*a*, *b*]$_\circ$]$_\circ$ | *a b c*.
a $\in_\circ$ *A* $\wedge$ *b* $\in_\circ$ *A* $\wedge$ *c* $\in_\circ$ *A* $\wedge$ *a* $\leq$ *b* $\wedge$ *b* $\leq$ *c*
}

lemma *small-ordinal-composable*[*simp*]:
small
{
[[*b*, *c*]$_\circ$, [*a*, *b*]$_\circ$]$_\circ$ | *a b c*.
a $\in_\circ$ *A* $\wedge$ *b* $\in_\circ$ *A* $\wedge$ *c* $\in_\circ$ *A* $\wedge$ *a* $\leq$ *b* $\wedge$ *b* $\leq$ *c*
}
⟨*proof*⟩

Rules.

lemma *ordinal-composableI*[*cat-ordinal-cs-intros*]:
assumes *x* = [[*b*, *c*]$_\circ$, [*a*, *b*]$_\circ$]$_\circ$
and *a* $\in_\circ$ *A*
and *b* $\in_\circ$ *A*
and *c* $\in_\circ$ *A*
and *a* $\leq$ *b*
and *b* $\leq$ *c*
shows *x* $\in_\circ$ *ordinal-composable A*
⟨*proof*⟩

lemma *ordinal-composableD*[*dest*]:
assumes [[*b*, *c*]$_\circ$, [*a*, *b*]$_\circ$]$_\circ$ $\in_\circ$ *ordinal-composable A*
shows *a* $\in_\circ$ *A* **and** *b* $\in_\circ$ *A* **and** *c* $\in_\circ$ *A* **and** *a* $\leq$ *b* **and** *b* $\leq$ *c*
⟨*proof*⟩

lemma *ordinal-composableE*[*elim*]:
assumes *x* $\in_\circ$ *ordinal-composable A*
obtains *a b c*
where *x* = [[*b*, *c*]$_\circ$, [*a*, *b*]$_\circ$]$_\circ$

and $a \in_\circ A$
and $b \in_\circ A$
and $c \in_\circ A$
and $a \leq b$
and $b \leq c$
⟨*proof*⟩

32.4 Ordinal number as a category

32.4.1 Definition and elementary properties

definition *cat-ordinal* :: $V \Rightarrow V$
where *cat-ordinal* A =
[
A,
ordinal-arrs A,
$(\lambda f \in_\circ$ *ordinal-arrs* A. $f(\!|0|\!))$,
$(\lambda f \in_\circ$ *ordinal-arrs* A. $f(\!|1_{\mathbb{N}}|\!))$,
$(\lambda gf \in_\circ$ *ordinal-composable* A. $[gf(\!|1_{\mathbb{N}}|\!)(\!|0|\!),\ gf(\!|0|\!)(\!|1_{\mathbb{N}}|\!)]_\circ)$,
$(\lambda x \in_\circ A.\ [x,\ x]_\circ)$
]$_\circ$

Components.

lemma *cat-ordinal-components*:
shows [*cat-ordinal-cs-simps*]: *cat-ordinal* $A(\!|Obj|\!) = A$
and [*cat-ordinal-cs-simps*]: *cat-ordinal* $A(\!|Arr|\!) =$ *ordinal-arrs* A
and *cat-ordinal* $A(\!|Dom|\!) = (\lambda f \in_\circ$ *ordinal-arrs* A. $f(\!|0|\!))$
and *cat-ordinal* $A(\!|Cod|\!) = (\lambda f \in_\circ$ *ordinal-arrs* A. $f(\!|1_{\mathbb{N}}|\!))$
and *cat-ordinal* $A(\!|Comp|\!) =$
$(\lambda gf \in_\circ$ *ordinal-composable* A. $[gf(\!|1_{\mathbb{N}}|\!)(\!|0|\!),\ gf(\!|0|\!)(\!|1_{\mathbb{N}}|\!)]_\circ)$
and *cat-ordinal* $A(\!|CId|\!) = (\lambda x \in_\circ A.\ [x,\ x]_\circ)$
⟨*proof*⟩

32.4.2 Domain

mk-VLambda *cat-ordinal-components*(*3*)
|*vsv cat-ordinal-Dom-vsv*[*cat-ordinal-cs-intros*]|
|*vdomain*
cat-ordinal-Dom-vdomain[
folded cat-ordinal-components, *cat-ordinal-cs-simps*
]
|

lemma *cat-ordinal-Dom-app*[*cat-ordinal-cs-simps*]:
assumes $x \in_\circ$ *cat-ordinal* $A(\!|Arr|\!)$ **and** $x = [a,\ b]_\circ$
shows *cat-ordinal* $A(\!|Dom|\!)(\!|x|\!) = a$
⟨*proof*⟩

lemma *cat-ordinal-Dom-vrange*: $\mathcal{R}_\circ$ (*cat-ordinal* $A(\!|Dom|\!)$) $\subseteq_\circ$ *cat-ordinal* $A(\!|Obj|\!)$
⟨*proof*⟩

32.4.3 Codomain

mk-VLambda *cat-ordinal-components*(*4*)
|*vsv cat-ordinal-Cod-vsv*[*cat-ordinal-cs-intros*]|
|*vdomain*
cat-ordinal-Cod-vdomain[
folded cat-ordinal-components, *cat-ordinal-cs-simps*
]

|

lemma *cat-ordinal-Cod-app*[*cat-ordinal-cs-simps*]:
assumes $x \in_{\circ} \textit{cat-ordinal}\ A(\!|\textit{Arr}|\!)$ **and** $x = [a, b]_{\circ}$
shows $\textit{cat-ordinal}\ A(\!|\textit{Cod}|\!)(\!|x|\!) = b$
⟨*proof*⟩

lemma *cat-ordinal-Cod-vrange*: $\mathcal{R}_{\circ}\ (\textit{cat-ordinal}\ A(\!|\textit{Cod}|\!)) \subseteq_{\circ} \textit{cat-ordinal}\ A(\!|\textit{Obj}|\!)$
⟨*proof*⟩

32.4.4 Arrow with a domain and a codomain

Rules.

lemma *cat-ordinal-is-arrI*[*cat-ordinal-cs-intros*]:
assumes $a \in_{\circ} A$ **and** $b \in_{\circ} A$ **and** $a \leq b$ **and** $f = [a, b]_{\circ}$
shows $f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
⟨*proof*⟩

lemma *cat-ordinal-is-arrD*[*dest*]:
assumes $f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
shows $a \in_{\circ} A$ **and** $b \in_{\circ} A$ **and** $a \leq b$ **and** $f = [a, b]_{\circ}$
⟨*proof*⟩

lemma *cat-ordinal-is-arrE*[*elim*]:
assumes $f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
obtains $a \in_{\circ} A$ **and** $b \in_{\circ} A$ **and** $a \leq b$ **and** $f = [a, b]_{\circ}$
⟨*proof*⟩

Elementary properties.

lemma *cat-ordinal-is-arr-not*:
assumes $\neg a \leq b$
shows $\neg f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
⟨*proof*⟩

lemma *cat-ordinal-is-arr-is-unique*:
assumes $f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$ **and** $g : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
shows $f = g$
⟨*proof*⟩

lemma *cat-ordinal-Hom-ne*:
assumes $f : a \mapsto_{\textit{cat-ordinal}\ A}\ b$
shows $\textit{Hom}\ (\textit{cat-ordinal}\ A)\ a\ b = \textit{set}\ \{f\}$
⟨*proof*⟩

lemma *cat-ordinal-Hom-empty*:
assumes $\neg a \leq b$
shows $\textit{Hom}\ (\textit{cat-ordinal}\ A)\ a\ b = 0$
⟨*proof*⟩

lemma *cat-ordinal-inj*:
assumes $\textit{cat-ordinal}\ m = \textit{cat-ordinal}\ n$
shows $m = n$
⟨*proof*⟩

32.4.5 Composition

mk-VLambda *cat-ordinal-components*(*5*)

|*vsv cat-ordinal-Comp-vsv*[*cat-ordinal-cs-intros*]|
|*vdomain cat-ordinal-Comp-vdomain*[*folded cat-ordinal-components*, *cat-cs-simps*]|

lemma *cat-ordinal-Comp-app*[*cat-ordinal-cs-simps*]:
assumes $g : b \mapsto_{cat\text{-}ordinal\ A}\ c$ **and** $f : a \mapsto_{cat\text{-}ordinal\ A}\ b$
shows $g \circ_{A\ cat\text{-}ordinal\ A}\ f = [a,\ c]_\circ$
⟨*proof*⟩

32.4.6 Identity

mk-VLambda *cat-ordinal-components*(*6*)
|*vsv cat-ordinal-CId-vsv*[*cat-ordinal-cs-intros*]|
|*vdomain cat-ordinal-CId-vdomain*[*cat-ordinal-cs-simps*]|
|*app cat-ordinal-CId-app*[*cat-ordinal-cs-simps*]|

32.4.7 Order relation

lemma *cat-ordinal-is-leD*[*dest*]:
assumes $a \le_{O\ cat\text{-}ordinal\ A}\ b$
shows $[a,\ b]_\circ : a \mapsto_{cat\text{-}ordinal\ A}\ b$
⟨*proof*⟩

lemma *cat-ordinal-is-leE*[*elim*]:
assumes $a \le_{O\ cat\text{-}ordinal\ A}\ b$
obtains $[a,\ b]_\circ : a \mapsto_{cat\text{-}ordinal\ A}\ b$
⟨*proof*⟩

lemma *cat-ordinal-is-le-iff*:
$a \le_{O\ cat\text{-}ordinal\ A}\ b \longleftrightarrow [a,\ b]_\circ : a \mapsto_{cat\text{-}ordinal\ A}\ b$
⟨*proof*⟩

32.4.8 Every ordinal number is a category

lemma (**in** $\mathcal{Z}$) *cat-linear-order-cat-ordinal*[*cat-ordinal-cs-intros*]:
assumes *Ord A* **and** $A \subseteq_\circ \alpha$
shows *cat-linear-order* α (*cat-ordinal A*)
⟨*proof*⟩

lemmas [*cat-ordinal-cs-intros*] = $\mathcal{Z}$.*cat-linear-order-cat-ordinal*

lemma (**in** $\mathcal{Z}$) *cat-tiny-linear-order-cat-ordinal*[*cat-ordinal-cs-intros*]:
assumes *Ord A* **and** $A \in_\circ \alpha$
shows *cat-tiny-linear-order* α (*cat-ordinal A*)
⟨*proof*⟩

lemmas [*cat-ordinal-cs-intros*] = $\mathcal{Z}$.*cat-linear-order-cat-ordinal*

lemma (**in** $\mathcal{Z}$) *finite-category-cat-ordinal*[*cat-ordinal-cs-intros*]:
assumes $a \in_\circ \omega$
shows *finite-category* α (*cat-ordinal a*)
⟨*proof*⟩

lemmas [*cat-ordinal-cs-intros*] = $\mathcal{Z}$.*finite-category-cat-ordinal*

33 Simplicial category

33.1 Background

The content of this section is based, primarily, on the elements of the content of Chapter I-2 in [7].

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

33.2 Composable arrows for simplicial category

definition *composable-cat-simplicial* :: $V \Rightarrow V \Rightarrow V$
  where *composable-cat-simplicial* α A = *set*
    {
      $[g, f]_\circ$ | g f. $\exists$ m n p.
        g : *cat-ordinal* $n \leq_{C.PEO\alpha}$ *cat-ordinal* p $\wedge$
        f : *cat-ordinal* $m \leq_{C.PEO\alpha}$ *cat-ordinal* n $\wedge$
        $m \in_\circ A \wedge n \in_\circ A \wedge p \in_\circ A$
    }

lemma *small-composable-cat-simplicial*[*simp*]:
  small
    {
      $[g, f]_\circ$ | g f. $\exists$ m n p.
        g : *cat-ordinal* $n \leq_{C.PEO\alpha}$ *cat-ordinal* p $\wedge$
        f : *cat-ordinal* $m \leq_{C.PEO\alpha}$ *cat-ordinal* n $\wedge$
        $m \in_\circ A \wedge n \in_\circ A \wedge p \in_\circ A$
    }
  (**is** ‹*small ?S*›)
⟨*proof*⟩

Rules.

lemma *composable-cat-simplicialI*:
  assumes g : *cat-ordinal* $n \leq_{C.PEO\alpha}$ *cat-ordinal* p
    and f : *cat-ordinal* $m \leq_{C.PEO\alpha}$ *cat-ordinal* n
    and $m \in_\circ A$
    and $n \in_\circ A$
    and $p \in_\circ A$
    and $gf = [g, f]_\circ$
  shows $gf \in_\circ$ *composable-cat-simplicial* α A
  ⟨*proof*⟩

lemma *composable-cat-simplicialE*[*elim*]:
  assumes $gf \in_\circ$ *composable-cat-simplicial* α A
  obtains g f m n p **where** $gf = [g, f]_\circ$
    and g : *cat-ordinal* $n \leq_{C.PEO\alpha}$ *cat-ordinal* p
    and f : *cat-ordinal* $m \leq_{C.PEO\alpha}$ *cat-ordinal* n
    and $m \in_\circ A$
    and $n \in_\circ A$
    and $p \in_\circ A$
⟨*proof*⟩

33.3 Simplicial category

33.3.1 Definition and elementary properties

definition *cat-simplicial* :: $V \Rightarrow V \Rightarrow V$
  where *cat-simplicial* α A =

```
[
  set {cat-ordinal m | m. m ∈∘ A},
  set
    {
      f. ∃ m n.
        f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A
    },
  (
    λf∈∘ set
      {
        f. ∃ m n.
          f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A
      }. f(⟦HomDom⟧)
  ),
  (
    λf∈∘ set
      {
        f. ∃ m n.
          f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A
      }. f(⟦HomCod⟧)
  ),
  (λgf∈∘composable-cat-simplicial α A. gf(⟦0⟧) ∘CF gf(⟦1ℕ⟧)),
  (λm∈∘set {cat-ordinal m | m. m ∈∘ A}. cf-id m)
]∘
```

Components.

```
lemma cat-simplicial-components:
  shows cat-simplicial α A(⟦Obj⟧) = set {cat-ordinal m | m. m ∈∘ A}
    and cat-simplicial α A(⟦Arr⟧) =
      set {f. ∃ m n. f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A}
    and cat-simplicial α A(⟦Dom⟧) =
      (
        λf∈∘set
          {
            f. ∃ m n.
              f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A
          }. f(⟦HomDom⟧)
      )
    and cat-simplicial α A(⟦Cod⟧) =
      (
        λf∈∘set
          {
            f. ∃ m n.
              f : cat-ordinal m ≤C.PEOα cat-ordinal n ∧ m ∈∘ A ∧ n ∈∘ A
          }. f(⟦HomCod⟧)
      )
    and cat-simplicial α A(⟦Comp⟧) =
      (λgf∈∘composable-cat-simplicial α A. gf(⟦0⟧) ∘CF gf(⟦1ℕ⟧))
    and cat-simplicial α A(⟦CId⟧) =
      (λm∈∘set {cat-ordinal m | m. m ∈∘ A}. cf-id m)
  ⟨proof⟩
```

33.3.2 Objects

```
lemma cat-simplicial-ObjI[cat-simplicial-cs-intros]:
  assumes m ∈∘ A and a = cat-ordinal m
  shows a ∈∘ cat-simplicial α A(⟦Obj⟧)
  ⟨proof⟩
```

lemma *cat-simplicial-ObjD*:
assumes *cat-ordinal* m $\in_\circ$ *cat-simplicial* α $A(\!|Obj|\!)$
shows $m \in_\circ A$
⟨*proof*⟩

lemma *cat-simplicial-ObjE*:
assumes $M \in_\circ$ *cat-simplicial* α $A(\!|Obj|\!)$
obtains m **where** $M =$ *cat-ordinal* m **and** $m \in_\circ A$
⟨*proof*⟩

33.3.3 Arrows

lemma *small-cat-simplicial-Arr*[*simp*]:
small $\{f.\ \exists\, m\ n.\ f :$ *cat-ordinal* $m \le_{C.PEO\alpha}$ *cat-ordinal* $n \wedge m \in_\circ A \wedge n \in_\circ A\}$
(**is** ‹*small ?S*›)
⟨*proof*⟩

lemma *cat-simplicial-ArrI*[*cat-simplicial-cs-intros*]:
assumes $f :$ *cat-ordinal* $m \le_{C.PEO\alpha}$ *cat-ordinal* n **and** $m \in_\circ A$ **and** $n \in_\circ A$
shows $f \in_\circ$ *cat-simplicial* α $A(\!|Arr|\!)$
⟨*proof*⟩

lemma *cat-simplicial-ArrE*:
assumes $f \in_\circ$ *cat-simplicial* α $A(\!|Arr|\!)$
obtains m n
where $f :$ *cat-ordinal* $m \le_{C.PEO\alpha}$ *cat-ordinal* n **and** $m \in_\circ A$ **and** $n \in_\circ A$
⟨*proof*⟩

33.3.4 Domain

mk-VLambda *cat-simplicial-components*(*3*)
|*vsv cat-simplicial-Dom-vsv*[*cat-simplicial-cs-intros*]|
|*vdomain*
cat-simplicial-Dom-vdomain[
folded cat-simplicial-components, *cat-simplicial-cs-simps*
]
|
|*app cat-simplicial-Dom-app*[*folded cat-simplicial-components*]|

lemma *cat-simplicial-Dom-app′*[*cat-simplicial-cs-simps*]:
assumes $f :$ *cat-ordinal* $m \le_{C.PEO\alpha}$ *cat-ordinal* n **and** $m \in_\circ A$ **and** $n \in_\circ A$
shows *cat-simplicial* α $A(\!|Dom|\!)(\!|f|\!) =$ *cat-ordinal* m
⟨*proof*⟩

33.3.5 Codomain

mk-VLambda *cat-simplicial-components*(*4*)
|*vsv cat-simplicial-Cod-vsv*[*cat-simplicial-cs-intros*]|
|*vdomain*
cat-simplicial-Cod-vdomain[
folded cat-simplicial-components, *cat-simplicial-cs-simps*
]
|
|*app cat-simplicial-Cod-app*[*folded cat-simplicial-components*]|

lemma *cat-simplicial-Cod-app′*[*cat-simplicial-cs-simps*]:
assumes $f :$ *cat-ordinal* $m \le_{C.PEO\alpha}$ *cat-ordinal* n **and** $m \in_\circ A$ **and** $n \in_\circ A$

shows *cat-simplicial* α $A(\!|Cod|\!)(\!|f|\!) = cat\text{-}ordinal\ n$
⟨*proof*⟩

33.3.6 Arrow with a domain and a codomain

lemma *cat-simplicial-is-arrI*:
assumes $f : cat\text{-}ordinal\ m \leq_{C.PEO\alpha} cat\text{-}ordinal\ n$
and $m \in_\circ A$
and $n \in_\circ A$
shows $f : cat\text{-}ordinal\ m \mapsto_{cat\text{-}simplicial\ \alpha\ A} cat\text{-}ordinal\ n$
⟨*proof*⟩

lemma *cat-simplicial-is-arrI'*[*cat-simplicial-cs-intros*]:
assumes $f : cat\text{-}ordinal\ m \leq_{C.PEO\alpha} cat\text{-}ordinal\ n$
and $m \in_\circ A$
and $n \in_\circ A$
and $a = cat\text{-}ordinal\ m$
and $b = cat\text{-}ordinal\ n$
shows $f : a \mapsto_{cat\text{-}simplicial\ \alpha\ A} b$
⟨*proof*⟩

lemma *cat-simplicial-is-arrD*[*dest*]:
assumes $f : cat\text{-}ordinal\ m \mapsto_{cat\text{-}simplicial\ \alpha\ A} cat\text{-}ordinal\ n$
and $m \in_\circ A$
and $n \in_\circ A$
shows $f : cat\text{-}ordinal\ m \leq_{C.PEO\alpha} cat\text{-}ordinal\ n$
⟨*proof*⟩

lemma *cat-simplicial-is-arrE*[*elim*]:
assumes $f : a \mapsto_{cat\text{-}simplicial\ \alpha\ A} b$
obtains $m\ n$ **where** $f : cat\text{-}ordinal\ m \leq_{C.PEO\alpha} cat\text{-}ordinal\ n$
and $m \in_\circ A$
and $n \in_\circ A$
and $a = cat\text{-}ordinal\ m$
and $b = cat\text{-}ordinal\ n$
⟨*proof*⟩

33.3.7 Composition

mk-VLambda *cat-simplicial-components*(5)
|*vsv cat-simplicial-Comp-vsv*[*cat-simplicial-cs-intros*]|
|*vdomain cat-simplicial-Comp-vdomain*[*cat-simplicial-cs-simps*]|

lemma *cat-simplicial-Comp-app*[*cat-simplicial-cs-simps*]:
assumes $g : cat\text{-}ordinal\ n \mapsto_{cat\text{-}simplicial\ \alpha\ A} cat\text{-}ordinal\ p$
and $f : cat\text{-}ordinal\ m \mapsto_{cat\text{-}simplicial\ \alpha\ A} cat\text{-}ordinal\ n$
and $m \in_\circ A$
and $n \in_\circ A$
and $p \in_\circ A$
shows $g \circ_{A\ cat\text{-}simplicial\ \alpha\ A} f = g \circ_{CF} f$
⟨*proof*⟩

33.3.8 Identity

context
fixes $\alpha\ A :: V$
begin

mk-VLambda *cat-simplicial-components*(*6*)[**where** α=α **and** A=A]
|*vsv cat-simplicial-CId-vsv*[*cat-simplicial-cs-intros*]|
|*vdomain*
cat-simplicial-CId-vdomain′[
folded cat-simplicial-components(*1*)[**where** α=α **and** A=A]
]
|
|*app cat-simplicial-CId-app′*[
folded cat-simplicial-components(*1*)[**where** α=α **and** A=A]
]
|

lemmas *cat-simplicial-CId-vdomain*[*cat-simplicial-cs-simps*] =
cat-simplicial-CId-vdomain′
lemmas *cat-simplicial-CId-app*[*cat-simplicial-cs-simps*] =
cat-simplicial-CId-app′

end

33.3.9 Simplicial category is a category

lemma (**in** $\mathcal{Z}$) *category-simplicial*:
assumes *Ord* A **and** $A \subseteq_\circ \alpha$
shows *category* α (*cat-simplicial* α A)
⟨*proof*⟩

34 Example: categories with additional structure

34.1 Background

The examples that are presented in this section showcase how the framework developed in this article can be used for the formalization of the theory of categories with additional structure. The content of this section also indicates some of the potential future directions for this body of work.

34.2 Dagger category

named-theorems *dag-field-simps*

named-theorems *dagcat-cs-simps*
named-theorems *dagcat-cs-intros*

definition *DagCat* :: *V* **where** [*dag-field-simps*]: *DagCat* = *0*
definition *DagDag* :: *V* **where** [*dag-field-simps*]: *DagDag* = $1_{\mathbb{N}}$

abbreviation *DagDag-app* :: *V* ⇒ *V* (‹$\dagger_C$›)
  where $\dagger_C$ $\mathfrak{C}$ ≡ $\mathfrak{C}$(|*DagDag*|)

34.2.1 Definition and elementary properties

For further information see [1][19].

locale *dagger-category* =
  $\mathcal{Z}$ α +
  vfsequence $\mathfrak{C}$ +
  DagCat: *category* α ‹$\mathfrak{C}$(|*DagCat*|)› +
  DagDag: *is-functor* α ‹*op-cat* ($\mathfrak{C}$(|*DagCat*|))› ‹$\mathfrak{C}$(|*DagCat*|)› ‹$\dagger_C$ $\mathfrak{C}$›
  for α $\mathfrak{C}$ +
  assumes *dagcat-length*: *vcard* $\mathfrak{C}$ = $2_{\mathbb{N}}$
    and *dagcat-ObjMap-identity*[*dagcat-cs-simps*]:
      a ∈$_\circ$ $\mathfrak{C}$(|*DagCat*|)(|*Obj*|) ⟹ ($\dagger_C$ $\mathfrak{C}$)(|*ObjMap*|)(|*a*|) = *a*
    and *dagcat-DagCat-idem*[*dagcat-cs-simps*]:
      $\dagger_C$ $\mathfrak{C}$ $_{CF}\circ$ $\dagger_C$ $\mathfrak{C}$ = *cf-id* ($\mathfrak{C}$(|*DagCat*|))

lemmas [*dagcat-cs-simps*] =
  dagger-category.dagcat-ObjMap-identity
  dagger-category.dagcat-DagCat-idem

Rules.

lemma (**in** *dagger-category*) *dagger-category-axioms'*[*dagcat-cs-intros*]:
  assumes α' = α
  shows *dagger-category* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf *dagger-category-def*[*unfolded dagger-category-axioms-def*]
  |*intro dagger-categoryI*|
  |*dest dagger-categoryD*[*dest*]|
  |*elim dagger-categoryE*[*elim*]|

lemma *category-if-dagger-category*[*dagcat-cs-intros*]:
  assumes $\mathfrak{C}'$ = ($\mathfrak{C}$(|*DagCat*|)) **and** *dagger-category* α $\mathfrak{C}$
  shows *category* α $\mathfrak{C}'$

[19] https://ncatlab.org/nlab/show/dagger+category

⟨*proof*⟩

lemma (**in** *dagger-category*) *dagcat-is-functor'*[*dagcat-cs-intros*]:
assumes $\mathfrak{A}' = op\text{-}cat\ (\mathfrak{C}(\!|DagCat|\!))$ **and** $\mathfrak{B}' = \mathfrak{C}(\!|DagCat|\!)$
shows $\dagger_C\ \mathfrak{C} : \mathfrak{A}' \mapsto\mapsto_{C\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*dagcat-cs-intros*] = *dagger-category.dagcat-is-functor'*

34.3 *Rel* as a dagger category

34.3.1 Definition and elementary properties

For further information see [1][20].

definition *dagcat-Rel* :: $V \Rightarrow V$
where $dagcat\text{-}Rel\ \alpha = [cat\text{-}Rel\ \alpha, \dagger_{C.Rel}\ \alpha]_\circ$

Components.

lemma *dagcat-Rel-components*:
shows $dagcat\text{-}Rel\ \alpha(\!|DagCat|\!) = cat\text{-}Rel\ \alpha$
and $dagcat\text{-}Rel\ \alpha(\!|DagDag|\!) = \dagger_{C.Rel}\ \alpha$
⟨*proof*⟩

34.3.2 *Rel* is a dagger category

lemma (**in** $\mathcal{Z}$) *dagger-category-dagcat-Rel*: *dagger-category* α (*dagcat-Rel* α)
⟨*proof*⟩

34.4 Monoidal category

For background information see Chapter 2 in [4].

34.4.1 Background

named-theorems *mcat-field-simps*

named-theorems *mcat-cs-simps*
named-theorems *mcat-cs-intros*

definition *Mcat* :: *V* **where** [*mcat-field-simps*]: *Mcat* = *0*
definition *Mcf* :: *V* **where** [*mcat-field-simps*]: *Mcf* = $1_{\mathbb{N}}$
definition *Me* :: *V* **where** [*mcat-field-simps*]: *Me* = $2_{\mathbb{N}}$
definition *Mα* :: *V* **where** [*mcat-field-simps*]: *Mα* = $3_{\mathbb{N}}$
definition *Ml* :: *V* **where** [*mcat-field-simps*]: *Ml* = $4_{\mathbb{N}}$
definition *Mr* :: *V* **where** [*mcat-field-simps*]: *Mr* = $5_{\mathbb{N}}$

34.4.2 Definition and elementary properties

locale *monoidal-category* =
— See Definition 2.2.8 in [4].
$\mathcal{Z}\ \alpha$ +
vfsequence $\mathfrak{C}$ +
Mcat: *category* α ‹$\mathfrak{C}(\!|Mcat|\!)$› +
Mcf: *is-functor* α ‹$(\mathfrak{C}(\!|Mcat|\!)) \times_C (\mathfrak{C}(\!|Mcat|\!))$› ‹$\mathfrak{C}(\!|Mcat|\!)$› ‹$\mathfrak{C}(\!|Mcf|\!)$› +
Mα: *is-iso-ntcf*
α ‹$\mathfrak{C}(\!|Mcat|\!)\hat{}_{C3}$› ‹$\mathfrak{C}(\!|Mcat|\!)$› ‹*cf-blcomp* $(\mathfrak{C}(\!|Mcf|\!))$› ‹*cf-brcomp* $(\mathfrak{C}(\!|Mcf|\!))$› ‹$\mathfrak{C}(\!|M\alpha|\!)$› +

[20] https://ncatlab.org/nlab/show/Rel

Ml: *is-iso-ntcf*
α
$\langle \mathfrak{C}(\lfloor Mcat \rfloor) \rangle$
$\langle \mathfrak{C}(\lfloor Mcat \rfloor) \rangle$
$\langle \mathfrak{C}(\lfloor Mcf \rfloor)_{\mathfrak{C}(\lfloor Mcat \rfloor),\mathfrak{C}(\lfloor Mcat \rfloor)}(\mathfrak{C}(\lfloor Me \rfloor),-)_{CF} \rangle$
$\langle cf\text{-}id\ (\mathfrak{C}(\lfloor Mcat \rfloor)) \rangle$
$\langle \mathfrak{C}(\lfloor Ml \rfloor) \rangle$ +
Mr: *is-iso-ntcf*
α
$\langle \mathfrak{C}(\lfloor Mcat \rfloor) \rangle$
$\langle \mathfrak{C}(\lfloor Mcat \rfloor) \rangle$
$\langle \mathfrak{C}(\lfloor Mcf \rfloor)_{\mathfrak{C}(\lfloor Mcat \rfloor),\mathfrak{C}(\lfloor Mcat \rfloor)}(-,\mathfrak{C}(\lfloor Me \rfloor))_{CF} \rangle$
$\langle cf\text{-}id\ (\mathfrak{C}(\lfloor Mcat \rfloor)) \rangle$
$\langle \mathfrak{C}(\lfloor Mr \rfloor) \rangle$
for α $\mathfrak{C}$ +
assumes *mcat-length*[*mcat-cs-simps*]: *vcard* $\mathfrak{C} = 6_{\mathbb{N}}$
and *mcat-Me-is-obj*[*mcat-cs-intros*]: $\mathfrak{C}(\lfloor Me \rfloor) \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)$
and *mcat-pentagon*:
$[\![$
$a \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)$;
$b \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)$;
$c \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)$;
$d \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)$
$]\!] \Longrightarrow$
$(\mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor CId \rfloor)(\lfloor a \rfloor) \otimes_{HM.A\mathfrak{C}(\lfloor Mcf \rfloor)} \mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor b,\ c,\ d \rfloor_{\bullet})) \circ_{A\mathfrak{C}(\lfloor Mcat \rfloor)}$
$\mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor a,\ b \otimes_{HM.O\mathfrak{C}(\lfloor Mcf \rfloor)} c,\ d \rfloor_{\bullet}) \circ_{A\mathfrak{C}(\lfloor Mcat \rfloor)}$
$(\mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor a,\ b,\ c \rfloor_{\bullet}) \otimes_{HM.A\mathfrak{C}(\lfloor Mcf \rfloor)} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor CId \rfloor)(\lfloor d \rfloor)) =$
$\mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor a,\ b,\ c \otimes_{HM.O\mathfrak{C}(\lfloor Mcf \rfloor)} d \rfloor_{\bullet}) \circ_{A\mathfrak{C}(\lfloor Mcat \rfloor)}$
$\mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor a \otimes_{HM.O\mathfrak{C}(\lfloor Mcf \rfloor)} b,\ c,\ d \rfloor_{\bullet})$
and *mcat-triangle*[*mcat-cs-simps*]:
$[\![\ a \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor);\ b \in_{\circ} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor Obj \rfloor)\ ]\!] \Longrightarrow$
$(\mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor CId \rfloor)(\lfloor a \rfloor) \otimes_{HM.A\mathfrak{C}(\lfloor Mcf \rfloor)} \mathfrak{C}(\lfloor Ml \rfloor)(\lfloor NTMap \rfloor)(\lfloor b \rfloor)) \circ_{A\mathfrak{C}(\lfloor Mcat \rfloor)}$
$\mathfrak{C}(\lfloor M\alpha \rfloor)(\lfloor NTMap \rfloor)(\lfloor a,\ \mathfrak{C}(\lfloor Me \rfloor),\ b \rfloor_{\bullet}) =$
$(\mathfrak{C}(\lfloor Mr \rfloor)(\lfloor NTMap \rfloor)(\lfloor a \rfloor) \otimes_{HM.A\mathfrak{C}(\lfloor Mcf \rfloor)} \mathfrak{C}(\lfloor Mcat \rfloor)(\lfloor CId \rfloor)(\lfloor b \rfloor))$

lemmas [*mcat-cs-intros*] = *monoidal-category.mcat-Me-is-obj*
lemmas [*mcat-cs-simps*] = *monoidal-category.mcat-triangle*

Rules.

lemma (**in** *monoidal-category*) *monoidal-category-axioms'*[*mcat-cs-intros*]:
assumes $\alpha' = \alpha$
shows *monoidal-category* α' $\mathfrak{C}$
⟨*proof*⟩

mk-ide rf *monoidal-category-def*[*unfolded monoidal-category-axioms-def*]
|*intro monoidal-categoryI*|
|*dest monoidal-categoryD*[*dest*]|
|*elim monoidal-categoryE*[*elim*]|

Elementary properties.

lemma *mcat-eqI*:
assumes *monoidal-category* α $\mathfrak{A}$
and *monoidal-category* α $\mathfrak{B}$
and $\mathfrak{A}(\lfloor Mcat \rfloor) = \mathfrak{B}(\lfloor Mcat \rfloor)$
and $\mathfrak{A}(\lfloor Mcf \rfloor) = \mathfrak{B}(\lfloor Mcf \rfloor)$
and $\mathfrak{A}(\lfloor Me \rfloor) = \mathfrak{B}(\lfloor Me \rfloor)$

and $\mathfrak{A}(\!|M\alpha|\!) = \mathfrak{B}(\!|M\alpha|\!)$
and $\mathfrak{A}(\!|Ml|\!) = \mathfrak{B}(\!|Ml|\!)$
and $\mathfrak{A}(\!|Mr|\!) = \mathfrak{B}(\!|Mr|\!)$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

34.5 Components for $M\alpha$ for *Rel*

34.5.1 Definition and elementary properties

definition *Mα-Rel-arrow-lr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *Mα-Rel-arrow-lr A B C* =
[
$(\lambda ab\text{-}c \in_\circ (A \times_\circ B) \times_\circ C.\ \langle vfst\ (vfst\ ab\text{-}c),\ \langle vsnd\ (vfst\ ab\text{-}c),\ vsnd\ ab\text{-}c\rangle\rangle)$,
$(A \times_\circ B) \times_\circ C$,
$A \times_\circ (B \times_\circ C)$
]$_\circ$

definition *Mα-Rel-arrow-rl* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *Mα-Rel-arrow-rl A B C* =
[
$(\lambda a\text{-}bc \in_\circ A \times_\circ (B \times_\circ C).\ \langle\langle vfst\ a\text{-}bc,\ vfst\ (vsnd\ a\text{-}bc)\rangle,\ vsnd\ (vsnd\ a\text{-}bc)\rangle)$,
$A \times_\circ (B \times_\circ C)$,
$(A \times_\circ B) \times_\circ C$
]$_\circ$

Components.

lemma *Mα-Rel-arrow-lr-components*:
shows *Mα-Rel-arrow-lr A B C*(|*ArrVal*|) =
$(\lambda ab\text{-}c \in_\circ (A \times_\circ B) \times_\circ C.\ \langle vfst\ (vfst\ ab\text{-}c),\ \langle vsnd\ (vfst\ ab\text{-}c),\ vsnd\ ab\text{-}c\rangle\rangle)$
and [*cat-cs-simps*]: *Mα-Rel-arrow-lr A B C*(|*ArrDom*|) = $(A \times_\circ B) \times_\circ C$
and [*cat-cs-simps*]: *Mα-Rel-arrow-lr A B C*(|*ArrCod*|) = $A \times_\circ (B \times_\circ C)$
⟨*proof*⟩

lemma *Mα-Rel-arrow-rl-components*:
shows *Mα-Rel-arrow-rl A B C*(|*ArrVal*|) =
$(\lambda a\text{-}bc \in_\circ A \times_\circ (B \times_\circ C).\ \langle\langle vfst\ a\text{-}bc,\ vfst\ (vsnd\ a\text{-}bc)\rangle,\ vsnd\ (vsnd\ a\text{-}bc)\rangle)$
and [*cat-cs-simps*]: *Mα-Rel-arrow-rl A B C*(|*ArrDom*|) = $A \times_\circ (B \times_\circ C)$
and [*cat-cs-simps*]: *Mα-Rel-arrow-rl A B C*(|*ArrCod*|) = $(A \times_\circ B) \times_\circ C$
⟨*proof*⟩

34.5.2 Arrow value

mk-VLambda *Mα-Rel-arrow-lr-components*(*1*)
|*vsv Mα-Rel-arrow-lr-ArrVal-vsv*[*cat-cs-intros*]|
|*vdomain Mα-Rel-arrow-lr-ArrVal-vdomain*[*cat-cs-simps*]|
|*app Mα-Rel-arrow-lr-ArrVal-app′*|

lemma *Mα-Rel-arrow-lr-ArrVal-app*[*cat-cs-simps*]:
assumes *ab-c* = $\langle\langle a,\ b\rangle,\ c\rangle$ **and** *ab-c* $\in_\circ$ $(A \times_\circ B) \times_\circ C$
shows *Mα-Rel-arrow-lr A B C*(|*ArrVal*|)(|*ab-c*|) = $\langle a,\ \langle b,\ c\rangle\rangle$
⟨*proof*⟩

mk-VLambda *Mα-Rel-arrow-rl-components*(*1*)
|*vsv Mα-Rel-arrow-rl-ArrVal-vsv*[*cat-cs-intros*]|
|*vdomain Mα-Rel-arrow-rl-ArrVal-vdomain*[*cat-cs-simps*]|
|*app Mα-Rel-arrow-rl-ArrVal-app′*|

lemma *Mα-Rel-arrow-rl-ArrVal-app*[*cat-cs-simps*]:

assumes $a\text{-}bc = \langle a,\ \langle b,\ c\rangle\rangle$ **and** $a\text{-}bc \in_\circ A \times_\circ (B \times_\circ C)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C(\lfloor ArrVal\rfloor)(\lfloor a\text{-}bc\rfloor) = \langle\langle a,\ b\rangle,\ c\rangle$
⟨*proof*⟩

34.5.3 Components for $M\alpha$ for Rel are arrows

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Set\text{-}arr\text{-}Vset$:
assumes $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$ **and** $C \in_\circ Vset\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{cat\text{-}Set\ \alpha} A \times_\circ (B \times_\circ C)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Set\text{-}arr\text{-}Vset$:
assumes $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$ **and** $C \in_\circ Vset\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A \times_\circ (B \times_\circ C) \mapsto_{cat\text{-}Set\ \alpha} (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Set\text{-}arr$:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $B \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $C \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{cat\text{-}Set\ \alpha} A \times_\circ (B \times_\circ C)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Set\text{-}arr'[cat\text{-}rel\text{-}par\text{-}Set\text{-}cs\text{-}intros]$:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $B \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $C \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $A' = (A \times_\circ B) \times_\circ C$
and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas $[cat\text{-}rel\text{-}par\text{-}Set\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Set\text{-}arr'$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Set\text{-}arr$:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $B \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $C \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A \times_\circ (B \times_\circ C) \mapsto_{cat\text{-}Set\ \alpha} (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Set\text{-}arr'[cat\text{-}rel\text{-}par\text{-}Set\text{-}cs\text{-}intros]$:
assumes $A \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $B \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $C \in_\circ cat\text{-}Set\ \alpha(\lfloor Obj\rfloor)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' = cat\text{-}Set\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas $[cat\text{-}rel\text{-}par\text{-}Set\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Set\text{-}arr'$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Par\text{-}arr$:
assumes $A \in_\circ cat\text{-}Par\ \alpha(\lfloor Obj\rfloor)$
and $B \in_\circ cat\text{-}Par\ \alpha(\lfloor Obj\rfloor)$
and $C \in_\circ cat\text{-}Par\ \alpha(\lfloor Obj\rfloor)$

shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{cat\text{-}Par\ \alpha} A \times_\circ (B \times_\circ C)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Par\text{-}arr'[cat\text{-}rel\text{-}Par\text{-}set\text{-}cs\text{-}intros]$:
assumes $A \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $A' = (A \times_\circ B) \times_\circ C$
and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = cat\text{-}Par\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas $[cat\text{-}rel\text{-}Par\text{-}set\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Par\text{-}arr'$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Par\text{-}arr$:
assumes $A \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A \times_\circ (B \times_\circ C) \mapsto_{cat\text{-}Par\ \alpha} (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Par\text{-}arr'[cat\text{-}rel\text{-}Par\text{-}set\text{-}cs\text{-}intros]$:
assumes $A \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Par\ \alpha(\!|Obj|\!)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' = cat\text{-}Par\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas $[cat\text{-}rel\text{-}Par\text{-}set\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Par\text{-}arr'$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Rel\text{-}arr$:
assumes $A \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{cat\text{-}Rel\ \alpha} A \times_\circ (B \times_\circ C)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Rel\text{-}arr'[cat\text{-}Rel\text{-}par\text{-}set\text{-}cs\text{-}intros]$:
assumes $A \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $A' = (A \times_\circ B) \times_\circ C$
and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = cat\text{-}Rel\ \alpha$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas $[cat\text{-}Rel\text{-}par\text{-}set\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}lr\text{-}is\text{-}cat\text{-}Rel\text{-}arr'$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Rel\text{-}arr$:
assumes $A \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $B \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
and $C \in_\circ cat\text{-}Rel\ \alpha(\!|Obj|\!)$
shows $M\alpha\text{-}Rel\text{-}arrow\text{-}rl\ A\ B\ C : A \times_\circ (B \times_\circ C) \mapsto_{cat\text{-}Rel\ \alpha} (A \times_\circ B) \times_\circ C$

⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *Mα-Rel-arrow-rl-is-cat-Rel-arr'*[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $C \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' =$ *cat-Rel* α
shows *Mα-Rel-arrow-rl* $A\ B\ C : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] = $\mathcal{Z}$.*Mα-Rel-arrow-rl-is-cat-Rel-arr'*

34.5.4 Further properties

lemma (in $\mathcal{Z}$**)** *Mα-Rel-arrow-rl-Mα-Rel-arrow-lr*[*cat-cs-simps*]:
assumes $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α **and** $C \in_\circ$ *Vset* α
shows
Mα-Rel-arrow-rl $A\ B\ C \circ_{A\,cat\text{-}Set\ \alpha}$ *Mα-Rel-arrow-lr* $A\ B\ C =$
cat-Set $\alpha(\!|CId|\!)(\!|(A \times_\circ B) \times_\circ C|\!)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *Mα-Rel-arrow-rl-Mα-Rel-arrow-lr'*[*cat-cs-simps*]:
assumes $A \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and $C \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
shows
Mα-Rel-arrow-rl $A\ B\ C \circ_{A\,cat\text{-}Set\ \alpha}$ *Mα-Rel-arrow-lr* $A\ B\ C =$
cat-Set $\alpha(\!|CId|\!)(\!|(A \times_\circ B) \times_\circ C|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = $\mathcal{Z}$.*Mα-Rel-arrow-rl-Mα-Rel-arrow-lr'*

lemma (in $\mathcal{Z}$**)** *Mα-Rel-arrow-lr-Mα-Rel-arrow-rl*[*cat-cs-simps*]:
assumes $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α **and** $C \in_\circ$ *Vset* α
shows
Mα-Rel-arrow-lr $A\ B\ C \circ_{A\,cat\text{-}Set\ \alpha}$ *Mα-Rel-arrow-rl* $A\ B\ C =$
cat-Set $\alpha(\!|CId|\!)(\!|A \times_\circ (B \times_\circ C)|\!)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *Mα-Rel-arrow-lr-Mα-Rel-arrow-rl'*[*cat-cs-simps*]:
assumes $A \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and $B \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
and $C \in_\circ$ *cat-Set* $\alpha(\!|Obj|\!)$
shows
Mα-Rel-arrow-lr $A\ B\ C \circ_{A\,cat\text{-}Set\ \alpha}$ *Mα-Rel-arrow-rl* $A\ B\ C =$
cat-Set $\alpha(\!|CId|\!)(\!|A \times_\circ (B \times_\circ C)|\!)$
⟨*proof*⟩

lemmas [*cat-cs-simps*] = $\mathcal{Z}$.*Mα-Rel-arrow-lr-Mα-Rel-arrow-rl'*

34.5.5 Components for $M\alpha$ for *Rel* are isomorphisms

lemma (in $\mathcal{Z}$**)**
assumes $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α **and** $C \in_\circ$ *Vset* α
shows *Mα-Rel-arrow-lr-is-cat-Set-iso-arr-Vset*:
Mα-Rel-arrow-lr $A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{iso\,cat\text{-}Set\ \alpha} A \times_\circ (B \times_\circ C)$

and *Mα-Rel-arrow-rl-is-cat-Set-iso-arr-Vset*:
Mα-Rel-arrow-rl A B C : $A \times_\circ (B \times_\circ C) \mapsto_{iso} \textit{cat-Set}\ \alpha\ (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)**
assumes $A \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
shows *Mα-Rel-arrow-lr-is-cat-Set-iso-arr*:
Mα-Rel-arrow-lr A B C : $(A \times_\circ B) \times_\circ C \mapsto_{iso} \textit{cat-Set}\ \alpha\ A \times_\circ (B \times_\circ C)$
and *Mα-Rel-arrow-rl-is-cat-Set-iso-arr*:
Mα-Rel-arrow-rl A B C : $A \times_\circ (B \times_\circ C) \mapsto_{iso} \textit{cat-Set}\ \alpha\ (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)**
Mα-Rel-arrow-lr-is-cat-Set-iso-arr'[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = (A \times_\circ B) \times_\circ C$
and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows *Mα-Rel-arrow-lr A B C* : $A' \mapsto_{iso \mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] =
$\mathcal{Z}$.*Mα-Rel-arrow-lr-is-cat-Set-iso-arr'*

lemma (in $\mathcal{Z}$**)**
Mα-Rel-arrow-rl-is-cat-Set-iso-arr'[*cat-rel-par-Set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' = \textit{cat-Set}\ \alpha$
shows *Mα-Rel-arrow-rl A B C* : $A' \mapsto_{iso \mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*cat-rel-par-Set-cs-intros*] =
$\mathcal{Z}$.*Mα-Rel-arrow-rl-is-cat-Set-iso-arr'*

lemma (in $\mathcal{Z}$**)**
assumes $A \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
shows *Mα-Rel-arrow-lr-is-cat-Par-iso-arr*:
Mα-Rel-arrow-lr A B C : $(A \times_\circ B) \times_\circ C \mapsto_{iso} \textit{cat-Par}\ \alpha\ A \times_\circ (B \times_\circ C)$
and *Mα-Rel-arrow-rl-is-cat-Par-iso-arr*:
Mα-Rel-arrow-rl A B C : $A \times_\circ (B \times_\circ C) \mapsto_{iso} \textit{cat-Par}\ \alpha\ (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)**
Mα-Rel-arrow-lr-is-cat-Par-iso-arr'[*cat-rel-Par-set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = (A \times_\circ B) \times_\circ C$

and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = \textit{cat-Par}\ \alpha$
shows $M\alpha\textit{-Rel-arrow-lr}\ A\ B\ C : A' \mapsto_{iso_{\mathfrak{C}'}} B'$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] =
$\mathcal{Z}.M\alpha\textit{-Rel-arrow-lr-is-cat-Par-iso-arr}'$

lemma (in $\mathcal{Z}$**)**
$M\alpha\textit{-Rel-arrow-rl-is-cat-Par-iso-arr}'$[*cat-rel-Par-set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' = \textit{cat-Par}\ \alpha$
shows $M\alpha\textit{-Rel-arrow-rl}\ A\ B\ C : A' \mapsto_{iso_{\mathfrak{C}'}} B'$
⟨*proof*⟩

lemmas [*cat-rel-Par-set-cs-intros*] =
$\mathcal{Z}.M\alpha\textit{-Rel-arrow-rl-is-cat-Par-iso-arr}'$

lemma (in $\mathcal{Z}$**)**
assumes $A \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
shows $M\alpha\textit{-Rel-arrow-lr-is-cat-Rel-iso-arr}$:
$M\alpha\textit{-Rel-arrow-lr}\ A\ B\ C : (A \times_\circ B) \times_\circ C \mapsto_{iso_{\textit{cat-Rel}\ \alpha}} A \times_\circ (B \times_\circ C)$
and $M\alpha\textit{-Rel-arrow-rl-is-cat-Rel-iso-arr}$:
$M\alpha\textit{-Rel-arrow-rl}\ A\ B\ C : A \times_\circ (B \times_\circ C) \mapsto_{iso_{\textit{cat-Rel}\ \alpha}} (A \times_\circ B) \times_\circ C$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)**
$M\alpha\textit{-Rel-arrow-lr-is-cat-Rel-iso-arr}'$[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = (A \times_\circ B) \times_\circ C$
and $B' = A \times_\circ (B \times_\circ C)$
and $\mathfrak{C}' = \textit{cat-Rel}\ \alpha$
shows $M\alpha\textit{-Rel-arrow-lr}\ A\ B\ C : A' \mapsto_{iso_{\mathfrak{C}'}} B'$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] =
$\mathcal{Z}.M\alpha\textit{-Rel-arrow-lr-is-cat-Rel-iso-arr}'$

lemma (in $\mathcal{Z}$**)**
$M\alpha\textit{-Rel-arrow-rl-is-cat-Rel-iso-arr}'$[*cat-Rel-par-set-cs-intros*]:
assumes $A \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $C \in_\circ \textit{cat-Rel}\ \alpha(\!|\textit{Obj}|\!)$
and $A' = A \times_\circ (B \times_\circ C)$
and $B' = (A \times_\circ B) \times_\circ C$
and $\mathfrak{C}' = \textit{cat-Rel}\ \alpha$
shows $M\alpha\textit{-Rel-arrow-rl}\ A\ B\ C : A' \mapsto_{iso_{\mathfrak{C}'}} B'$
⟨*proof*⟩

lemmas [*cat-Rel-par-set-cs-intros*] =

$\mathcal{Z}.M\alpha\text{-}Rel\text{-}arrow\text{-}rl\text{-}is\text{-}cat\text{-}Rel\text{-}iso\text{-}arr'$

34.6 $M\alpha$ for Rel

34.6.1 Definition and elementary properties

definition $M\alpha\text{-}Rel :: V \Rightarrow V$
where $M\alpha\text{-}Rel\ \mathfrak{C} =$
$[$
$(\lambda abc \in_\circ (\mathfrak{C}\hat{}_{C3})(\lfloor Obj \rfloor).\ M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ (abc(\lfloor 0 \rfloor))\ (abc(\lfloor 1_{\mathbb{N}} \rfloor))\ (abc(\lfloor 2_{\mathbb{N}} \rfloor)))$,
$cf\text{-}blcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{C})$,
$cf\text{-}brcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{C})$,
$\mathfrak{C}\hat{}_{C3}$,
$\mathfrak{C}$
$]_\circ$

Components.

lemma $M\alpha\text{-}Rel\text{-}components$:
shows $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTMap \rfloor) =$
$(\lambda abc \in_\circ (\mathfrak{C}\hat{}_{C3})(\lfloor Obj \rfloor).\ M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ (abc(\lfloor 0 \rfloor))\ (abc(\lfloor 1_{\mathbb{N}} \rfloor))\ (abc(\lfloor 2_{\mathbb{N}} \rfloor)))$
and $[cat\text{-}cs\text{-}simps]$: $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTDom \rfloor) = cf\text{-}blcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{C})$
and $[cat\text{-}cs\text{-}simps]$: $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTCod \rfloor) = cf\text{-}brcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ \mathfrak{C})$
and $[cat\text{-}cs\text{-}simps]$: $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTDGDom \rfloor) = \mathfrak{C}\hat{}_{C3}$
and $[cat\text{-}cs\text{-}simps]$: $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTDGCod \rfloor) = \mathfrak{C}$
$\langle proof \rangle$

34.6.2 Natural transformation map

mk-VLambda $M\alpha\text{-}Rel\text{-}components(1)$
$|vsv\ M\alpha\text{-}Rel\text{-}NTMap\text{-}vsv[cat\text{-}cs\text{-}intros]|$
$|vdomain\ M\alpha\text{-}Rel\text{-}NTMap\text{-}vdomain[cat\text{-}cs\text{-}simps]|$
$|app\ M\alpha\text{-}Rel\text{-}NTMap\text{-}app'|$

lemma $M\alpha\text{-}Rel\text{-}NTMap\text{-}app[cat\text{-}cs\text{-}simps]$:
assumes $ABC = [A, B, C]_\circ$ **and** $ABC \in_\circ (\mathfrak{C}\hat{}_{C3})(\lfloor Obj \rfloor)$
shows $M\alpha\text{-}Rel\ \mathfrak{C}(\lfloor NTMap \rfloor)(\lfloor ABC \rfloor) = M\alpha\text{-}Rel\text{-}arrow\text{-}lr\ A\ B\ C$
$\langle proof \rangle$

34.6.3 $M\alpha$ for Rel is a natural isomorphism

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}is\text{-}iso\text{-}ntcf$:
$M\alpha\text{-}Rel\ (cat\text{-}Rel\ \alpha) :$
$cf\text{-}blcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha)) \mapsto_{CF.iso}$
$cf\text{-}brcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha)) :$
$cat\text{-}Rel\ \alpha\hat{}_{C3} \mapsto\mapsto_{C\alpha} cat\text{-}Rel\ \alpha$
$\langle proof \rangle$

lemma (in $\mathcal{Z}$**)** $M\alpha\text{-}Rel\text{-}is\text{-}iso\text{-}ntcf'[cat\text{-}cs\text{-}intros]$:
assumes $\mathfrak{F}' = cf\text{-}blcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha))$
and $\mathfrak{G}' = cf\text{-}brcomp\ (cf\text{-}prod\text{-}2\text{-}Rel\ (cat\text{-}Rel\ \alpha))$
and $\mathfrak{A}' = cat\text{-}Rel\ \alpha\hat{}_{C3}$
and $\mathfrak{B}' = cat\text{-}Rel\ \alpha$
and $\alpha' = \alpha$
shows $M\alpha\text{-}Rel\ (cat\text{-}Rel\ \alpha) : \mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{C\alpha'} \mathfrak{B}'$
$\langle proof \rangle$

lemmas $[cat\text{-}cs\text{-}intros] = \mathcal{Z}.M\alpha\text{-}Rel\text{-}is\text{-}iso\text{-}ntcf'$

34.7 *Ml* and *Mr* for *Rel*

34.7.1 Definition and elementary properties

definition *Ml-Rel* :: $V \Rightarrow V \Rightarrow V$
  where *Ml-Rel* $\mathfrak{C}$ a =
    [
      $(\lambda B \in_\circ \mathfrak{C}(\!|Obj|\!).$ *vsnd-arrow* (*set* $\{a\}$) $B)$,
      cf-prod-2-Rel $\mathfrak{C}_{\mathfrak{C},\mathfrak{C}}$(*set* $\{a\}$,−)$_{CF}$,
      cf-id $\mathfrak{C}$,
      $\mathfrak{C}$,
      $\mathfrak{C}$
    ]$_\circ$

definition *Mr-Rel* :: $V \Rightarrow V \Rightarrow V$
  where *Mr-Rel* $\mathfrak{C}$ b =
    [
      $(\lambda A \in_\circ \mathfrak{C}(\!|Obj|\!).$ *vfst-arrow* A (*set* $\{b\}$)),
      cf-prod-2-Rel $\mathfrak{C}_{\mathfrak{C},\mathfrak{C}}$(−,*set* $\{b\}$)$_{CF}$,
      cf-id $\mathfrak{C}$,
      $\mathfrak{C}$,
      $\mathfrak{C}$
    ]$_\circ$

Components.

lemma *Ml-Rel-components*:
  shows *Ml-Rel* $\mathfrak{C}$ $a(\!|NTMap|\!)$ = $(\lambda B \in_\circ \mathfrak{C}(\!|Obj|\!).$ *vsnd-arrow* (*set* $\{a\}$) $B)$
    and [*cat-cs-simps*]: *Ml-Rel* $\mathfrak{C}$ $a(\!|NTDom|\!)$ = *cf-prod-2-Rel* $\mathfrak{C}_{\mathfrak{C},\mathfrak{C}}$(*set* $\{a\}$,−)$_{CF}$
    and [*cat-cs-simps*]: *Ml-Rel* $\mathfrak{C}$ $a(\!|NTCod|\!)$ = *cf-id* $\mathfrak{C}$
    and [*cat-cs-simps*]: *Ml-Rel* $\mathfrak{C}$ $a(\!|NTDGDom|\!)$ = $\mathfrak{C}$
    and [*cat-cs-simps*]: *Ml-Rel* $\mathfrak{C}$ $a(\!|NTDGCod|\!)$ = $\mathfrak{C}$
  ⟨*proof*⟩

lemma *Mr-Rel-components*:
  shows *Mr-Rel* $\mathfrak{C}$ $b(\!|NTMap|\!)$ = $(\lambda A \in_\circ \mathfrak{C}(\!|Obj|\!).$ *vfst-arrow* A (*set* $\{b\}$))
    and [*cat-cs-simps*]: *Mr-Rel* $\mathfrak{C}$ $b(\!|NTDom|\!)$ = *cf-prod-2-Rel* $\mathfrak{C}_{\mathfrak{C},\mathfrak{C}}$(−,*set* $\{b\}$)$_{CF}$
    and [*cat-cs-simps*]: *Mr-Rel* $\mathfrak{C}$ $b(\!|NTCod|\!)$ = *cf-id* $\mathfrak{C}$
    and [*cat-cs-simps*]: *Mr-Rel* $\mathfrak{C}$ $b(\!|NTDGDom|\!)$ = $\mathfrak{C}$
    and [*cat-cs-simps*]: *Mr-Rel* $\mathfrak{C}$ $b(\!|NTDGCod|\!)$ = $\mathfrak{C}$
  ⟨*proof*⟩

34.7.2 Natural transformation map

mk-VLambda *Ml-Rel-components*(*1*)
  |*vsv Ml-Rel-components-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain Ml-Rel-components-NTMap-vdomain*[*cat-cs-simps*]|
  |*app Ml-Rel-components-NTMap-app*[*cat-cs-simps*]|

mk-VLambda *Mr-Rel-components*(*1*)
  |*vsv Mr-Rel-components-NTMap-vsv*[*cat-cs-intros*]|
  |*vdomain Mr-Rel-components-NTMap-vdomain*[*cat-cs-simps*]|
  |*app Mr-Rel-components-NTMap-app*[*cat-cs-simps*]|

34.7.3 *Ml* and *Mr* for *Rel* are natural isomorphisms

lemma (**in** $\mathcal{Z}$) *Ml-Rel-is-iso-ntcf*:
  assumes $a \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
  shows *Ml-Rel* (*cat-Rel* α) a:
    cf-prod-2-Rel (*cat-Rel* α)$_{\text{cat-Rel } \alpha,\text{cat-Rel } \alpha}$(*set* $\{a\}$,−)$_{CF} \mapsto_{CF.iso}$

cf-id (*cat-Rel* α) :
cat-Rel α $\mapsto\!\!\mapsto_{C\alpha}$ *cat-Rel* α
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *Ml-Rel-is-iso-ntcf'*[*cat-cs-intros*]:
assumes $a \in_\circ$ *cat-Rel* α(|*Obj*|)
and $\mathfrak{F}'$ = *cf-prod-2-Rel* (*cat-Rel* α)$_{\text{cat-Rel }\alpha,\text{cat-Rel }\alpha}$(*set* {$a$},−)$_{CF}$
and $\mathfrak{G}'$ = *cf-id* (*cat-Rel* α)
and $\mathfrak{A}'$ = *cat-Rel* α
and $\mathfrak{B}'$ = *cat-Rel* α
and α' = α
shows *Ml-Rel* (*cat-Rel* α) a : $\mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\!\!\mapsto_{C_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

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

lemma (**in** $\mathcal{Z}$) *Mr-Rel-is-iso-ntcf*:
assumes $b \in_\circ$ *cat-Rel* α(|*Obj*|)
shows *Mr-Rel* (*cat-Rel* α) b :
cf-prod-2-Rel (*cat-Rel* α)$_{\text{cat-Rel }\alpha,\text{cat-Rel }\alpha}$(−,*set* {$b$})$_{CF}$ $\mapsto_{CF.iso}$
cf-id (*cat-Rel* α) :
cat-Rel α $\mapsto\!\!\mapsto_{C\alpha}$ *cat-Rel* α
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *Mr-Rel-is-iso-ntcf'*[*cat-cs-intros*]:
assumes $b \in_\circ$ *cat-Rel* α(|*Obj*|)
and $\mathfrak{F}'$ = *cf-prod-2-Rel* (*cat-Rel* α)$_{\text{cat-Rel }\alpha,\text{cat-Rel }\alpha}$(−,*set* {$b$})$_{CF}$
and $\mathfrak{G}'$ = *cf-id* (*cat-Rel* α)
and $\mathfrak{A}'$ = *cat-Rel* α
and $\mathfrak{B}'$ = *cat-Rel* α
and α' = α
shows *Mr-Rel* (*cat-Rel* α) b : $\mathfrak{F}' \mapsto_{CF.iso} \mathfrak{G}' : \mathfrak{A}' \mapsto\!\!\mapsto_{C_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

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

34.8 *Rel* as a monoidal category

34.8.1 Definition and elementary properties

For further information see [2][21].

definition *mcat-Rel* :: $V \Rightarrow V \Rightarrow V$
where *mcat-Rel* α a =
[
cat-Rel α,
cf-prod-2-Rel (*cat-Rel* α),
set {a},
Mα-Rel (*cat-Rel* α),
Ml-Rel (*cat-Rel* α) a,
Mr-Rel (*cat-Rel* α) a
]$_\circ$

Components.

lemma *mcat-Rel-components*:
shows *mcat-Rel* α a(|*Mcat*|) = *cat-Rel* α
and *mcat-Rel* α a(|*Mcf*|) = *cf-prod-2-Rel* (*cat-Rel* α)

[21] https://en.wikipedia.org/wiki/Category_of_relations

and *mcat-Rel* α $a(\!|Me|\!)$ = *set* $\{a\}$
and *mcat-Rel* α $a(\!|M\alpha|\!)$ = $M\alpha$-*Rel* (*cat-Rel* α)
and *mcat-Rel* α $a(\!|Ml|\!)$ = *Ml-Rel* (*cat-Rel* α) a
and *mcat-Rel* α $a(\!|Mr|\!)$ = *Mr-Rel* (*cat-Rel* α) a
⟨*proof*⟩

34.8.2 *Rel* is a monoidal category

lemma (in $\mathcal{Z}$**)** *monoidal-category-mcat-Rel*:
assumes $a \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
shows *monoidal-category* α (*mcat-Rel* α a)
⟨*proof*⟩

34.9 Dagger monoidal categories

34.9.1 Background

See [13] for further information.

named-theorems *dmcat-field-simps*

named-theorems *dmcat-cs-simps*
named-theorems *dmcat-cs-intros*

definition *DMcat* :: V **where** [*dmcat-field-simps*]: *DMcat* = *0*
definition *DMdag* :: V **where** [*dmcat-field-simps*]: *DMdag* = $1_{\mathbb{N}}$
definition *DMcf* :: V **where** [*dmcat-field-simps*]: *DMcf* = $2_{\mathbb{N}}$
definition *DMe* :: V **where** [*dmcat-field-simps*]: *DMe* = $3_{\mathbb{N}}$
definition *DM*α :: V **where** [*dmcat-field-simps*]: *DM*α = $4_{\mathbb{N}}$
definition *DMl* :: V **where** [*dmcat-field-simps*]: *DMl* = $5_{\mathbb{N}}$
definition *DMr* :: V **where** [*dmcat-field-simps*]: *DMr* = $6_{\mathbb{N}}$

abbreviation *DMDag-app* :: $V \Rightarrow V$ ($\langle\dagger_{MC}\rangle$)
where $\dagger_{MC}$ $\mathfrak{C}$ ≡ $\mathfrak{C}(\!|DMdag|\!)$

34.9.2 Slicing

Dagger category.

definition *dmcat-dagcat* :: $V \Rightarrow V$
where *dmcat-dagcat* $\mathfrak{C}$ = $[\mathfrak{C}(\!|DMcat|\!), \mathfrak{C}(\!|DMdag|\!)]_\circ$

lemma *dmcat-dagcat-components*[*slicing-simps*]:
shows *dmcat-dagcat* $\mathfrak{C}(\!|DagCat|\!)$ = $\mathfrak{C}(\!|DMcat|\!)$
and *dmcat-dagcat* $\mathfrak{C}(\!|DagDag|\!)$ = $\mathfrak{C}(\!|DMdag|\!)$
⟨*proof*⟩

Monoidal category.

definition *dmcat-mcat* :: $V \Rightarrow V$
where *dmcat-mcat* $\mathfrak{C}$ = $[\mathfrak{C}(\!|DMcat|\!), \mathfrak{C}(\!|DMcf|\!), \mathfrak{C}(\!|DMe|\!), \mathfrak{C}(\!|DM\alpha|\!), \mathfrak{C}(\!|DMl|\!), \mathfrak{C}(\!|DMr|\!)]_\circ$

lemma *dmcat-mcat-components*[*slicing-simps*]:
shows *dmcat-mcat* $\mathfrak{C}(\!|Mcat|\!)$ = $\mathfrak{C}(\!|DMcat|\!)$
and *dmcat-mcat* $\mathfrak{C}(\!|Mcf|\!)$ = $\mathfrak{C}(\!|DMcf|\!)$
and *dmcat-mcat* $\mathfrak{C}(\!|Me|\!)$ = $\mathfrak{C}(\!|DMe|\!)$
and *dmcat-mcat* $\mathfrak{C}(\!|M\alpha|\!)$ = $\mathfrak{C}(\!|DM\alpha|\!)$
and *dmcat-mcat* $\mathfrak{C}(\!|Ml|\!)$ = $\mathfrak{C}(\!|DMl|\!)$
and *dmcat-mcat* $\mathfrak{C}(\!|Mr|\!)$ = $\mathfrak{C}(\!|DMr|\!)$
⟨*proof*⟩

34.9.3 Definition and elementary properties

locale *dagger-monoidal-category* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ **for** α $\mathfrak{C}$ +
  assumes *dmcat-length*[*dmcat-cs-simps*]: *vcard* $\mathfrak{C}$ = $\gamma_{\mathbb{N}}$
    and *dmcat-dagger-category*: *dagger-category* α (*dmcat-dagcat* $\mathfrak{C}$)
    and *dmcat-monoidal-category*: *monoidal-category* α (*dmcat-mcat* $\mathfrak{C}$)
    and *dmcat-compatibility*:
      $[\![\ g : c \mapsto_{\mathfrak{C}(\!|DMcat|\!)} d;\ f : a \mapsto_{\mathfrak{C}(\!|DMcat|\!)} b\ ]\!] \Longrightarrow$
        $\dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|g \otimes_{HM.A\mathfrak{C}(\!|DMcf|\!)} f|\!) =$
          $\dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|g|\!) \otimes_{HM.A\mathfrak{C}(\!|DMcf|\!)} \dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|f|\!)$
    and *dmcat-Mα-unital*: $A \in_{\circ} (\mathfrak{C}(\!|DMcat|\!)\hat{}_{C3})(\!|Obj|\!) \Longrightarrow$
      $\dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|\mathfrak{C}(\!|DM\alpha|\!)(\!|NTMap|\!)(\!|A|\!)|\!) = (\mathfrak{C}(\!|DM\alpha|\!)(\!|NTMap|\!)(\!|A|\!))^{-1}{}_{C}\mathfrak{C}(\!|DMcat|\!)$
    and *dmcat-Ml-unital*: $a \in_{\circ} \mathfrak{C}(\!|DMcat|\!)(\!|Obj|\!) \Longrightarrow$
      $\dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|\mathfrak{C}(\!|DMl|\!)(\!|NTMap|\!)(\!|a|\!)|\!) = (\mathfrak{C}(\!|DMl|\!)(\!|NTMap|\!)(\!|a|\!))^{-1}{}_{C}\mathfrak{C}(\!|DMcat|\!)$
    and *dmcat-Mr-unital*: $a \in_{\circ} \mathfrak{C}(\!|DMcat|\!)(\!|Obj|\!) \Longrightarrow$
      $\dagger_{MC}\ \mathfrak{C}(\!|ArrMap|\!)(\!|\mathfrak{C}(\!|DMr|\!)(\!|NTMap|\!)(\!|a|\!)|\!) = (\mathfrak{C}(\!|DMr|\!)(\!|NTMap|\!)(\!|a|\!))^{-1}{}_{C}\mathfrak{C}(\!|DMcat|\!)$

Rules.

lemma (**in** *dagger-monoidal-category*)
  dagger-monoidal-category-axioms'[*dmcat-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *dagger-monoidal-category* α' $\mathfrak{C}$
  ⟨*proof*⟩

mk-ide rf
  dagger-monoidal-category-def[*unfolded dagger-monoidal-category-axioms-def*]
  |*intro dagger-monoidal-categoryI*[*intro*]|
  |*dest dagger-monoidal-categoryD*[*dest*]|
  |*elim dagger-monoidal-categoryE*[*elim*]|

Elementary properties.

lemma *dmcat-eqI*:
  assumes *dagger-monoidal-category* α $\mathfrak{A}$
    and *dagger-monoidal-category* α $\mathfrak{B}$
    and $\mathfrak{A}(\!|DMcat|\!) = \mathfrak{B}(\!|DMcat|\!)$
    and $\mathfrak{A}(\!|DMdag|\!) = \mathfrak{B}(\!|DMdag|\!)$
    and $\mathfrak{A}(\!|DMcf|\!) = \mathfrak{B}(\!|DMcf|\!)$
    and $\mathfrak{A}(\!|DMe|\!) = \mathfrak{B}(\!|DMe|\!)$
    and $\mathfrak{A}(\!|DM\alpha|\!) = \mathfrak{B}(\!|DM\alpha|\!)$
    and $\mathfrak{A}(\!|DMl|\!) = \mathfrak{B}(\!|DMl|\!)$
    and $\mathfrak{A}(\!|DMr|\!) = \mathfrak{B}(\!|DMr|\!)$
  shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

Slicing.

context *dagger-monoidal-category*
begin

interpretation *dagcat*: *dagger-category* α ‹*dmcat-dagcat* $\mathfrak{C}$›
  ⟨*proof*⟩

sublocale *DMCat*: *category* α ‹$\mathfrak{C}(\!|DMcat|\!)$›
  ⟨*proof*⟩

sublocale *DMDag*: *is-functor* α ‹*op-cat* ($\mathfrak{C}(\!|DMcat|\!)$)› ‹$\mathfrak{C}(\!|DMcat|\!)$› ‹$\dagger_{MC}$ $\mathfrak{C}$›
  ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
dmcat-Dom-vdomain[*dmcat-cs-simps*] = *dagcat.dagcat-ObjMap-identity*
and *dmcat-DagCat-idem*[*dmcat-cs-simps*] = *dagcat.dagcat-DagCat-idem*
and *dmcat-is-functor'*[*dmcat-cs-intros*] = *dagcat.dagcat-is-functor'*

end

lemmas [*dmcat-cs-simps*] =
dagger-monoidal-category.dmcat-Dom-vdomain
dagger-monoidal-category.dmcat-DagCat-idem

lemmas [*dmcat-cs-intros*] = *dagger-monoidal-category.dmcat-is-functor'*

context *dagger-monoidal-category*
begin

interpretation *mcat*: *monoidal-category* α ‹*dmcat-mcat* $\mathfrak{C}$›
⟨*proof*⟩

sublocale *DMcf*: *is-functor* α ‹$\mathfrak{C}(\lfloor DMcat \rfloor) \times_C \mathfrak{C}(\lfloor DMcat \rfloor)$› ‹$\mathfrak{C}(\lfloor DMcat \rfloor)$› ‹$\mathfrak{C}(\lfloor DMcf \rfloor)$›
⟨*proof*⟩

sublocale *DM*α: *is-iso-ntcf*
α ‹$\mathfrak{C}(\lfloor DMcat \rfloor)^{\frown}{}_{C3}$› ‹$\mathfrak{C}(\lfloor DMcat \rfloor)$› ‹*cf-blcomp* ($\mathfrak{C}(\lfloor DMcf \rfloor)$)› ‹*cf-brcomp* ($\mathfrak{C}(\lfloor DMcf \rfloor)$)› ‹$\mathfrak{C}(\lfloor DM\alpha \rfloor)$›
⟨*proof*⟩

sublocale *DMl*: *is-iso-ntcf*
α
‹$\mathfrak{C}(\lfloor DMcat \rfloor)$›
‹$\mathfrak{C}(\lfloor DMcat \rfloor)$›
‹$\mathfrak{C}(\lfloor DMcf \rfloor)_{\mathfrak{C}(\lfloor DMcat \rfloor),\mathfrak{C}(\lfloor DMcat \rfloor)}(\mathfrak{C}(\lfloor DMe \rfloor),-)_{CF}$›
‹*cf-id* ($\mathfrak{C}(\lfloor DMcat \rfloor)$)›
‹$\mathfrak{C}(\lfloor DMl \rfloor)$›
⟨*proof*⟩

sublocale *DMr*: *is-iso-ntcf*
α
‹$\mathfrak{C}(\lfloor DMcat \rfloor)$›
‹$\mathfrak{C}(\lfloor DMcat \rfloor)$›
‹$\mathfrak{C}(\lfloor DMcf \rfloor)_{\mathfrak{C}(\lfloor DMcat \rfloor),\mathfrak{C}(\lfloor DMcat \rfloor)}(-,\mathfrak{C}(\lfloor DMe \rfloor))_{CF}$›
‹*cf-id* ($\mathfrak{C}(\lfloor DMcat \rfloor)$)›
‹$\mathfrak{C}(\lfloor DMr \rfloor)$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
dmcat-Me-is-obj[*dmcat-cs-intros*] = *mcat.mcat-Me-is-obj*
and *dmcat-pentagon* = *mcat.mcat-pentagon*
and *dmcat-triangle*[*dmcat-cs-simps*] = *mcat.mcat-triangle*

end

lemmas [*dmcat-cs-intros*] = *dagger-monoidal-category.dmcat-Me-is-obj*
lemmas [*dmcat-cs-simps*] = *dagger-monoidal-category.dmcat-triangle*

34.10 *Rel* as a dagger monoidal category

34.10.1 Definition and elementary properties

definition *dmcat-Rel* :: $V \Rightarrow V \Rightarrow V$
  where *dmcat-Rel* α a =
    [
      cat-Rel α,
      $\dagger_{C.Rel}$ α,
      cf-prod-2-Rel (*cat-Rel* α),
      set $\{a\}$,
      Mα-Rel (*cat-Rel* α),
      Ml-Rel (*cat-Rel* α) a,
      Mr-Rel (*cat-Rel* α) a
    ]$_\circ$

Components.

lemma *dmcat-Rel-components*:
  shows *dmcat-Rel* α $a(\!|DMcat|\!)$ = *cat-Rel* α
    and *dmcat-Rel* α $a(\!|DMdag|\!)$ = $\dagger_{C.Rel}$ α
    and *dmcat-Rel* α $a(\!|DMcf|\!)$ = *cf-prod-2-Rel* (*cat-Rel* α)
    and *dmcat-Rel* α $a(\!|DMe|\!)$ = *set* $\{a\}$
    and *dmcat-Rel* α $a(\!|DM\alpha|\!)$ = *Mα-Rel* (*cat-Rel* α)
    and *dmcat-Rel* α $a(\!|DMl|\!)$ = *Ml-Rel* (*cat-Rel* α) a
    and *dmcat-Rel* α $a(\!|DMr|\!)$ = *Mr-Rel* (*cat-Rel* α) a
  ⟨*proof*⟩

Slicing.

lemma *dmcat-dagcat-dmcat-Rel*: *dmcat-dagcat* (*dmcat-Rel* α a) = *dagcat-Rel* α
⟨*proof*⟩

lemma *dmcat-mcat-dmcat-Rel*: *dmcat-mcat* (*dmcat-Rel* α a) = *mcat-Rel* α a
⟨*proof*⟩

34.10.2 *Rel* is a dagger monoidal category

lemma (**in** $\mathcal{Z}$) *dagger-monoidal-category-dmcat-Rel*:
  assumes $A \in_\circ$ *cat-Rel* $\alpha(\!|Obj|\!)$
  shows *dagger-monoidal-category* α (*dmcat-Rel* α A)
⟨*proof*⟩

References

[1] nLab. URL https://ncatlab.org/nlab/show/HomePage.

[2] Wikipedia, 2001. URL https://www.wikipedia.org/.

[3] P. Bodo. *Categories and Functors*. Academic Press, New York, 1970.

[4] P. I. Etingof, S. Gelaki, D. Nikshych, and V. Ostrik. *Tensor Categories*. Number 205 in Mathematical Surveys and Monographs. American Mathematical Society, Providence, 2015. ISBN 978-1-4704-2024-6.

[5] S. Feferman and G. Kreisel. Set-Theoretical Foundations of Category Theory. In M. Barr, P. Berthiaume, B. J. Day, J. Duskin, S. Feferman, G. M. Kelly, S. Mac Lane, M. Tierney, and R. F. C. Walters, editors, *Reports of the Midwest Category Seminar III*, Lecture Notes in Mathematics, pages 201–247, Heidelberg, 1969. Springer.

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

[7] 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.

[8] M. Milehins. Category Theory for ZFC in HOL I: Foundations: Design Patterns, Set Theory, Digraphs, Semicategories. *Archive of Formal Proofs*, 2021.

[9] 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.

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

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

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

[13] P. Selinger. A Survey of Graphical Languages for Monoidal Categories. In B. Coecke, editor, *New Structures for Physics*, volume 813, pages 289–355. Springer, Heidelberg, 2010. ISBN 978-3-642-12820-2.

[14] G. Takeuti and W. M. Zaring. *Introduction to Axiomatic Set Theory*. Springer-Verlag, Heidelberg, 1971. ISBN 0-387-05302-6.