Category Theory for ZFC in HOL I: Foundations Design Patterns, Set Theory, Digraphs, Semicategories

Mihails Milehins

March 19, 2025

Abstract

This article provides a foundational framework for the formalization of category theory in the object logic *ZFC in HOL* ([51], also see [47]) of the formal proof assistant *Isabelle* [49]. More specifically, this article provides a formalization of canonical set-theoretic constructions internalized in the type V associated with the ZFC in HOL, establishes a design pattern for the formalization of mathematical structures using sequences and locales [33, 8, 9], and showcases the developed infrastructure by providing formalizations of the elementary theories of digraphs and semicategories. The methodology chosen for the formalization of the theories of digraphs and semicategories (and categories in future articles) rests on the ideas that were originally expressed in [28]. Thus, in the context of this work, each of the aforementioned mathematical structures is represented as a term of the type V embedded into a stage of the von Neumann hierarchy [59].

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" [31] 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

Introduction

1.1 Background

This article presents a foundational framework that will be used for the formalization of elements of the theory of 1-categories in the object logic *ZFC in HOL* ([51], also see [47]) of the formal proof assistant *Isabelle* [49] in future articles. It is important to note that this chapter serves as an introduction to the entire development and not merely its foundational part.

There already exist several formalizations of the foundations of category theory in Isabelle. In the context of the work presented here, the most relevant formalizations (listed in the chronological order) are [25, 24], [48], [34] and [58]. Arguably, the most well developed and maintained entry is [58]: it subsumes the majority of the content of [48] and [34].

From the perspective of the methodology that was chosen for the formalization, this work differs significantly from the aforementioned previous work. In particular, the categories are modelled as terms of the type V and no attempt is made to generalize the concept of a category to arbitrary types. The inspiration for the chosen approach is drawn from [28], [52] and [57].

The primary references for this work are *Categories for the Working Mathematician* [39] by Saunders Mac Lane, *Category Theory in Context* by Emily Riehl [56] and *Categories and Functors* by Bodo Pareigis [15]. The secondary sources of information include the textbooks [7] and [32], as well as several online encyclopedias (including [3], [5], [4] and [2]). Of course, inspiration was also drawn from the previous formalizations of category theory in Isabelle.

It is likely that none of the content that is formalized in this work is original in nature. However, explicit citations are not provided for many results that were deemed to be trivial.

1.2 Related and previous work

To the best knowledge of the author, this work is the first attempt to develop a formalization of elements of category theory in the object logic ZFC in HOL by modelling categories as terms of the type V. However, it should be noted that the formalization of category theory in [34] largely rested on the object logic HOL/ZF [47], which is equiconsistent with the ZFC in HOL [51]. Nonetheless, in [34], the objects and arrows associated with categories were modelled as terms of arbitrary types. The object logic HOL/ZF was used for the exposition of the category *Set* of all sets and functions between them and a variety of closely related concepts. In this sense, the methodology employed in [34] could be seen as a combination of the methodology employed in this work and the methodology followed in [48] and [58]. Furthermore, in [26], the authors have experimented with the formalization of category theory in Higher-Order Tarski-Grothendieck (HOTG) theory [17] using a methodology that shares many similarities with the approach that was chosen in this study.

The formalizations of various elements of category theory in other proof assistants are abundant. While a survey of such formalizations is outside of the scope of this work, it is important to note that there exist at least two examples of the formalization of elements of category theory in a set-theoretic setting similar to the one that is used in this work. More specifically, elements of category theory were formalized in the Tarski-Grothendieck Set Theory in the Mizar proof

assistant [1] (and published in the associated electronic journal [30]) and the proof assistant Metamath [40]. The following references contain some of the relevant articles in [30], but the list may not be exhaustive: [18, 19, 21, 61, 20, 43, 60, 44, 45, 13, 22, 62, 23, 10, 63, 11, 64, 46, 36, 37, 12, 38, 53, 29, 54, 55].

Set Theory for Category Theory

2.1 Introduction

2.1.1 Background

This chapter presents a formalization of the elements of set theory in the object logic *ZFC in HOL* ([51], also see [47]) of the formal proof assistant Isabelle [49]. The emphasis of this work is on the improvement of the convenience of the formalization of abstract mathematics internalized in the type *V*.

2.1.2 References, related and previous work

The results that are presented in this chapter cannot be traced to a single source of information. Nonetheless, the results are not original. A significant number of these results was carried over (with amendments) from the main library of Isabelle/HOL [6]. Other results were inspired by elements of the content of the books *Introduction to Axiomatic Set Theory* by G. Takeuti and W. M. Zaring [59], *Theory of Sets* by N. Bourbaki [16] and *Algebra* by T. W. Hungerford [32]. Furthermore, several online encyclopedias and forums (including Wikipedia [5], ProofWiki [4], Encyclopedia of Mathematics [2], nLab [3] and Mathematics Stack Exchange) were used consistently throughout the development of this chapter. Inspiration for the work presented in this chapter has also been drawn from a similar ongoing project in the formalization of mathematics in the system HOTG (Higher Order Tarski-Grothendieck) [17, 26].

It should also be mentioned that the Isabelle/HOL and the Isabelle/ML code from the main distribution of Isabelle2020 and certain posts on the mailing list of Isabelle were frequently reused (with amendments) during the development of this chapter. Some of the specific examples of such reuse are

- The adoption of the implementation of the command **lemmas-with** that is available as part of the framework Types-To-Sets in the main distribution of Isabelle2020.
- The notation for the "multiway-if" was written by Manuel Eberl and appeared in a post on the mailing list of Isabelle: [27].

hide-const (**open**) *list.set Sum subset*

lemmas *ord-of-nat-zero* = *ord-of-nat.simps*(1)

2.1.3 Notation

abbreviation (*input*) *qm* (‹(- *?* - : -)› [0, 0, 10] 10)
  where *C ? A : B* ≡ *if C then A else B*
abbreviation (*input*) *if*2 **where** *if*2 *a b* ≡ (λ*i*. (*i* = 0 *? a* : *b*))

2.1.4 Further foundational results

lemma *theD*:
  assumes ∃!*x*. *P x* **and** *x* = (*THE x*. *P x*)
  shows *P x* **and** *P y* ⟹ *x* = *y*

⟨*proof*⟩

lemmas [*intro*] = *bij-betw-imageI*

lemma *bij-betwE*[*elim*]:
assumes *bij-betw f A B* **and** ⟦ *inj-on f A*; *f ‘ A = B* ⟧ ⟹ *P*
shows *P*
⟨*proof*⟩

lemma *bij-betwD*[*dest*]:
assumes *bij-betw f A B*
shows *inj-on f A* **and** *f ‘ A = B*
⟨*proof*⟩

lemma *ex1D*: ∃!*x*. *P x* ⟹ *P x* ⟹ *P y* ⟹ *x = y* ⟨*proof*⟩

2.2 Further set algebra and other miscellaneous results

2.2.1 Background

This section presents further set algebra and various elementary properties of sets.

Many of the results that are presented in this section were carried over (with amendments) from the theories *Set* and *Complete-Lattices* in the main library.

declare *elts-sup-iff*[*simp del*]

2.2.2 Further notation

Set membership

abbreviation *vmember* :: $V \Rightarrow V \Rightarrow bool$ (‹(-/ $\in_\circ$ -)› [51, 51] 50)
  where *vmember x A* ≡ ($x \in elts\ A$)
notation *vmember* (‹$'(\in_\circ')$›)
  and *vmember* (‹(-/ $\in_\circ$ -)› [51, 51] 50)

abbreviation *not-vmember* :: $V \Rightarrow V \Rightarrow bool$ (‹(-/ $\notin_\circ$ -)› [51, 51] 50)
  where *not-vmember x A* ≡ ($x \notin elts\ A$)
notation
  not-vmember (‹$'(\notin_\circ')$›) **and**
  not-vmember (‹(-/ $\notin_\circ$ -)› [51, 51] 50)

Subsets

abbreviation *vsubset* :: $V \Rightarrow V \Rightarrow bool$
  where *vsubset* ≡ *less*
abbreviation *vsubset-eq* :: $V \Rightarrow V \Rightarrow bool$
  where *vsubset-eq* ≡ *less-eq*

notation *vsubset* (‹$'(\subset_\circ')$›)
  and *vsubset* (‹(-/ $\subset_\circ$ -)› [51, 51] 50)
  and *vsubset-eq* (‹$'(\subseteq_\circ')$›)
  and *vsubset-eq* (‹(-/ $\subseteq_\circ$ -)› [51, 51] 50)

Ball

syntax
  -*VBall* :: *pttrn* ⇒ $V \Rightarrow bool \Rightarrow bool$ (‹(3∀ (-/$\in_\circ$-)./ -)› [0, 0, 10] 10)
  -*VBex* :: *pttrn* ⇒ $V \Rightarrow bool \Rightarrow bool$ (‹(3∃ (-/$\in_\circ$-)./ -)› [0, 0, 10] 10)
  -*VBex*1 :: *pttrn* ⇒ $V \Rightarrow bool \Rightarrow bool$ (‹(3∃!(-/$\in_\circ$-)./ -)› [0, 0, 10] 10)

syntax-consts
  -*VBall* ⇌ *Ball* **and**
  -*VBex* ⇌ *Bex* **and**
  -*VBex*1 ⇌ *Ex*1

translations
  $\forall x \in_\circ A.\ P \rightleftharpoons CONST\ Ball\ (CONST\ elts\ A)\ (\lambda x.\ P)$
  $\exists x \in_\circ A.\ P \rightleftharpoons CONST\ Bex\ (CONST\ elts\ A)\ (\lambda x.\ P)$
  $\exists! x \in_\circ A.\ P \rightharpoonup \exists! x.\ x \in_\circ A \wedge P$

VLambda

The following notation was adapted from [50].

syntax -*vlam* :: [*pttrn*, V, V] ⇒ V (‹(3λ-$\in_\circ$-./ -)› 10)

syntax-consts *-vlam* ⇌ *VLambda*
translations $\lambda x \in_\circ A.\ f \rightleftharpoons CONST\ VLambda\ A\ (\lambda x.\ f)$

Intersection and union

abbreviation *vintersection* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\cap_\circ$› 70)
  where $(\cap_\circ) \equiv (\sqcap)$
notation *vintersection* (**infixl** ‹$\cap_\circ$› 70)

abbreviation *vunion* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\cup_\circ$› 65)
  where *vunion* ≡ *sup*
notation *vunion* (**infixl** ‹$\cup_\circ$› 65)

abbreviation *VInter* :: $V \Rightarrow V$ (‹$\bigcap_\circ$›)
  where $\bigcap_\circ\ A \equiv \bigsqcap\ (elts\ A)$
notation *VInter* (‹$\bigcap_\circ$›)

abbreviation *VUnion* :: $V \Rightarrow V$ (‹$\bigcup_\circ$›)
  where $\bigcup_\circ A \equiv \bigsqcup\ (elts\ A)$
notation *VUnion* (‹$\bigcup_\circ$›)

Miscellaneous

notation *app* (‹-(|-|)› [999, 50] 999)
notation *vtimes* (**infixr** ‹$\times_\circ$› 80)

2.2.3 Elementary results.

lemma *vempty-nin*[*simp*]: $a \notin_\circ 0$ ⟨*proof*⟩

lemma *vemptyE*:
  assumes $A \neq 0$
  obtains x **where** $x \in_\circ A$
  ⟨*proof*⟩

lemma *in-set-CollectI*:
  assumes $P\ x$ **and** *small* $\{x.\ P\ x\}$
  shows $x \in_\circ set\ \{x.\ P\ x\}$
  ⟨*proof*⟩

lemma *small-setcompr*2:
  assumes *small* $\{f\ x\ y \mid x\ y.\ P\ x\ y\}$ **and** $a \in_\circ set\ \{f\ x\ y \mid x\ y.\ P\ x\ y\}$
  obtains $x'\ y'$ **where** $a = f\ x'\ y'$ **and** $P\ x'\ y'$
  ⟨*proof*⟩

lemma *in-small-setI*:
  assumes *small* A **and** $x \in A$
  shows $x \in_\circ set\ A$
  ⟨*proof*⟩

lemma *in-small-setD*:
  assumes $x \in_\circ set\ A$ **and** *small* A
  shows $x \in A$
  ⟨*proof*⟩

lemma *in-small-setE*:
  assumes $a \in_\circ set\ A$ **and** *small* A
  obtains $a \in A$
  ⟨*proof*⟩

lemma *small-set-vsubset*:
assumes *small A* **and** $A \subseteq$ *elts B*
shows *set A* $\subseteq_\circ$ *B*
⟨*proof*⟩

lemma *some-in-set-if-set-neq-vempty*[*simp*]:
assumes $A \neq 0$
shows (*SOME x*. $x \in_\circ A$) $\in_\circ A$
⟨*proof*⟩

lemma *small-vsubset-set*[*intro*, *simp*]:
assumes *small B* **and** $A \subseteq B$
shows *set A* $\subseteq_\circ$ *set B*
⟨*proof*⟩

lemma *vset-neq*-1:
assumes $b \notin_\circ A$ **and** $a \in_\circ A$
shows $b \neq a$
⟨*proof*⟩

lemma *vset-neq*-2:
assumes $b \in_\circ A$ **and** $a \notin_\circ A$
shows $b \neq a$
⟨*proof*⟩

lemma *nin-vinsertI*:
assumes $a \neq b$ **and** $a \notin_\circ A$
shows $a \notin_\circ$ *vinsert b A*
⟨*proof*⟩

lemma *vsubset-if-subset*:
assumes *elts A* $\subseteq$ *elts B*
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *small-set-comprehension*[*simp*]: *small* $\{A\ i \mid i.\ i \in_\circ I\}$
⟨*proof*⟩

2.2.4 *VBall*

lemma *vball-cong*:
assumes $A = B$ **and** $\bigwedge x.\ x \in_\circ B \Longrightarrow P\ x \longleftrightarrow Q\ x$
shows $(\forall\, x{\in_\circ}A.\ P\ x) \longleftrightarrow (\forall\, x{\in_\circ}B.\ Q\ x)$
⟨*proof*⟩

lemma *vball-cong-simp*[*cong*]:
assumes $A = B$ **and** $\bigwedge x.\ x \in_\circ B$ =*simp*=> $P\ x \longleftrightarrow Q\ x$
shows $(\forall\, x{\in_\circ}A.\ P\ x) \longleftrightarrow (\forall\, x{\in_\circ}B.\ Q\ x)$
⟨*proof*⟩

2.2.5 *VBex*

lemma *vbex-cong*:
assumes $A = B$ **and** $\bigwedge x.\ x \in_\circ B \Longrightarrow P\ x \longleftrightarrow Q\ x$
shows $(\exists\, x{\in_\circ}A.\ P\ x) \longleftrightarrow (\exists\, x{\in_\circ}B.\ Q\ x)$
⟨*proof*⟩

lemma *vbex-cong-simp*[*cong*]:
assumes $A = B$ **and** $\bigwedge x.\ x \in_\circ B \Rightarrow P\ x \longleftrightarrow Q\ x$
shows $(\exists\, x{\in_\circ}A.\ P\ x) \longleftrightarrow (\exists\, x{\in_\circ}B.\ Q\ x)$
⟨*proof*⟩

2.2.6 Subset

Rules.

lemma *vsubset-antisym*:
assumes $A \subseteq_\circ B$ **and** $B \subseteq_\circ A$
shows $A = B$
⟨*proof*⟩

lemma *vsubsetI*:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow x \in_\circ B$
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *vpsubsetI*:
assumes $A \subseteq_\circ B$ **and** $x \notin_\circ A$ **and** $x \in_\circ B$
shows $A \subset_\circ B$
⟨*proof*⟩

lemma *vsubsetD*:
assumes $A \subseteq_\circ B$
shows $\bigwedge x.\ x \in_\circ A \Longrightarrow x \in_\circ B$
⟨*proof*⟩

lemma *vsubsetE*:
assumes $A \subseteq_\circ B$ **and** $(\bigwedge x.\ x \in_\circ A \Longrightarrow x \in_\circ B) \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vpsubsetE*:
assumes $A \subset_\circ B$
obtains x **where** $A \subseteq_\circ B$ **and** $x \notin_\circ A$ **and** $x \in_\circ B$
⟨*proof*⟩

lemma *vsubset-iff*: $A \subseteq_\circ B \longleftrightarrow (\forall\, t.\ t \in_\circ A \longrightarrow t \in_\circ B)$ ⟨*proof*⟩

Elementary properties.

lemma *vsubset-eq*: $A \subseteq_\circ B \longleftrightarrow (\forall\, x{\in_\circ}A.\ x \in_\circ B)$ ⟨*proof*⟩

lemma *vsubset-transitive*[*intro*]:
assumes $A \subseteq_\circ B$ **and** $B \subseteq_\circ C$
shows $A \subseteq_\circ C$
⟨*proof*⟩

lemma *vsubset-reflexive*: $A \subseteq_\circ A$ ⟨*proof*⟩

Set operations.

lemma *vsubset-vempty*: $0 \subseteq_\circ A$ ⟨*proof*⟩

lemma *vsubset-vsingleton-left*: $set\ \{a\} \subseteq_\circ A \longleftrightarrow a \in_\circ A$ ⟨*proof*⟩

lemmas *vsubset-vsingleton-leftD*[*dest*] = *vsubset-vsingleton-left*[*THEN iffD*1]
and *vsubset-vsingleton-leftI*[*intro*] = *vsubset-vsingleton-left*[*THEN iffD*2]

lemma *vsubset-vsingleton-right*: $A \subseteq_\circ set\ \{a\} \longleftrightarrow A = set\ \{a\} \vee A = 0$
⟨*proof*⟩

lemmas *vsubset-vsingleton-rightD*[*dest*] = *vsubset-vsingleton-right*[*THEN iffD*1]
and *vsubset-vsingleton-rightI*[*intro*] = *vsubset-vsingleton-right*[*THEN iffD*2]

lemma *vsubset-vdoubleton-leftD*[*dest*]:
assumes $set\ \{a, b\} \subseteq_\circ A$
shows $a \in_\circ A$ **and** $b \in_\circ A$
⟨*proof*⟩

lemma *vsubset-vdoubleton-leftI*[*intro*]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows $set\ \{a, b\} \subseteq_\circ A$
⟨*proof*⟩

lemma *vsubset-vinsert-leftD*[*dest*]:
assumes *vinsert a A* $\subseteq_\circ B$
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *vsubset-vinsert-leftI*[*intro*]:
assumes $A \subseteq_\circ B$ **and** $a \in_\circ B$
shows *vinsert a A* $\subseteq_\circ B$
⟨*proof*⟩

lemma *vsubset-vinsert-vinsertI*[*intro*]:
assumes $A \subseteq_\circ$ *vinsert b B*
shows *vinsert b A* $\subseteq_\circ$ *vinsert b B*
⟨*proof*⟩

lemma *vsubset-vinsert-rightI*[*intro*]:
assumes $A \subseteq_\circ B$
shows $A \subseteq_\circ$ *vinsert b B*
⟨*proof*⟩

lemmas *vsubset-VPow* = *VPow-le-VPow-iff*

lemmas *vsubset-VPowD* = *vsubset-VPow*[*THEN iffD*1]
and *vsubset-VPowI* = *vsubset-VPow*[*THEN iffD*2]

Special properties.

lemma *vsubset-contraD*:
assumes $A \subseteq_\circ B$ **and** $c \notin_\circ B$
shows $c \notin_\circ A$
⟨*proof*⟩

2.2.7 Equality

Rules.

lemma *vequalityD*1:
assumes $A = B$
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *vequalityD*2:

assumes $A = B$
shows $B \subseteq_\circ A$
⟨*proof*⟩

lemma *vequalityE*:
assumes $A = B$ **and** $[\![\ A \subseteq_\circ B;\ B \subseteq_\circ A\ ]\!] \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vequalityCE*[*elim*]:
assumes $A = B$ **and** $[\![\ c \in_\circ A;\ c \in_\circ B\ ]\!] \Longrightarrow P$ **and** $[\![\ c \notin_\circ A;\ c \notin_\circ B\ ]\!] \Longrightarrow P$
shows P
⟨*proof*⟩

2.2.8 Binary intersection

lemma *vintersection-def*: $A \cap_\circ B = set\ \{x.\ x \in_\circ A \wedge x \in_\circ B\}$
⟨*proof*⟩

lemma *small-vintersection-set*[*simp*]: *small* $\{x.\ x \in_\circ A \wedge x \in_\circ B\}$
⟨*proof*⟩

Rules.

lemma *vintersection-iff*[*simp*]: $x \in_\circ A \cap_\circ B \longleftrightarrow x \in_\circ A \wedge x \in_\circ B$
⟨*proof*⟩

lemma *vintersectionI*[*intro*!]:
assumes $x \in_\circ A$ **and** $x \in_\circ B$
shows $x \in_\circ A \cap_\circ B$
⟨*proof*⟩

lemma *vintersectionD1*[*dest*]:
assumes $x \in_\circ A \cap_\circ B$
shows $x \in_\circ A$
⟨*proof*⟩

lemma *vintersectionD2*[*dest*]:
assumes $x \in_\circ A \cap_\circ B$
shows $x \in_\circ B$
⟨*proof*⟩

lemma *vintersectionE*[*elim*!]:
assumes $x \in_\circ A \cap_\circ B$ **and** $x \in_\circ A \Longrightarrow x \in_\circ B \Longrightarrow P$
shows P
⟨*proof*⟩

Elementary properties.

lemma *vintersection-intersection*: $A \cap_\circ B = set\ (elts\ A \cap elts\ B)$
⟨*proof*⟩

lemma *vintersection-assoc*: $A \cap_\circ (B \cap_\circ C) = (A \cap_\circ B) \cap_\circ C$ ⟨*proof*⟩

lemma *vintersection-commute*: $A \cap_\circ B = B \cap_\circ A$ ⟨*proof*⟩

Previous set operations.

lemma *vsubset-vintersection-right*: $A \subseteq_\circ (B \cap_\circ C) = (A \subseteq_\circ B \wedge A \subseteq_\circ C)$
⟨*proof*⟩

lemma *vsubset-vintersection-rightD*[*dest*]:
assumes $A \subseteq_\circ (B \cap_\circ C)$
shows $A \subseteq_\circ B$ **and** $A \subseteq_\circ C$
⟨*proof*⟩

lemma *vsubset-vintersection-rightI*[*intro*]:
assumes $A \subseteq_\circ B$ **and** $A \subseteq_\circ C$
shows $A \subseteq_\circ (B \cap_\circ C)$
⟨*proof*⟩

Set operations.

lemma *vintersection-vempty*: $0 \subseteq_\circ A$ ⟨*proof*⟩

lemma *vintersection-vsingleton*: $a \in_\circ set\ \{a\}$ ⟨*proof*⟩

lemma *vintersection-vdoubleton*: $a \in_\circ set\ \{a, b\}$ **and** $b \in_\circ set\ \{a, b\}$
⟨*proof*⟩

lemma *vintersection-VPow*[*simp*]: $VPow\ (A \cap_\circ B) = VPow\ A \cap_\circ VPow\ B$ ⟨*proof*⟩

Special properties.

lemma *vintersection-function-mono*:
assumes *mono f*
shows $f\ (A \cap_\circ B) \subseteq_\circ f\ A \cap_\circ f\ B$
⟨*proof*⟩

2.2.9 Binary union

lemma *small-vunion-set*: $small\ \{x.\ x \in_\circ A \vee x \in_\circ B\}$
⟨*proof*⟩

Rules.

lemma *vunion-def*: $A \cup_\circ B = set\ \{x.\ x \in_\circ A \vee x \in_\circ B\}$
⟨*proof*⟩

lemma *vunion-iff*[*simp*]: $x \in_\circ A \cup_\circ B \longleftrightarrow x \in_\circ A \vee x \in_\circ B$
⟨*proof*⟩

lemma *vunionI*1:
assumes $a \in_\circ A$
shows $a \in_\circ A \cup_\circ B$
⟨*proof*⟩

lemma *vunionI*2:
assumes $a \in_\circ B$
shows $a \in_\circ A \cup_\circ B$
⟨*proof*⟩

lemma *vunionCI*[*intro*!]:
assumes $x \notin_\circ B \Longrightarrow x \in_\circ A$
shows $x \in_\circ A \cup_\circ B$
⟨*proof*⟩

lemma *vunionE*[*elim*!]:
assumes $x \in_\circ A \cup_\circ B$ **and** $x \in_\circ A \Longrightarrow P$ **and** $x \in_\circ B \Longrightarrow P$
shows P
⟨*proof*⟩

Elementary properties.

lemma *vunion-union*: $A \cup_\circ B = set\ (elts\ A \cup elts\ B)$ ⟨*proof*⟩

lemma *vunion-assoc*: $A \cup_\circ (B \cup_\circ C) = (A \cup_\circ B) \cup_\circ C$ ⟨*proof*⟩

lemma *vunion-commute*: $A \cup_\circ B = B \cup_\circ A$ ⟨*proof*⟩

Previous set operations.

lemma *vsubset-vunion-left*: $(A \cup_\circ B) \subseteq_\circ C \longleftrightarrow (A \subseteq_\circ C \wedge B \subseteq_\circ C)$ ⟨*proof*⟩

lemma *vsubset-vunion-leftD*[*dest*]:
assumes $(A \cup_\circ B) \subseteq_\circ C$
shows $A \subseteq_\circ C$ **and** $B \subseteq_\circ C$
⟨*proof*⟩

lemma *vsubset-vunion-leftI*[*intro*]:
assumes $A \subseteq_\circ C$ **and** $B \subseteq_\circ C$
shows $(A \cup_\circ B) \subseteq_\circ C$
⟨*proof*⟩

lemma *vintersection-vunion-left*: $(A \cup_\circ B) \cap_\circ C = (A \cap_\circ C) \cup_\circ (B \cap_\circ C)$
⟨*proof*⟩

lemma *vintersection-vunion-right*: $A \cap_\circ (B \cup_\circ C) = (A \cap_\circ B) \cup_\circ (A \cap_\circ C)$
⟨*proof*⟩

Set operations.

lemmas *vunion-vempty-left* = *sup-V-0-left*
and *vunion-vempty-right* = *sup-V-0-right*

lemma *vunion-vsingleton*[*simp*]: $set\ \{a\} \cup_\circ A = vinsert\ a\ A$ ⟨*proof*⟩

lemma *vunion-vdoubleton*[*simp*]: $set\ \{a, b\} \cup_\circ A = vinsert\ a\ (vinsert\ b\ A)$
⟨*proof*⟩

lemma *vunion-vinsert-comm-left*:
$(vinsert\ a\ A) \cup_\circ B = A \cup_\circ (vinsert\ a\ B)$
⟨*proof*⟩

lemma *vunion-vinsert-comm-right*:
$A \cup_\circ (vinsert\ a\ B) = (vinsert\ a\ A) \cup_\circ B$
⟨*proof*⟩

lemma *vinsert-def*: $vinsert\ y\ B = set\ \{x.\ x = y\} \cup_\circ B$ ⟨*proof*⟩

lemma *vunion-vinsert-left*: $(vinsert\ a\ A) \cup_\circ B = vinsert\ a\ (A \cup_\circ B)$ ⟨*proof*⟩

lemma *vunion-vinsert-right*: $A \cup_\circ (vinsert\ a\ B) = vinsert\ a\ (A \cup_\circ B)$ ⟨*proof*⟩

Special properties.

lemma *vunion-fun-mono*:
assumes *mono f*
shows $f\ A \cup_\circ f\ B \subseteq_\circ f\ (A \cup_\circ B)$
⟨*proof*⟩

2.2.10 Set difference

definition *vdiff* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹−∘› 65)

where $A -_\circ B = \mathit{set}\ \{x.\ x \in_\circ A \wedge x \notin_\circ B\}$
notation *vdiff* (**infixl** $\langle -_\circ \rangle$ 65)

lemma *small-set-vdiff*[*simp*]: *small* $\{x.\ x \in_\circ A \wedge x \notin_\circ B\}$
⟨*proof*⟩

Rules.

lemma *vdiff-iff*[*simp*]: $x \in_\circ A -_\circ B \longleftrightarrow x \in_\circ A \wedge x \notin_\circ B$
⟨*proof*⟩

lemma *vdiffI*[*intro*!]:
assumes $x \in_\circ A$ **and** $x \notin_\circ B$
shows $x \in_\circ A -_\circ B$
⟨*proof*⟩

lemma *vdiffD*1:
assumes $x \in_\circ A -_\circ B$
shows $x \in_\circ A$
⟨*proof*⟩

lemma *vdiffD*2:
assumes $x \in_\circ A -_\circ B$ **and** $x \in_\circ B$
shows P
⟨*proof*⟩

lemma *vdiffE*[*elim*!]:
assumes $x \in_\circ A -_\circ B$ **and** $[\![\ x \in_\circ A;\ x \notin_\circ B\ ]\!] \Longrightarrow P$
shows P
⟨*proof*⟩

Previous set operations.

lemma *vsubset-vdiff*:
assumes $A \subseteq_\circ B -_\circ C$
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *vinsert-vdiff-nin*[*simp*]:
assumes $a \notin_\circ B$
shows *vinsert* $a\ (A -_\circ B) = \mathit{vinsert}\ a\ A -_\circ B$
⟨*proof*⟩

Set operations.

lemma *vdiff-vempty-left*[*simp*]: $0 -_\circ A = 0$ ⟨*proof*⟩

lemma *vdiff-vempty-right*[*simp*]: $A -_\circ 0 = A$ ⟨*proof*⟩

lemma *vdiff-vsingleton-vinsert*[*simp*]: *set* $\{a\} -_\circ \mathit{vinsert}\ a\ A = 0$ ⟨*proof*⟩

lemma *vdiff-vsingleton-in*[*simp*]:
assumes $a \in_\circ A$
shows *set* $\{a\} -_\circ A = 0$
⟨*proof*⟩

lemma *vdiff-vsingleton-nin*[*simp*]:
assumes $a \notin_\circ A$
shows *set* $\{a\} -_\circ A = \mathit{set}\ \{a\}$
⟨*proof*⟩

lemma *vdiff-vinsert-vsingleton*[*simp*]: *vinsert a A* $-_\circ$ *set* $\{a\}$ = *A* $-_\circ$ *set* $\{a\}$
⟨*proof*⟩

lemma *vdiff-vsingleton*[*simp*]:
assumes $a \notin_\circ A$
shows *A* $-_\circ$ *set* $\{a\}$ = *A*
⟨*proof*⟩

lemma *vdiff-vsubset*:
assumes $A \subseteq_\circ B$ **and** $D \subseteq_\circ C$
shows *A* $-_\circ$ *C* $\subseteq_\circ$ *B* $-_\circ$ *D*
⟨*proof*⟩

lemma *vdiff-vinsert-left-in*[*simp*]:
assumes $a \in_\circ B$
shows (*vinsert a A*) $-_\circ$ *B* = *A* $-_\circ$ *B*
⟨*proof*⟩

lemma *vdiff-vinsert-left-nin*:
assumes $a \notin_\circ B$
shows (*vinsert a A*) $-_\circ$ *B* = *vinsert a* (*A* $-_\circ$ *B*)
⟨*proof*⟩

lemma *vdiff-vinsert-right-in*: *A* $-_\circ$ (*vinsert a B*) = *A* $-_\circ$ *B* $-_\circ$ *set* $\{a\}$ ⟨*proof*⟩

lemma *vdiff-vinsert-right-nin*[*simp*]:
assumes $a \notin_\circ A$
shows *A* $-_\circ$ (*vinsert a B*) = *A* $-_\circ$ *B*
⟨*proof*⟩

lemma *vdiff-vintersection-left*: (*A* $\cap_\circ$ *B*) $-_\circ$ *C* = (*A* $-_\circ$ *C*) $\cap_\circ$ (*B* $-_\circ$ *C*) ⟨*proof*⟩

lemma *vdiff-vunion-left*: (*A* $\cup_\circ$ *B*) $-_\circ$ *C* = (*A* $-_\circ$ *C*) $\cup_\circ$ (*B* $-_\circ$ *C*) ⟨*proof*⟩

Special properties.

lemma *complement-vsubset*: *I* $-_\circ$ *J* $\subseteq_\circ$ *I* ⟨*proof*⟩

lemma *vintersection-complement*: (*I* $-_\circ$ *J*) $\cap_\circ$ *J* = 0 ⟨*proof*⟩

lemma *vunion-complement*:
assumes $J \subseteq_\circ I$
shows *I* $-_\circ$ *J* $\cup_\circ$ *J* = *I*
⟨*proof*⟩

2.2.11 Augmenting a set with an element

lemma *vinsert-compr*: *vinsert y A* = *set* $\{x.\ x = y \vee x \in_\circ A\}$
⟨*proof*⟩

Rules.

lemma *vinsert-iff*[*simp*]: *x* $\in_\circ$ *vinsert y A* $\longleftrightarrow$ *x* = *y* $\vee$ *x* $\in_\circ$ *A* ⟨*proof*⟩

lemma *vinsertI*1: *x* $\in_\circ$ *vinsert x A* ⟨*proof*⟩

lemma *vinsertI*2:
assumes $x \in_\circ A$
shows *x* $\in_\circ$ *vinsert y A*

⟨*proof*⟩

lemma *vinsertE*1[*elim*!]:
assumes $x \in_\circ$ *vinsert* y A **and** $x = y \Longrightarrow P$ **and** $x \in_\circ A \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vinsertCI*[*intro*!]:
assumes $x \notin_\circ A \Longrightarrow x = y$
shows $x \in_\circ$ *vinsert* y A
⟨*proof*⟩

Elementary properties.

lemma *vinsert-insert*: *vinsert* a A = *set* (*insert* a (*elts* A)) ⟨*proof*⟩

lemma *vinsert-commute*: *vinsert* a (*vinsert* b C) = *vinsert* b (*vinsert* a C)
⟨*proof*⟩

lemma *vinsert-ident*:
assumes $x \notin_\circ A$ **and** $x \notin_\circ B$
shows *vinsert* x A = *vinsert* x $B \longleftrightarrow A = B$
⟨*proof*⟩

lemmas *vinsert-identD*[*dest*] = *vinsert-ident*[*THEN iffD*1, *rotated* 2]
and *vinsert-identI*[*intro*] = *vinsert-ident*[*THEN iffD*2]

Set operations.

lemma *vinsert-vempty*: *vinsert* a 0 = *set* $\{a\}$ ⟨*proof*⟩

lemma *vinsert-vsingleton*: *vinsert* a (*set* $\{b\}$) = *set* $\{a, b\}$ ⟨*proof*⟩

lemma *vinsert-vdoubleton*: *vinsert* a (*set* $\{b, c\}$) = *set* $\{a, b, c\}$ ⟨*proof*⟩

lemma *vinsert-vinsert*: *vinsert* a (*vinsert* b C) = *set* $\{a, b\} \cup_\circ C$ ⟨*proof*⟩

lemma *vinsert-vunion-left*: *vinsert* a ($A \cup_\circ B$) = *vinsert* a $A \cup_\circ B$ ⟨*proof*⟩

lemma *vinsert-vunion-right*: *vinsert* a ($A \cup_\circ B$) = $A \cup_\circ$ *vinsert* a B ⟨*proof*⟩

lemma *vinsert-vintersection*: *vinsert* a ($A \cap_\circ B$) = *vinsert* a $A \cap_\circ$ *vinsert* a B
⟨*proof*⟩

Special properties.

lemma *vinsert-set-insert-empty-anyI*:
assumes P (*vinsert* a 0)
shows P (*set* (*insert* a $\{\}$))
⟨*proof*⟩

lemma *vinsert-set-insert-anyI*:
assumes *small* B **and** P (*vinsert* a (*set* (*insert* b B)))
shows P (*set* (*insert* a (*insert* b B)))
⟨*proof*⟩

lemma *vinsert-set-insert-eq*:
assumes *small* B
shows *set* (*insert* a (*insert* b B)) = *vinsert* a (*set* (*insert* b B))
⟨*proof*⟩

lemma *vsubset-vinsert*:
$A \subseteq_\circ$ *vinsert* $x\ B \longleftrightarrow$ (*if* $x \in_\circ A$ *then* $A -_\circ$ *set* $\{x\} \subseteq_\circ B$ *else* $A \subseteq_\circ B$)
⟨*proof*⟩

lemma *vinsert-obtain-ne*:
assumes $A \neq 0$
obtains $a\ A'$ **where** $A =$ *vinsert* $a\ A'$ **and** $a \notin_\circ A'$
⟨*proof*⟩

2.2.12 Power set

Rules.

lemma *VPowI*:
assumes $A \subseteq_\circ B$
shows $A \in_\circ$ *VPow* B
⟨*proof*⟩

lemma *VPowD*:
assumes $A \in_\circ$ *VPow* B
shows $A \subseteq_\circ B$
⟨*proof*⟩

lemma *VPowE*[*elim*]:
assumes $A \in_\circ$ *VPow* B **and** $A \subseteq_\circ B \Longrightarrow P$
shows P
⟨*proof*⟩

Elementary properties.

lemma *VPow-bottom*: $0 \in_\circ$ *VPow* B ⟨*proof*⟩

lemma *VPow-top*: $A \in_\circ$ *VPow* A ⟨*proof*⟩

Set operations.

lemma *VPow-vempty*[*simp*]: *VPow* $0 =$ *set* $\{0\}$ ⟨*proof*⟩

lemma *VPow-vsingleton*[*simp*]: *VPow* (*set* $\{a\}$) = *set* $\{0,$ *set* $\{a\}\}$
⟨*proof*⟩

lemma *VPow-not-vempty*: *VPow* $A \neq 0$ ⟨*proof*⟩

lemma *VPow-mono*:
assumes $A \subseteq_\circ B$
shows *VPow* $A \subseteq_\circ$ *VPow* B
⟨*proof*⟩

lemma *VPow-vunion-subset*: *VPow* $A \cup_\circ$ *VPow* $B \subseteq_\circ$ *VPow* $(A \cup_\circ B)$ ⟨*proof*⟩

2.2.13 Singletons, using insert

Rules.

lemma *vsingletonI*[*intro*!]: $x \in_\circ$ *set* $\{x\}$ ⟨*proof*⟩

lemma *vsingletonD*[*dest*!]:
assumes $y \in_\circ$ *set* $\{x\}$
shows $y = x$

⟨*proof*⟩

lemma *vsingleton-iff*: $y \in_\circ set\ \{x\} \longleftrightarrow y = x$ ⟨*proof*⟩

Previous set operations.

lemma *VPow-vdoubleton*[*simp*]:
$VPow\ (set\ \{a,\ b\}) = set\ \{0,\ set\ \{a\},\ set\ \{b\},\ set\ \{a,\ b\}\}$
⟨*proof*⟩

lemma *vsubset-vinsertI*:
assumes $A -_\circ set\ \{x\} \subseteq_\circ B$
shows $A \subseteq_\circ vinsert\ x\ B$
⟨*proof*⟩

Special properties.

lemma *vsingleton-inject*:
assumes $set\ \{x\} = set\ \{y\}$
shows $x = y$
⟨*proof*⟩

lemma *vsingleton-insert-inj-eq*[*iff*]:
$set\ \{y\} = vinsert\ x\ A \longleftrightarrow x = y \wedge A \subseteq_\circ set\ \{y\}$
⟨*proof*⟩

lemma *vsingleton-insert-inj-eq'*[*iff*]:
$vinsert\ x\ A = set\ \{y\} \longleftrightarrow x = y \wedge A \subseteq_\circ set\ \{y\}$
⟨*proof*⟩

lemma *vsubset-vsingletonD*:
assumes $A \subseteq_\circ set\ \{x\}$
shows $A = 0 \vee A = set\ \{x\}$
⟨*proof*⟩

lemma *vsubset-vsingleton-iff*: $a \subseteq_\circ set\ \{x\} \longleftrightarrow a = 0 \vee a = set\ \{x\}$ ⟨*proof*⟩

lemma *vsubset-vdiff-vinsert*: $A \subseteq_\circ B -_\circ vinsert\ x\ C \longleftrightarrow A \subseteq_\circ B -_\circ C \wedge x \notin_\circ A$
⟨*proof*⟩

lemma *vunion-vsingleton-iff*:
$A \cup_\circ B = set\ \{x\} \longleftrightarrow$
$A = 0 \wedge B = set\ \{x\} \vee A = set\ \{x\} \wedge B = 0 \vee A = set\ \{x\} \wedge B = set\ \{x\}$
⟨*proof*⟩

lemma *vsingleton-Un-iff*:
$set\ \{x\} = A \cup_\circ B \longleftrightarrow$
$A = 0 \wedge B = set\ \{x\} \vee A = set\ \{x\} \wedge B = 0 \vee A = set\ \{x\} \wedge B = set\ \{x\}$
⟨*proof*⟩

lemma *VPow-vsingleton-iff*[*simp*]: $VPow\ X = set\ \{Y\} \longleftrightarrow X = 0 \wedge Y = 0$
⟨*proof*⟩

2.2.14 Intersection of elements

lemma *small-VInter*[*simp*]:
assumes $A \neq 0$
shows $small\ \{a.\ \forall\, x \in_\circ A.\ a \in_\circ x\}$
⟨*proof*⟩

lemma *VInter-def*: $\bigcap_\circ A = (\mathit{if}\ A = 0\ \mathit{then}\ 0\ \mathit{else}\ \mathit{set}\ \{a.\ \forall x \in_\circ A.\ a \in_\circ x\})$
⟨*proof*⟩

Rules.

lemma *VInter-iff*[*simp*]:
assumes [*simp*]: $A \neq 0$
shows $a \in_\circ \bigcap_\circ A \longleftrightarrow (\forall x \in_\circ A.\ a \in_\circ x)$
⟨*proof*⟩

lemma *VInterI*[*intro*]:
assumes $A \neq 0$ **and** $\bigwedge x.\ x \in_\circ A \Longrightarrow a \in_\circ x$
shows $a \in_\circ \bigcap_\circ A$
⟨*proof*⟩

lemma *VInter0I*[*intro*]:
assumes $A = 0$
shows $\bigcap_\circ A = 0$
⟨*proof*⟩

lemma *VInterD*[*dest*]:
assumes $a \in_\circ \bigcap_\circ A$ **and** $x \in_\circ A$
shows $a \in_\circ x$
⟨*proof*⟩

lemma *VInterE1*[*elim*]:
assumes $a \in_\circ \bigcap_\circ A$ **and** $x \notin_\circ A \Longrightarrow R$ **and** $a \in_\circ x \Longrightarrow R$
shows R
⟨*proof*⟩

lemma *VInterE2*[*elim*]:
assumes $a \in_\circ \bigcap_\circ A$
obtains x **where** $a \in_\circ x$ **and** $x \in_\circ A$
⟨*proof*⟩

lemma *VInterE3*:
assumes $a \in_\circ \bigcap_\circ A$ **and** $(\bigwedge y.\ y \in_\circ A \Longrightarrow a \in_\circ y) \Longrightarrow P$
shows P
⟨*proof*⟩

Elementary properties.

lemma *VInter-Inter*: $\bigcap_\circ A = \mathit{set}\ (\bigcap\ (\mathit{elts}\ `\ (\mathit{elts}\ A)))$
⟨*proof*⟩

lemma *VInter-eq*:
assumes [*simp*]: $A \neq 0$
shows $\bigcap_\circ A = \mathit{set}\ \{a.\ \forall x \in_\circ A.\ a \in_\circ x\}$
⟨*proof*⟩

Set operations.

lemma *VInter-vempty*[*simp*]: $\bigcap_\circ 0 = 0$ ⟨*proof*⟩

lemma *VInf-vempty*[*simp*]: $\prod \{\} = (0::V)$ ⟨*proof*⟩

lemma *VInter-vdoubleton*: $\bigcap_\circ (\mathit{set}\ \{a, b\}) = a \cap_\circ b$
⟨*proof*⟩

lemma *VInter-antimono*:

assumes $B \neq 0$ **and** $B \subseteq_\circ A$
shows $\bigcap_\circ A \subseteq_\circ \bigcap_\circ B$
⟨*proof*⟩

lemma *VInter-vsubset*:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow x \subseteq_\circ B$ **and** $A \neq 0$
shows $\bigcap_\circ A \subseteq_\circ B$
⟨*proof*⟩

lemma *VInter-vinsert*:
assumes $A \neq 0$
shows $\bigcap_\circ (\mathit{vinsert}\ a\ A) = a \cap_\circ \bigcap_\circ A$
⟨*proof*⟩

lemma *VInter-vunion*:
assumes $A \neq 0$ **and** $B \neq 0$
shows $\bigcap_\circ(A \cup_\circ B) = \bigcap_\circ A \cap_\circ \bigcap_\circ B$
⟨*proof*⟩

lemma *VInter-vintersection*:
assumes $A \cap_\circ B \neq 0$
shows $\bigcap_\circ A \cup_\circ \bigcap_\circ B \subseteq_\circ \bigcap_\circ (A \cap_\circ B)$
⟨*proof*⟩

lemma *VInter-VPow*: $\bigcap_\circ (\mathit{VPow}\ A) \subseteq_\circ \mathit{VPow}\ (\bigcap_\circ A)$ ⟨*proof*⟩

Elementary properties.

lemma *VInter-lower*:
assumes $x \in_\circ A$
shows $\bigcap_\circ A \subseteq_\circ x$
⟨*proof*⟩

lemma *VInter-greatest*:
assumes $A \neq 0$ **and** $\bigwedge x.\ x \in_\circ A \Longrightarrow B \subseteq_\circ x$
shows $B \subseteq_\circ \bigcap_\circ A$
⟨*proof*⟩

2.2.15 Union of elements

lemma *Union-eq-VUnion*: $\bigcup(\mathit{elts}\ `\ \mathit{elts}\ A) = \{a.\ \exists x \in_\circ A.\ a \in_\circ x\}$ ⟨*proof*⟩

lemma *small-VUnion*[*simp*]: *small* $\{a.\ \exists x \in_\circ A.\ a \in_\circ x\}$
⟨*proof*⟩

lemma *VUnion-def*: $\bigcup_\circ A = \mathit{set}\ \{a.\ \exists x \in_\circ A.\ a \in_\circ x\}$
⟨*proof*⟩

Rules.

lemma *VUnion-iff*[*simp*]: $A \in_\circ \bigcup_\circ C \longleftrightarrow (\exists x \in_\circ C.\ A \in_\circ x)$ ⟨*proof*⟩

lemma *VUnionI*[*intro*]:
assumes $x \in_\circ A$ **and** $a \in_\circ x$
shows $a \in_\circ \bigcup_\circ A$
⟨*proof*⟩

lemma *VUnionE*[*elim*!]:
assumes $a \in_\circ \bigcup_\circ A$ **and** $\bigwedge x.\ a \in_\circ x \Longrightarrow x \in_\circ A \Longrightarrow R$
shows R

⟨*proof*⟩

Elementary properties.

lemma *VUnion-Union*: $\bigcup_\circ A = set\ (\bigcup\ (elts\ `\ (elts\ A)))$
⟨*proof*⟩

Set operations.

lemma *VUnion-vempty*[*simp*]: $\bigcup_\circ 0 = 0$ ⟨*proof*⟩

lemma *VUnion-vsingleton*[*simp*]: $\bigcup_\circ(set\ \{a\}) = a$ ⟨*proof*⟩

lemma *VUnion-vdoubleton*[*simp*]: $\bigcup_\circ(set\ \{a,\ b\}) = a \cup_\circ b$ ⟨*proof*⟩

lemma *VUnion-mono*:
assumes $A \subseteq_\circ B$
shows $\bigcup_\circ A \subseteq_\circ \bigcup_\circ B$
⟨*proof*⟩

lemma *VUnion-vinsert*: $\bigcup_\circ(vinsert\ x\ A) = x \cup_\circ \bigcup_\circ A$ ⟨*proof*⟩

lemma *VUnion-vintersection*: $\bigcup_\circ(A \cap_\circ B) \subseteq_\circ \bigcup_\circ A \cap_\circ \bigcup_\circ B$ ⟨*proof*⟩

lemma *VUnion-vunion*[*simp*]: $\bigcup_\circ(A \cup_\circ B) = \bigcup_\circ A \cup_\circ \bigcup_\circ B$ ⟨*proof*⟩

lemma *VUnion-VPow*[*simp*]: $\bigcup_\circ(VPow\ A) = A$ ⟨*proof*⟩

Special properties.

lemma *VUnion-vempty-conv-left*: $0 = \bigcup_\circ A \longleftrightarrow (\forall\, x{\in_\circ}A.\ x = 0)$ ⟨*proof*⟩

lemma *VUnion-vempty-conv-right*: $\bigcup_\circ A = 0 \longleftrightarrow (\forall\, x{\in_\circ}A.\ x = 0)$ ⟨*proof*⟩

lemma *vsubset-VPow-VUnion*: $A \subseteq_\circ VPow\ (\bigcup_\circ A)$ ⟨*proof*⟩

lemma *VUnion-vsubsetI*:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow \exists\, y.\ y \in_\circ B \wedge x \subseteq_\circ y$
shows $\bigcup_\circ A \subseteq_\circ \bigcup_\circ B$
⟨*proof*⟩

lemma *VUnion-upper*:
assumes $x \in_\circ A$
shows $x \subseteq_\circ \bigcup_\circ A$
⟨*proof*⟩

lemma *VUnion-least*:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow x \subseteq_\circ B$
shows $\bigcup_\circ A \subseteq_\circ B$
⟨*proof*⟩

2.2.16 Pairs

Further results

lemma *small-elts-of-set*[*simp*, *intro*]:
assumes *small x*
shows *elts* (*set x*) = *x*
⟨*proof*⟩

lemma *small-vpair*[*intro*, *simp*]:

assumes *small* {*a*. *P a*}
shows *small* {⟨*a*, *b*⟩ | *a*. *P a*}
⟨*proof*⟩

vpairs

definition *vpairs* :: *V* ⇒ *V* **where**
vpairs r = *set* {*x*. *x* ∈∘ *r* ∧ (∃ *a b*. *x* = ⟨*a*, *b*⟩)}

lemma *small-vpairs*[*simp*]: *small* {⟨*a*, *b*⟩ | *a b*. ⟨*a*, *b*⟩ ∈∘ *r*}
⟨*proof*⟩

Rules.

lemma *vpairsI*[*intro*]:
assumes *x* ∈∘ *r* **and** *x* = ⟨*a*, *b*⟩
shows *x* ∈∘ *vpairs r*
⟨*proof*⟩

lemma *vpairsD*[*dest*]:
assumes *x* ∈∘ *vpairs r*
shows *x* ∈∘ *r* **and** ∃ *a b*. *x* = ⟨*a*, *b*⟩
⟨*proof*⟩

lemma *vpairsE*[*elim*]:
assumes *x* ∈∘ *vpairs r*
obtains *a b* **where** *x* = ⟨*a*, *b*⟩ **and** ⟨*a*, *b*⟩ ∈∘ *r*
⟨*proof*⟩

lemma *vpairs-iff*: *x* ∈∘ *vpairs r* ⟷ *x* ∈∘ *r* ∧ (∃ *a b*. *x* = ⟨*a*, *b*⟩) ⟨*proof*⟩

Elementary properties.

lemma *vpairs-iff-elts*: ⟨*a*, *b*⟩ ∈∘ *vpairs r* ⟷ ⟨*a*, *b*⟩ ∈∘ *r* ⟨*proof*⟩

lemma *vpairs-iff-pairs*: ⟨*a*, *b*⟩ ∈∘ *vpairs r* ⟷ (*a*, *b*) ∈ *pairs r*
⟨*proof*⟩

Set operations.

lemma *vpairs-vempty*[*simp*]: *vpairs* 0 = 0 ⟨*proof*⟩

lemma *vpairs-vsingleton*[*simp*]: *vpairs* (*set* {⟨*a*, *b*⟩}) = *set* {⟨*a*, *b*⟩} ⟨*proof*⟩

lemma *vpairs-vinsert*: *vpairs* (*vinsert* ⟨*a*, *b*⟩ *A*) = *set* {⟨*a*, *b*⟩} ∪∘ *vpairs A*
⟨*proof*⟩

lemma *vpairs-mono*:
assumes *r* ⊆∘ *s*
shows *vpairs r* ⊆∘ *vpairs s*
⟨*proof*⟩

lemma *vpairs-vunion*: *vpairs* (*A* ∪∘ *B*) = *vpairs A* ∪∘ *vpairs B* ⟨*proof*⟩

lemma *vpairs-vintersection*: *vpairs* (*A* ∩∘ *B*) = *vpairs A* ∩∘ *vpairs B* ⟨*proof*⟩

lemma *vpairs-vdiff*: *vpairs* (*A* −∘ *B*) = *vpairs A* −∘ *vpairs B* ⟨*proof*⟩

Special properties.

lemma *vpairs-ex-vfst*:

assumes $x \in_\circ \mathit{vpairs}\ r$
shows $\exists\, b.\ \langle \mathit{vfst}\ x,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vpairs-ex-vsnd*:
assumes $y \in_\circ \mathit{vpairs}\ r$
shows $\exists\, a.\ \langle a,\ \mathit{vsnd}\ y\rangle \in_\circ r$
⟨*proof*⟩

2.2.17 Cartesian products

The following lemma is based on Theorem 6.2 from [59].

lemma *vtimes-vsubset-VPowVPow*: $A \times_\circ B \subseteq_\circ \mathit{VPow}\ (\mathit{VPow}\ (A \cup_\circ B))$
⟨*proof*⟩

2.2.18 Pairwise

definition *vpairwise* :: $(V \Rightarrow V \Rightarrow \mathit{bool}) \Rightarrow V \Rightarrow \mathit{bool}$
where $\mathit{vpairwise}\ R\ S \longleftrightarrow (\forall\, x{\in_\circ}S.\ \forall\, y{\in_\circ}S.\ x \neq y \longrightarrow R\ x\ y)$

Rules.

lemma *vpairwiseI*[*intro?*]:
assumes $\bigwedge x\ y.\ x \in_\circ S \Longrightarrow y \in_\circ S \Longrightarrow x \neq y \Longrightarrow R\ x\ y$
shows *vpairwise R S*
⟨*proof*⟩

lemma *vpairwiseD*[*dest*]:
assumes *vpairwise R S* **and** $x \in_\circ S$ **and** $y \in_\circ S$ **and** $x \neq y$
shows $R\ x\ y$ **and** $R\ y\ x$
⟨*proof*⟩

Elementary properties.

lemma *vpairwise-trivial*[*simp*]: $\mathit{vpairwise}\ (\lambda i\ j.\ j \neq i)\ I$
⟨*proof*⟩

Set operations.

lemma *vpairwise-vempty*[*simp*]: *vpairwise P* 0 ⟨*proof*⟩

lemma *vpairwise-vsingleton*[*simp*]: $\mathit{vpairwise}\ P\ (\mathit{set}\ \{A\})$
⟨*proof*⟩

lemma *vpairwise-vinsert*:
$\mathit{vpairwise}\ r\ (\mathit{vinsert}\ x\ s) \longleftrightarrow$
$(\forall\, y.\ y \in_\circ s \wedge y \neq x \longrightarrow r\ x\ y \wedge r\ y\ x) \wedge \mathit{vpairwise}\ r\ s$
⟨*proof*⟩

lemma *vpairwise-vsubset*:
assumes *vpairwise P S* **and** $T \subseteq_\circ S$
shows *vpairwise P T*
⟨*proof*⟩

lemma *vpairwise-mono*:
assumes *vpairwise P A* **and** $\bigwedge x\ y.\ P\ x\ y \Longrightarrow Q\ x\ y$ **and** $B \subseteq_\circ A$
shows *vpairwise Q B*
⟨*proof*⟩

2.2.19 Disjoint sets

abbreviation *vdisjnt* :: $V \Rightarrow V \Rightarrow bool$
where *vdisjnt A B* ≡ $A \cap_\circ B = 0$

Elementary properties.

lemma *vdisjnt-sym*:
assumes *vdisjnt A B*
shows *vdisjnt B A*
⟨*proof*⟩

lemma *vdisjnt-iff*: *vdisjnt A B* ⟷ $(\forall x.\ ^\sim\ (x \in_\circ A \wedge x \in_\circ B))$ ⟨*proof*⟩

Set operations.

lemma *vdisjnt-vempty*1[*simp*]: *vdisjnt* 0 *A*
and *vdisjnt-vempty*2[*simp*]: *vdisjnt A* 0
⟨*proof*⟩

lemma *vdisjnt-singleton*0[*simp*]: *vdisjnt* (*set* {*a*}) (*set* {*b*}) ⟷ $a \neq b$
and *vdisjnt-singleton*1[*simp*]: *vdisjnt* (*set* {*a*}) *A* ⟷ $a \notin_\circ A$
and *vdisjnt-singleton*2[*simp*]: *vdisjnt A* (*set* {*a*}) ⟷ $a \notin_\circ A$
⟨*proof*⟩

lemma *vdisjnt-vinsert-left*: *vdisjnt* (*vinsert a X*) *Y* ⟷ $a \notin_\circ Y \wedge$ *vdisjnt X Y*
⟨*proof*⟩

lemma *vdisjnt-vinsert-right*: *vdisjnt Y* (*vinsert a X*) ⟷ $a \notin_\circ Y \wedge$ *vdisjnt Y X*
⟨*proof*⟩

lemma *vdisjnt-vsubset-left*:
assumes *vdisjnt X Y* **and** $Z \subseteq_\circ X$
shows *vdisjnt Z Y*
⟨*proof*⟩

lemma *vdisjnt-vsubset-right*:
assumes *vdisjnt X Y* **and** $Z \subseteq_\circ Y$
shows *vdisjnt X Z*
⟨*proof*⟩

lemma *vdisjnt-vunion-left*: *vdisjnt* ($A \cup_\circ B$) *C* ⟷ *vdisjnt A C* ∧ *vdisjnt B C*
⟨*proof*⟩

lemma *vdisjnt-vunion-right*: *vdisjnt C* ($A \cup_\circ B$) ⟷ *vdisjnt C A* ∧ *vdisjnt C B*
⟨*proof*⟩

Special properties.

lemma *vdisjnt-vemptyI*[*intro*]:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow x \in_\circ B \Longrightarrow$ *False*
shows *vdisjnt A B*
⟨*proof*⟩

lemma *vdisjnt-self-iff-vempty*[*simp*]: *vdisjnt S S* ⟷ $S = 0$ ⟨*proof*⟩

lemma *vdisjntI*:
assumes $\bigwedge x\ y.\ x \in_\circ A \Longrightarrow y \in_\circ B \Longrightarrow x \neq y$
shows *vdisjnt A B*
⟨*proof*⟩

lemma *vdisjnt-nin-right*:
assumes *vdisjnt A B* **and** $a \in_\circ A$
shows $a \notin_\circ B$
⟨*proof*⟩

lemma *vdisjnt-nin-left*:
assumes *vdisjnt B A* **and** $a \in_\circ A$
shows $a \notin_\circ B$
⟨*proof*⟩

2.3 Further properties of natural numbers

2.3.1 Background

The section exposes certain fundamental properties of natural numbers and provides convenience utilities for doing arithmetic within the type V.

Many of the results that are presented in this sections were carried over (with amendments) from the theory *Nat* that can be found in the main library of Isabelle/HOL.

notation *ord-of-nat* ($\langle$-$_{\mathbb{N}}\rangle$ [999] 999)

named-theorems *nat-omega-simps*

declare *One-nat-def*[*simp del*]

abbreviation (*input*) *vpfst* **where** *vpfst a* ≡ $a(\!|0|\!)$
abbreviation (*input*) *vpsnd* **where** *vpsnd a* ≡ $a(\!|1_{\mathbb{N}}|\!)$
abbreviation (*input*) *vpthrd* **where** *vpthrd a* ≡ $a(\!|2_{\mathbb{N}}|\!)$

2.3.2 Conversion between V and *nat*

Primitive arithmetic

lemma *ord-of-nat-plus*[*nat-omega-simps*]: $a_{\mathbb{N}} + b_{\mathbb{N}} = (a + b)_{\mathbb{N}}$
⟨*proof*⟩

lemma *ord-of-nat-times*[*nat-omega-simps*]: $a_{\mathbb{N}} * b_{\mathbb{N}} = (a * b)_{\mathbb{N}}$
⟨*proof*⟩

lemma *ord-of-nat-succ*[*nat-omega-simps*]: *succ* ($a_{\mathbb{N}}$) = (*Suc a*)$_{\mathbb{N}}$ ⟨*proof*⟩

lemmas [*nat-omega-simps*] = *nat-cadd-eq-add*

lemma *ord-of-nat-csucc*[*nat-omega-simps*]: *csucc* ($a_{\mathbb{N}}$) = *succ* ($a_{\mathbb{N}}$)
⟨*proof*⟩

lemma *ord-of-nat-succ-vempty*[*nat-omega-simps*]: *succ* 0 = $1_{\mathbb{N}}$ ⟨*proof*⟩

lemma *ord-of-nat-vone*[*nat-omega-simps*]: 1 = $1_{\mathbb{N}}$ ⟨*proof*⟩

Transfer

definition *cr-omega* :: $V \Rightarrow nat \Rightarrow bool$
where *cr-omega a b* ⟷ (*a* = *ord-of-nat b*)

Transfer setup.

lemma *cr-omega-right-total*[*transfer-rule*]: *right-total cr-omega*
⟨*proof*⟩

lemma *cr-omega-bi-unqie*[*transfer-rule*]: *bi-unique cr-omega*
⟨*proof*⟩

lemma *omega-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-omega = ($\lambda x.\ x \in_\circ \omega$)
⟨*proof*⟩

lemma *omega-transfer*[*transfer-rule*]:
(*rel-set cr-omega*) (*elts* ω) (*UNIV*::*nat set*)
⟨*proof*⟩

lemma *omega-of-real-transfer*[*transfer-rule*]: *cr-omega* (*ord-of-nat a*) *a*
⟨*proof*⟩

Operations.

lemma *omega-succ-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-omega* ===> *cr-omega*) *succ Suc*
⟨*proof*⟩

lemma *omega-plus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-omega* ===> *cr-omega* ===> *cr-omega*) (+) (+)
⟨*proof*⟩

lemma *omega-mult-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-omega* ===> *cr-omega* ===> *cr-omega*) (*) (*)
⟨*proof*⟩

lemma *ord-of-nat-card-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*rel-set* (=) ===> *cr-omega*) (λ*x*. *ord-of-nat* (*card x*)) *card*
⟨*proof*⟩

lemma *ord-of-nat-transfer*[*transfer-rule*]:
(*rel-fun cr-omega* (=)) *id ord-of-nat*
⟨*proof*⟩

2.3.3 Elementary results

lemma *ord-of-nat-vempty*: $0 = 0_{\mathbb{N}}$ ⟨*proof*⟩

lemma *set-vzero-eq-ord-of-nat-vone*: *set* $\{0\} = 1_{\mathbb{N}}$
⟨*proof*⟩

lemma *vone-in-omega*[*simp*]: $1 \in_\circ \omega$ ⟨*proof*⟩

lemma *nat-of-omega*:
assumes $n \in_\circ \omega$
obtains *m* **where** $n = m_{\mathbb{N}}$
⟨*proof*⟩

lemma *omega-prev*:
assumes $n \in_\circ \omega$ **and** $0 \in_\circ n$
obtains *k* **where** $n = succ\ k$
⟨*proof*⟩

lemma *omega-vplus-commutative*:
assumes $a \in_\circ \omega$ **and** $b \in_\circ \omega$
shows $a + b = b + a$
⟨*proof*⟩

lemma *omega-vinetrsection*[*intro*]:
assumes $m \in_\circ \omega$ **and** $n \in_\circ \omega$
shows $m \cap_\circ n \in_\circ \omega$
⟨*proof*⟩

2.3.4 Induction

lemma *omega-induct-all*[*consumes* 1, *case-names step*]:
assumes $n \in_\circ \omega$ **and** $\bigwedge x.\ [\![x \in_\circ \omega;\ \bigwedge y.\ y \in_\circ x \Longrightarrow P\ y]\!] \Longrightarrow P\ x$
shows $P\ n$
⟨*proof*⟩

lemma *omega-induct*[*consumes* 1, *case-names* 0 *succ*]:
assumes $n \in_\circ \omega$ **and** $P\ 0$ **and** $\bigwedge n.\ [\![\ n \in_\circ \omega;\ P\ n\ ]\!] \Longrightarrow P\ (succ\ n)$
shows $P\ n$
⟨*proof*⟩

2.3.5 Methods

The following methods provide an infrastructure for working with goals of the form $a \in_\circ n_{\mathbb{N}} \Longrightarrow P\ a$.

lemma *in-succE*:
assumes $a \in_\circ succ\ n$ **and** $\bigwedge a.\ a \in_\circ n \Longrightarrow P\ a$ **and** $P\ n$
shows $P\ a$
⟨*proof*⟩

method *Suc-of-numeral* =
(
unfold numeral.simps add.assoc,
use nothing **in** ‹*unfold Suc-eq-plus*1*-left*[*symmetric*], *unfold One-nat-def*›
)

method *succ-of-numeral* =
(
Suc-of-numeral,
use nothing **in** ‹*unfold ord-of-nat-succ*[*symmetric*] *ord-of-nat-zero*›
)

method *numeral-of-succ* =
(
unfold nat-omega-simps,
use nothing **in**
‹
unfold numeral.simps[*symmetric*] *Suc-numeral add-num-simps*,
(*unfold numerals*(1))*?*
›
)

method *elim-in-succ* =
(
(
elim in-succE;
use nothing **in** ‹(*unfold triv-forall-equality*)*?*; (*numeral-of-succ*)*?*›
),
simp
)

method *elim-in-numeral* = (*succ-of-numeral*, *use nothing* **in** ‹*elim-in-succ*›)

2.3.6 Auxiliary

lemma *one*: $1_{\mathbb{N}} = set\ \{0\}$ ⟨*proof*⟩

lemma *two*: $2_{\mathbb{N}} = set\ \{0, 1_{\mathbb{N}}\}\ \langle proof \rangle$

lemma *three*: $3_{\mathbb{N}} = set\ \{0, 1_{\mathbb{N}}, 2_{\mathbb{N}}\}\ \langle proof \rangle$

lemma *four*: $4_{\mathbb{N}} = set\ \{0, 1_{\mathbb{N}}, 2_{\mathbb{N}}, 3_{\mathbb{N}}\}\ \langle proof \rangle$

lemma *two-vdiff-zero*[*simp*]: $set\ \{0, 1_{\mathbb{N}}\} -_\circ set\ \{0\} = set\ \{1_{\mathbb{N}}\}\ \langle proof \rangle$
lemma *two-vdiff-one*[*simp*]: $set\ \{0, 1_{\mathbb{N}}\} -_\circ set\ \{1_{\mathbb{N}}\} = set\ \{0\}\ \langle proof \rangle$

2.4 Elementary binary relations

2.4.1 Background

This section presents a theory of binary relations internalized in the type V and exposes elementary properties of two special types of binary relations: single-valued binary relations and injective single-valued binary relations.

Many of the results that are presented in this section were carried over (with amendments) from the theories *Set* and *Relation* in the main library.

2.4.2 Constructors

Identity relation

definition *vid-on* :: $V \Rightarrow V$
  where *vid-on* A = *set* $\{\langle a,\ a\rangle \mid a.\ a \in_\circ A\}$

lemma *vid-on-small*[*simp*]: *small* $\{\langle a,\ a\rangle \mid a.\ a \in_\circ A\}$
  ⟨*proof*⟩

Rules.

lemma *vid-on-eqI*:
  assumes $a = b$ **and** $a \in_\circ A$
  shows $\langle a,\ b\rangle \in_\circ$ *vid-on* A
  ⟨*proof*⟩

lemma *vid-onI*[*intro*!]:
  assumes $a \in_\circ A$
  shows $\langle a,\ a\rangle \in_\circ$ *vid-on* A
  ⟨*proof*⟩

lemma *vid-onD*[*dest*!]:
  assumes $\langle a,\ a\rangle \in_\circ$ *vid-on* A
  shows $a \in_\circ A$
  ⟨*proof*⟩

lemma *vid-onE*[*elim*!]:
  assumes $x \in_\circ$ *vid-on* A **and** $\exists\, a{\in_\circ}A.\ x = \langle a,\ a\rangle \Longrightarrow P$
  shows P
  ⟨*proof*⟩

lemma *vid-on-iff*: $\langle a,\ b\rangle \in_\circ$ *vid-on* $A \longleftrightarrow a = b \wedge a \in_\circ A$ ⟨*proof*⟩

Set operations.

lemma *vid-on-vempty*[*simp*]: *vid-on* $0 = 0$ ⟨*proof*⟩

lemma *vid-on-vsingleton*[*simp*]: *vid-on* (*set* $\{a\}$) = *set* $\{\langle a,\ a\rangle\}$ ⟨*proof*⟩

lemma *vid-on-vdoubleton*[*simp*]: *vid-on* (*set* $\{a,\ b\}$) = *set* $\{\langle a,\ a\rangle, \langle b,\ b\rangle\}$
  ⟨*proof*⟩

lemma *vid-on-mono*:
  assumes $A \subseteq_\circ B$
  shows *vid-on* $A \subseteq_\circ$ *vid-on* B
  ⟨*proof*⟩

lemma *vid-on-vinsert*: (*vinsert* $\langle a,\ a\rangle$ (*vid-on* A)) = (*vid-on* (*vinsert* a A))

⟨*proof*⟩

lemma *vid-on-vintersection*: *vid-on* (*A* ∩$_\circ$ *B*) = *vid-on A* ∩$_\circ$ *vid-on B* ⟨*proof*⟩

lemma *vid-on-vunion*: *vid-on* (*A* ∪$_\circ$ *B*) = *vid-on A* ∪$_\circ$ *vid-on B* ⟨*proof*⟩

lemma *vid-on-vdiff*: *vid-on* (*A* −$_\circ$ *B*) = *vid-on A* −$_\circ$ *vid-on B* ⟨*proof*⟩

Special properties.

lemma *vid-on-vsubset-vtimes*: *vid-on A* ⊆$_\circ$ *A* ×$_\circ$ *A* ⟨*proof*⟩

lemma *VLambda-id*[*simp*]: *VLambda A id* = *vid-on A*
⟨*proof*⟩

Constant function

definition *vconst-on* :: *V* ⇒ *V* ⇒ *V*
where *vconst-on A c* = *set* {⟨*a*, *c*⟩ | *a*. *a* ∈$_\circ$ *A*}

lemma *small-vconst-on*[*simp*]: *small* {⟨*a*, *c*⟩ | *a*. *a* ∈$_\circ$ *A*}
⟨*proof*⟩

Rules.

lemma *vconst-onI*[*intro*!]:
assumes *a* ∈$_\circ$ *A*
shows ⟨*a*, *c*⟩ ∈$_\circ$ *vconst-on A c*
⟨*proof*⟩

lemma *vconst-onD*[*dest*!]:
assumes ⟨*a*, *c*⟩ ∈$_\circ$ *vconst-on A c*
shows *a* ∈$_\circ$ *A*
⟨*proof*⟩

lemma *vconst-onE*[*elim*!]:
assumes *x* ∈$_\circ$ *vconst-on A c*
obtains *a* **where** *a* ∈$_\circ$ *A* **and** *x* = ⟨*a*, *c*⟩
⟨*proof*⟩

lemma *vconst-on-iff*: ⟨*a*, *c*⟩ ∈$_\circ$ *vconst-on A c* ⟷ *a* ∈$_\circ$ *A* ⟨*proof*⟩

Set operations.

lemma *vconst-on-vempty*[*simp*]: *vconst-on* 0 *c* = 0
⟨*proof*⟩

lemma *vconst-on-vsingleton*[*simp*]: *vconst-on* (*set* {*a*}) *c* = *set* {⟨*a*, *c*⟩} ⟨*proof*⟩

lemma *vconst-on-vdoubleton*[*simp*]: *vconst-on* (*set* {*a*, *b*}) *c* = *set* {⟨*a*, *c*⟩, ⟨*b*, *c*⟩}
⟨*proof*⟩

lemma *vconst-on-mono*:
assumes *A* ⊆$_\circ$ *B*
shows *vconst-on A c* ⊆$_\circ$ *vconst-on B c*
⟨*proof*⟩

lemma *vconst-on-vinsert*:
(*vinsert* ⟨*a*, *c*⟩ (*vconst-on A c*)) = (*vconst-on* (*vinsert a A*) *c*)
⟨*proof*⟩

lemma *vconst-on-vintersection*:
vconst-on $(A \cap_\circ B)$ c = *vconst-on* A $c \cap_\circ$ *vconst-on* B c
⟨*proof*⟩

lemma *vconst-on-vunion*: *vconst-on* $(A \cup_\circ B)$ c = *vconst-on* A $c \cup_\circ$ *vconst-on* B c
⟨*proof*⟩

lemma *vconst-on-vdiff*: *vconst-on* $(A -_\circ B)$ c = *vconst-on* A $c -_\circ$ *vconst-on* B c
⟨*proof*⟩

Special properties.

lemma *vconst-on-eq-vtimes*: *vconst-on* A c = $A \times_\circ$ *set* $\{c\}$
⟨*proof*⟩

VLambda

Rules.

lemma *VLambdaI*[*intro*!]:
assumes $a \in_\circ A$
shows $\langle a, f\ a\rangle \in_\circ (\lambda a \in_\circ A.\ f\ a)$
⟨*proof*⟩

lemma *VLambdaD*[*dest*!]:
assumes $\langle a, f\ a\rangle \in_\circ (\lambda a \in_\circ A.\ f\ a)$
shows $a \in_\circ A$
⟨*proof*⟩

lemma *VLambdaE*[*elim*!]:
assumes $x \in_\circ (\lambda a \in_\circ A.\ f\ a)$
obtains a **where** $a \in_\circ A$ **and** $x = \langle a, f\ a\rangle$
⟨*proof*⟩

lemma *VLambda-iff*1: $x \in_\circ (\lambda a \in_\circ A.\ f\ a) \longleftrightarrow (\exists\, a \in_\circ A.\ x = \langle a, f\ a\rangle)$ ⟨*proof*⟩

lemma *VLambda-iff*2: $\langle a, b\rangle \in_\circ (\lambda a \in_\circ A.\ f\ a) \longleftrightarrow b = f\ a \wedge a \in_\circ A$ ⟨*proof*⟩

lemma *small-VLambda*[*simp*]: *small* $\{\langle a, f\ a\rangle \mid a.\ a \in_\circ A\}$ ⟨*proof*⟩

lemma *VLambda-set-def*: $(\lambda a \in_\circ A.\ f\ a)$ = *set* $\{\langle a, f\ a\rangle \mid a.\ a \in_\circ A\}$ ⟨*proof*⟩

Set operations.

lemma *VLambda-vempty*[*simp*]: $(\lambda a \in_\circ 0.\ f\ a) = 0$ ⟨*proof*⟩

lemma *VLambda-vsingleton*: $(\lambda a \in_\circ$ *set* $\{a\}.\ f\ a)$ = *set* $\{\langle a, f\ a\rangle\}$
⟨*proof*⟩

lemma *VLambda-vdoubleton*:
$(\lambda a \in_\circ$ *set* $\{a, b\}.\ f\ a)$ = *set* $\{\langle a, f\ a\rangle, \langle b, f\ b\rangle\}$
⟨*proof*⟩

lemma *VLambda-mono*:
assumes $A \subseteq_\circ B$
shows $(\lambda a \in_\circ A.\ f\ a) \subseteq_\circ (\lambda a \in_\circ B.\ f\ a)$
⟨*proof*⟩

lemma *VLambda-vinsert*:
$(\lambda a \in_\circ$ *vinsert* a $A.\ f\ a) = (\lambda a \in_\circ$ *set* $\{a\}.\ f\ a) \cup_\circ (\lambda a \in_\circ A.\ f\ a)$

⟨*proof*⟩

lemma *VLambda-vintersection*: $(\lambda a \in_\circ A \cap_\circ B.\ f\ a) = (\lambda a \in_\circ A.\ f\ a) \cap_\circ (\lambda a \in_\circ B.\ f\ a)$
⟨*proof*⟩

lemma *VLambda-vunion*: $(\lambda a \in_\circ A \cup_\circ B.\ f\ a) = (\lambda a \in_\circ A.\ f\ a) \cup_\circ (\lambda a \in_\circ B.\ f\ a)$ ⟨*proof*⟩

lemma *VLambda-vdiff*: $(\lambda a \in_\circ A -_\circ B.\ f\ a) = (\lambda a \in_\circ A.\ f\ a) -_\circ (\lambda a \in_\circ B.\ f\ a)$ ⟨*proof*⟩

Connections.

lemma *VLambda-vid-on*: $(\lambda a \in_\circ A.\ a) = \mathit{vid\text{-}on}\ A$ ⟨*proof*⟩

lemma *VLambda-vconst-on*: $(\lambda a \in_\circ A.\ c) = \mathit{vconst\text{-}on}\ A\ c$ ⟨*proof*⟩

Composition

definition *vcomp* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\circ_\circ$› 75)
  where $r \circ_\circ s = \mathit{set}\ \{\langle a, c\rangle \mid a\ c.\ \exists b.\ \langle a, b\rangle \in_\circ s \wedge \langle b, c\rangle \in_\circ r\}$
notation *vcomp* (**infixr** ‹$\circ_\circ$› 75)

lemma *vcomp-small*[*simp*]: *small* $\{\langle a, c\rangle \mid a\ c.\ \exists b.\ \langle a, b\rangle \in_\circ s \wedge \langle b, c\rangle \in_\circ r\}$
  (**is** ‹*small ?s*›)
⟨*proof*⟩

Rules.

lemma *vcompI*[*intro*!]:
  assumes $\langle b, c\rangle \in_\circ r$ **and** $\langle a, b\rangle \in_\circ s$
  shows $\langle a, c\rangle \in_\circ r \circ_\circ s$
  ⟨*proof*⟩

lemma *vcompD*[*dest*!]:
  assumes $\langle a, c\rangle \in_\circ r \circ_\circ s$
  shows $\exists b.\ \langle b, c\rangle \in_\circ r \wedge \langle a, b\rangle \in_\circ s$
  ⟨*proof*⟩

lemma *vcompE*[*elim*!]:
  assumes $ac \in_\circ r \circ_\circ s$
  obtains $a\ b\ c$ **where** $ac = \langle a, c\rangle$ **and** $\langle a, b\rangle \in_\circ s$ **and** $\langle b, c\rangle \in_\circ r$
  ⟨*proof*⟩

Elementary properties.

lemma *vcomp-assoc*: $(r \circ_\circ s) \circ_\circ t = r \circ_\circ (s \circ_\circ t)$ ⟨*proof*⟩

Set operations.

lemma *vcomp-vempty-left*[*simp*]: $0 \circ_\circ r = 0$ ⟨*proof*⟩

lemma *vcomp-vempty-right*[*simp*]: $r \circ_\circ 0 = 0$ ⟨*proof*⟩

lemma *vcomp-mono*:
  assumes $r' \subseteq_\circ r$ **and** $s' \subseteq_\circ s$
  shows $r' \circ_\circ s' \subseteq_\circ r \circ_\circ s$
  ⟨*proof*⟩

lemma *vcomp-vinsert-left*[*simp*]:
  $(\mathit{vinsert}\ \langle a, b\rangle\ s) \circ_\circ r = (\mathit{set}\ \{\langle a, b\rangle\} \circ_\circ r) \cup_\circ (s \circ_\circ r)$
  ⟨*proof*⟩

lemma *vcomp-vinsert-right*[*simp*]:
$r \circ_\circ (\mathit{vinsert}\ \langle a,\ b\rangle\ s) = (r \circ_\circ \mathit{set}\ \{\langle a,\ b\rangle\}) \cup_\circ (r \circ_\circ s)$
⟨*proof*⟩

lemma *vcomp-vunion-left*[*simp*]: $(s \cup_\circ t) \circ_\circ r = (s \circ_\circ r) \cup_\circ (t \circ_\circ r)$ ⟨*proof*⟩

lemma *vcomp-vunion-right*[*simp*]: $r \circ_\circ (s \cup_\circ t) = (r \circ_\circ s) \cup_\circ (r \circ_\circ t)$ ⟨*proof*⟩

Connections.

lemma *vcomp-vid-on-idem*[*simp*]: *vid-on A* $\circ_\circ$ *vid-on A* = *vid-on A* ⟨*proof*⟩

lemma *vcomp-vid-on*[*simp*]: *vid-on A* $\circ_\circ$ *vid-on B* = *vid-on* $(A \cap_\circ B)$ ⟨*proof*⟩

lemma *vcomp-vconst-on-vid-on*[*simp*]: *vconst-on A c* $\circ_\circ$ *vid-on A* = *vconst-on A c*
⟨*proof*⟩

lemma *vcomp-VLambda-vid-on*[*simp*]: $(\lambda a{\in_\circ} A.\ f\ a) \circ_\circ$ *vid-on A* = $(\lambda a{\in_\circ} A.\ f\ a)$
⟨*proof*⟩

Special properties.

lemma *vcomp-vsubset-vtimes*:
assumes $r \subseteq_\circ B \times_\circ C$ **and** $s \subseteq_\circ A \times_\circ B$
shows $r \circ_\circ s \subseteq_\circ A \times_\circ C$
⟨*proof*⟩

lemma *vcomp-obtain-middle*[*elim*]:
assumes $\langle a,\ c\rangle \in_\circ r \circ_\circ s$
obtains b **where** $\langle a,\ b\rangle \in_\circ s$ **and** $\langle b,\ c\rangle \in_\circ r$
⟨*proof*⟩

Converse relation

definition *vconverse* :: $V \Rightarrow V$
where *vconverse A* = $(\lambda r{\in_\circ} A.\ \mathit{set}\ \{\langle b,\ a\rangle \mid a\ b.\ \langle a,\ b\rangle \in_\circ r\})$

abbreviation *app-vconverse* $(\langle(\text{-}^{-1}{}_\circ)\rangle\ [1000]\ 999)$
where $r^{-1}{}_\circ \equiv \mathit{vconverse}\ (\mathit{set}\ \{r\})\ (\lfloor r \rfloor)$

lemma *app-vconverse-def*: $r^{-1}{}_\circ = \mathit{set}\ \{\langle b,\ a\rangle \mid a\ b.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vconverse-small*[*simp*]: *small* $\{\langle b,\ a\rangle \mid a\ b.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *vconverseI*[*intro*!]:
assumes $r \in_\circ A$
shows $\langle r,\ r^{-1}{}_\circ\rangle \in_\circ \mathit{vconverse}\ A$
⟨*proof*⟩

lemma *vconverseD*[*dest*]:
assumes $\langle r,\ s\rangle \in_\circ \mathit{vconverse}\ A$
shows $r \in_\circ A$ **and** $s = r^{-1}{}_\circ$
⟨*proof*⟩

lemma *vconverseE*[*elim*]:
assumes $x \in_\circ \mathit{vconverse}\ A$
obtains r **where** $x = \langle r,\ r^{-1}{}_\circ\rangle$ **and** $r \in_\circ A$

⟨*proof*⟩

lemma *app-vconverseI*[*sym*, *intro*!]:
assumes ⟨*a*, *b*⟩ ∈$_\circ$ *r*
shows ⟨*b*, *a*⟩ ∈$_\circ$ *r*$^{-1}{}_\circ$
⟨*proof*⟩

lemma *app-vconverseD*[*sym*, *dest*]:
assumes ⟨*a*, *b*⟩ ∈$_\circ$ *r*$^{-1}{}_\circ$
shows ⟨*b*, *a*⟩ ∈$_\circ$ *r*
⟨*proof*⟩

lemma *app-vconverseE*[*elim*!]:
assumes *x* ∈$_\circ$ *r*$^{-1}{}_\circ$
obtains *a b* **where** *x* = ⟨*b*, *a*⟩ **and** ⟨*a*, *b*⟩ ∈$_\circ$ *r*
⟨*proof*⟩

lemma *vconverse-iff*: ⟨*b*, *a*⟩ ∈$_\circ$ *r*$^{-1}{}_\circ$ ⟷ ⟨*a*, *b*⟩ ∈$_\circ$ *r* ⟨*proof*⟩

Set operations.

lemma *vconverse-vempty*[*simp*]: $0^{-1}{}_\circ$ = 0 ⟨*proof*⟩

lemma *vconverse-vsingleton*: (*set* {⟨*a*, *b*⟩})$^{-1}{}_\circ$ = *set* {⟨*b*, *a*⟩} ⟨*proof*⟩

lemma *vconverse-vdoubleton*[*simp*]: (*set* {⟨*a*, *b*⟩, ⟨*c*, *d*⟩})$^{-1}{}_\circ$ = *set* {⟨*b*, *a*⟩, ⟨*d*, *c*⟩}
⟨*proof*⟩

lemma *vconverse-vinsert*: (*vinsert* ⟨*a*, *b*⟩ *r*)$^{-1}{}_\circ$ = *vinsert* ⟨*b*, *a*⟩ (*r*$^{-1}{}_\circ$) ⟨*proof*⟩

lemma *vconverse-vintersection*: (*r* ∩$_\circ$ *s*)$^{-1}{}_\circ$ = *r*$^{-1}{}_\circ$ ∩$_\circ$ *s*$^{-1}{}_\circ$ ⟨*proof*⟩

lemma *vconverse-vunion*: (*r* ∪$_\circ$ *s*)$^{-1}{}_\circ$ = *r*$^{-1}{}_\circ$ ∪$_\circ$ *s*$^{-1}{}_\circ$ ⟨*proof*⟩

Connections.

lemma *vconverse-vid-on*[*simp*]: (*vid-on A*)$^{-1}{}_\circ$ = *vid-on A* ⟨*proof*⟩

lemma *vconverse-vconst-on*[*simp*]: (*vconst-on A c*)$^{-1}{}_\circ$ = *set* {*c*} ×$_\circ$ *A* ⟨*proof*⟩

lemma *vconverse-vcomp*: (*r* ∘$_\circ$ *s*)$^{-1}{}_\circ$ = *s*$^{-1}{}_\circ$ ∘$_\circ$ *r*$^{-1}{}_\circ$ ⟨*proof*⟩

lemma *vconverse-vtimes*: (*A* ×$_\circ$ *B*)$^{-1}{}_\circ$ = (*B* ×$_\circ$ *A*) ⟨*proof*⟩

Left restriction

definition *vlrestriction* :: *V* ⇒ *V*
where *vlrestriction D* =
VLambda D (λ⟨*r*, *A*⟩. *set* {⟨*a*, *b*⟩ | *a b*. *a* ∈$_\circ$ *A* ∧ ⟨*a*, *b*⟩ ∈$_\circ$ *r*})

abbreviation *app-vlrestriction* :: *V* ⇒ *V* ⇒ *V* (**infixr** ‹↾$^l{}_\circ$› 80)
where *r* ↾$^l{}_\circ$ *A* ≡ *vlrestriction* (*set* {⟨*r*, *A*⟩}) (‖⟨*r*, *A*⟩‖)

lemma *app-vlrestriction-def*: *r* ↾$^l{}_\circ$ *A* = *set* {⟨*a*, *b*⟩ | *a b*. *a* ∈$_\circ$ *A* ∧ ⟨*a*, *b*⟩ ∈$_\circ$ *r*}
⟨*proof*⟩

lemma *vlrestriction-small*[*simp*]: *small* {⟨*a*, *b*⟩ | *a b*. *a* ∈$_\circ$ *A* ∧ ⟨*a*, *b*⟩ ∈$_\circ$ *r*}
⟨*proof*⟩

Rules.

lemma *vlrestrictionI*[*intro*!]:
assumes $\langle r,\ A\rangle \in_\circ D$
shows $\langle\langle r,\ A\rangle,\ r \upharpoonright^l_\circ A\rangle \in_\circ$ *vlrestriction* D
⟨*proof*⟩

lemma *vlrestrictionD*[*dest*]:
assumes $\langle\langle r,\ A\rangle,\ s\rangle \in_\circ$ *vlrestriction* D
shows $\langle r,\ A\rangle \in_\circ D$ **and** $s = r \upharpoonright^l_\circ A$
⟨*proof*⟩

lemma *vlrestrictionE*[*elim*]:
assumes $x \in_\circ$ *vlrestriction* D **and** $D \subseteq_\circ R \times_\circ X$
obtains $r\ A$ **where** $x = \langle\langle r,\ A\rangle,\ r \upharpoonright^l_\circ A\rangle$ **and** $r \in_\circ R$ **and** $A \in_\circ X$
⟨*proof*⟩

lemma *app-vlrestrictionI*[*intro*!]:
assumes $a \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
shows $\langle a,\ b\rangle \in_\circ r \upharpoonright^l_\circ A$
⟨*proof*⟩

lemma *app-vlrestrictionD*[*dest*]:
assumes $\langle a,\ b\rangle \in_\circ r \upharpoonright^l_\circ A$
shows $a \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *app-vlrestrictionE*[*elim*]:
assumes $x \in_\circ r \upharpoonright^l_\circ A$
obtains $a\ b$ **where** $x = \langle a,\ b\rangle$ **and** $a \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *vlrestriction-on-vempty*[*simp*]: $r \upharpoonright^l_\circ 0 = 0$
⟨*proof*⟩

lemma *vlrestriction-vempty*[*simp*]: $0 \upharpoonright^l_\circ A = 0$ ⟨*proof*⟩

lemma *vlrestriction-vsingleton-in*[*simp*]:
assumes $a \in_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \upharpoonright^l_\circ A =$ *set* $\{\langle a,\ b\rangle\}$
⟨*proof*⟩

lemma *vlrestriction-vsingleton-nin*[*simp*]:
assumes $a \notin_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \upharpoonright^l_\circ A = 0$
⟨*proof*⟩

lemma *vlrestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \upharpoonright^l_\circ A \subseteq_\circ r \upharpoonright^l_\circ B$
⟨*proof*⟩

lemma *vlrestriction-vinsert-nin*[*simp*]:
assumes $a \notin_\circ A$
shows $(\mathit{vinsert}\ \langle a,\ b\rangle\ r) \upharpoonright^l_\circ A = r \upharpoonright^l_\circ A$
⟨*proof*⟩

lemma *vlrestriction-vinsert-in*:
assumes $a \in_\circ A$

shows $(vinsert\ \langle a,\ b\rangle\ r)\ \restriction^{l}_{\circ}\ A = vinsert\ \langle a,\ b\rangle\ (r\ \restriction^{l}_{\circ}\ A)$
⟨*proof*⟩

lemma *vlrestriction-vintersection*: $(r \cap_{\circ} s)\ \restriction^{l}_{\circ}\ A = r\ \restriction^{l}_{\circ}\ A \cap_{\circ} s\ \restriction^{l}_{\circ}\ A$ ⟨*proof*⟩

lemma *vlrestriction-vunion*: $(r \cup_{\circ} s)\ \restriction^{l}_{\circ}\ A = r\ \restriction^{l}_{\circ}\ A \cup_{\circ} s\ \restriction^{l}_{\circ}\ A$ ⟨*proof*⟩

lemma *vlrestriction-vdiff*: $(r -_{\circ} s)\ \restriction^{l}_{\circ}\ A = r\ \restriction^{l}_{\circ}\ A -_{\circ} s\ \restriction^{l}_{\circ}\ A$ ⟨*proof*⟩

Connections.

lemma *vlrestriction-vid-on*[*simp*]: $(vid\text{-}on\ A)\ \restriction^{l}_{\circ}\ B = vid\text{-}on\ (A \cap_{\circ} B)$ ⟨*proof*⟩

lemma *vlrestriction-vconst-on*: $(vconst\text{-}on\ A\ c)\ \restriction^{l}_{\circ}\ B = (vconst\text{-}on\ B\ c)\ \restriction^{l}_{\circ}\ A$
⟨*proof*⟩

lemma *vlrestriction-vconst-on-commute*:
assumes $x \in_{\circ} vconst\text{-}on\ A\ c\ \restriction^{l}_{\circ}\ B$
shows $x \in_{\circ} vconst\text{-}on\ B\ c\ \restriction^{l}_{\circ}\ A$
⟨*proof*⟩

lemma *vlrestriction-vcomp*[*simp*]: $(r \circ_{\circ} s)\ \restriction^{l}_{\circ}\ A = r \circ_{\circ} (s\ \restriction^{l}_{\circ}\ A)$ ⟨*proof*⟩

Previous connections.

lemma *vcomp-rel-vid-on*[*simp*]: $r \circ_{\circ} vid\text{-}on\ A = r\ \restriction^{l}_{\circ}\ A$ ⟨*proof*⟩

lemma *vcomp-vconst-on*:
$r \circ_{\circ} (vconst\text{-}on\ A\ c) = (r\ \restriction^{l}_{\circ}\ set\ \{c\}) \circ_{\circ} (vconst\text{-}on\ A\ c)$
⟨*proof*⟩

Special properties.

lemma *vlrestriction-vsubset-vpairs*: $r\ \restriction^{l}_{\circ}\ A \subseteq_{\circ} vpairs\ r$
⟨*proof*⟩

lemma *vlrestriction-vsubset-rel*: $r\ \restriction^{l}_{\circ}\ A \subseteq_{\circ} r$ ⟨*proof*⟩

lemma *vlrestriction-VLambda*: $(\lambda a{\in_{\circ}}A.\ f\ a)\ \restriction^{l}_{\circ}\ B = (\lambda a{\in_{\circ}}A \cap_{\circ} B.\ f\ a)$ ⟨*proof*⟩

Right restriction

definition *vrrestriction* :: $V \Rightarrow V$
where *vrrestriction D* =
$VLambda\ D\ (\lambda\langle r,\ A\rangle.\ set\ \{\langle a,\ b\rangle \mid a\ b.\ b \in_{\circ} A \wedge \langle a,\ b\rangle \in_{\circ} r\})$

abbreviation *app-vrrestriction* :: $V \Rightarrow V \Rightarrow V$ (**infixr** $\langle\restriction^{r}_{\circ}\rangle$ 80)
where $r\ \restriction^{r}_{\circ}\ A \equiv vrrestriction\ (set\ \{\langle r,\ A\rangle\})\ (\!|\langle r,\ A\rangle|\!)$

lemma *app-vrrestriction-def*: $r\ \restriction^{r}_{\circ}\ A = set\ \{\langle a,\ b\rangle \mid a\ b.\ b \in_{\circ} A \wedge \langle a,\ b\rangle \in_{\circ} r\}$
⟨*proof*⟩

lemma *vrrestriction-small*[*simp*]: $small\ \{\langle a,\ b\rangle \mid a\ b.\ b \in_{\circ} A \wedge \langle a,\ b\rangle \in_{\circ} r\}$
⟨*proof*⟩

Rules.

lemma *vrrestrictionI*[*intro*!]:
assumes $\langle r,\ A\rangle \in_{\circ} D$
shows $\langle\langle r,\ A\rangle,\ r\ \restriction^{r}_{\circ}\ A\rangle \in_{\circ} vrrestriction\ D$
⟨*proof*⟩

lemma *vrrestrictionD*[*dest*]:
assumes $\langle\langle r,\ A\rangle,\ s\rangle \in_\circ$ *vrrestriction* D
shows $\langle r,\ A\rangle \in_\circ D$ **and** $s = r \restriction^r_\circ A$
⟨*proof*⟩

lemma *vrrestrictionE*[*elim*]:
assumes $x \in_\circ$ *vrrestriction* D **and** $D \subseteq_\circ R \times_\circ X$
obtains $r\ A$ **where** $x = \langle\langle r,\ A\rangle,\ r \restriction^r_\circ A\rangle$ **and** $r \in_\circ R$ **and** $A \in_\circ X$
⟨*proof*⟩

lemma *app-vrrestrictionI*[*intro*!]:
assumes $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
shows $\langle a,\ b\rangle \in_\circ r \restriction^r_\circ A$
⟨*proof*⟩

lemma *app-vrrestrictionD*[*dest*]:
assumes $\langle a,\ b\rangle \in_\circ r \restriction^r_\circ A$
shows $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *app-vrrestrictionE*[*elim*]:
assumes $x \in_\circ r \restriction^r_\circ A$
obtains $a\ b$ **where** $x = \langle a,\ b\rangle$ **and** $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *vrrestriction-on-vempty*[*simp*]: $r \restriction^r_\circ 0 = 0$
⟨*proof*⟩

lemma *vrrestriction-vempty*[*simp*]: $0 \restriction^r_\circ A = 0$ ⟨*proof*⟩

lemma *vrrestriction-vsingleton-in*[*simp*]:
assumes $b \in_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \restriction^r_\circ A =$ *set* $\{\langle a,\ b\rangle\}$
⟨*proof*⟩

lemma *vrrestriction-vsingleton-nin*[*simp*]:
assumes $b \notin_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \restriction^r_\circ A = 0$
⟨*proof*⟩

lemma *vrrestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \restriction^r_\circ A \subseteq_\circ r \restriction^r_\circ B$
⟨*proof*⟩

lemma *vrrestriction-vinsert-nin*[*simp*]:
assumes $b \notin_\circ A$
shows $(\mathit{vinsert}\ \langle a,\ b\rangle\ r) \restriction^r_\circ A = r \restriction^r_\circ A$
⟨*proof*⟩

lemma *vrrestriction-vinsert-in*:
assumes $b \in_\circ A$
shows $(\mathit{vinsert}\ \langle a,\ b\rangle\ r) \restriction^r_\circ A = \mathit{vinsert}\ \langle a,\ b\rangle\ (r \restriction^r_\circ A)$
⟨*proof*⟩

lemma *vrrestriction-vintersection*: $(r \cap_\circ s) \restriction^r_\circ A = r \restriction^r_\circ A \cap_\circ s \restriction^r_\circ A$ ⟨*proof*⟩

lemma *vrrestriction-vunion*: $(r \cup_\circ s) \restriction^r_\circ A = r \restriction^r_\circ A \cup_\circ s \restriction^r_\circ A$ ⟨*proof*⟩

lemma *vrrestriction-vdiff*: $(r -_\circ s) \restriction^r_\circ A = r \restriction^r_\circ A -_\circ s \restriction^r_\circ A$ ⟨*proof*⟩

Connections.

lemma *vrrestriction-vid-on*[*simp*]: $(\textit{vid-on}\ A) \restriction^r_\circ B = \textit{vid-on}\ (A \cap_\circ B)$ ⟨*proof*⟩

lemma *vrrestriction-vconst-on*:
assumes $c \in_\circ B$
shows $(\textit{vconst-on}\ A\ c) \restriction^r_\circ B = \textit{vconst-on}\ A\ c$
⟨*proof*⟩

lemma *vrrestriction-vcomp*[*simp*]: $(r \circ_\circ s) \restriction^r_\circ A = (r \restriction^r_\circ A) \circ_\circ s$ ⟨*proof*⟩

Previous connections.

lemma *vcomp-vid-on-rel*[*simp*]: $\textit{vid-on}\ A \circ_\circ r = r \restriction^r_\circ A$
⟨*proof*⟩

lemma *vcomp-vconst-on-rel*: $(\textit{vconst-on}\ A\ c) \circ_\circ r = (\textit{vconst-on}\ A\ c) \circ_\circ (r \restriction^r_\circ A)$
⟨*proof*⟩

lemma *vlrestriction-vconverse*: $r^{-1}_\circ \restriction^l_\circ A = (r \restriction^r_\circ A)^{-1}_\circ$ ⟨*proof*⟩

lemma *vrrestriction-vconverse*: $r^{-1}_\circ \restriction^r_\circ A = (r \restriction^l_\circ A)^{-1}_\circ$ ⟨*proof*⟩

Special properties.

lemma *vrrestriction-vsubset-rel*: $r \restriction^r_\circ A \subseteq_\circ r$ ⟨*proof*⟩

lemma *vrrestriction-vsubset-vpairs*: $r \restriction^r_\circ A \subseteq_\circ \textit{vpairs}\ r$ ⟨*proof*⟩

Restriction

definition *vrestriction* :: $V \Rightarrow V$
where *vrestriction D* =
VLambda D $(\lambda\langle r, A\rangle.\ \textit{set}\ \{\langle a, b\rangle \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge \langle a, b\rangle \in_\circ r\})$

abbreviation *app-vrestriction* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\restriction_\circ$› 80)
where $r \restriction_\circ A \equiv \textit{vrestriction}\ (\textit{set}\ \{\langle r, A\rangle\})\ (\!|\langle r, A\rangle|\!)$

lemma *app-vrestriction-def*:
$r \restriction_\circ A = \textit{set}\ \{\langle a, b\rangle \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge \langle a, b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vrestriction-small*[*simp*]:
$\textit{small}\ \{\langle a, b\rangle \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge \langle a, b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *vrestrictionI*[*intro*!]:
assumes $\langle r, A\rangle \in_\circ D$
shows $\langle\langle r, A\rangle, r \restriction_\circ A\rangle \in_\circ \textit{vrestriction}\ D$
⟨*proof*⟩

lemma *vrestrictionD*[*dest*]:
assumes $\langle\langle r, A\rangle, s\rangle \in_\circ \textit{vrestriction}\ D$
shows $\langle r, A\rangle \in_\circ D$ **and** $s = r \restriction_\circ A$

⟨*proof*⟩

lemma *vrestrictionE*[*elim*]:
assumes $x \in_\circ$ *vrestriction* D **and** $D \subseteq_\circ R \times_\circ X$
obtains $r\ A$ **where** $x = \langle\langle r,\ A\rangle,\ r \upharpoonright_\circ A\rangle$ **and** $r \in_\circ R$ **and** $A \in_\circ X$
⟨*proof*⟩

lemma *app-vrestrictionI*[*intro*!]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
shows $\langle a,\ b\rangle \in_\circ r \upharpoonright_\circ A$
⟨*proof*⟩

lemma *app-vrestrictionD*[*dest*]:
assumes $\langle a,\ b\rangle \in_\circ r \upharpoonright_\circ A$
shows $a \in_\circ A$ **and** $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *app-vrestrictionE*[*elim*]:
assumes $x \in_\circ r \upharpoonright_\circ A$
obtains $a\ b$ **where** $x = \langle a,\ b\rangle$ **and** $a \in_\circ A$ **and** $b \in_\circ A$ **and** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *vrestriction-on-vempty*[*simp*]: $r \upharpoonright_\circ 0 = 0$
⟨*proof*⟩

lemma *vrestriction-vempty*[*simp*]: $0 \upharpoonright_\circ A = 0$ ⟨*proof*⟩

lemma *vrestriction-vsingleton-in*[*simp*]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \upharpoonright_\circ A =$ *set* $\{\langle a,\ b\rangle\}$
⟨*proof*⟩

lemma *vrestriction-vsingleton-nin-left*[*simp*]:
assumes $a \notin_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \upharpoonright_\circ A = 0$
⟨*proof*⟩

lemma *vrestriction-vsingleton-nin-right*[*simp*]:
assumes $b \notin_\circ A$
shows *set* $\{\langle a,\ b\rangle\} \upharpoonright_\circ A = 0$
⟨*proof*⟩

lemma *vrestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \upharpoonright_\circ A \subseteq_\circ r \upharpoonright_\circ B$
⟨*proof*⟩

lemma *vrestriction-vinsert-nin*[*simp*]:
assumes $a \notin_\circ A$ **and** $b \notin_\circ A$
shows $(\mathit{vinsert}\ \langle a,\ b\rangle\ r) \upharpoonright_\circ A = r \upharpoonright_\circ A$
⟨*proof*⟩

lemma *vrestriction-vinsert-in*:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows $(\mathit{vinsert}\ \langle a,\ b\rangle\ r) \upharpoonright_\circ A = \mathit{vinsert}\ \langle a,\ b\rangle\ (r \upharpoonright_\circ A)$
⟨*proof*⟩

lemma *vrestriction-vintersection*: $(r \cap_\circ s) \restriction_\circ A = r \restriction_\circ A \cap_\circ s \restriction_\circ A$ ⟨*proof*⟩

lemma *vrestriction-vunion*: $(r \cup_\circ s) \restriction_\circ A = r \restriction_\circ A \cup_\circ s \restriction_\circ A$ ⟨*proof*⟩

lemma *vrestriction-vdiff*: $(r -_\circ s) \restriction_\circ A = r \restriction_\circ A -_\circ s \restriction_\circ A$ ⟨*proof*⟩

Connections.

lemma *vrestriction-vid-on*[*simp*]: $(\textit{vid-on}\ A) \restriction_\circ B = \textit{vid-on}\ (A \cap_\circ B)$ ⟨*proof*⟩

lemma *vrestriction-vconst-on-ex*:
assumes $c \in_\circ B$
shows $(\textit{vconst-on}\ A\ c) \restriction_\circ B = \textit{vconst-on}\ (A \cap_\circ B)\ c$
⟨*proof*⟩

lemma *vrestriction-vconst-on-nex*:
assumes $c \notin_\circ B$
shows $(\textit{vconst-on}\ A\ c) \restriction_\circ B = 0$
⟨*proof*⟩

lemma *vrestriction-vcomp*[*simp*]: $(r \circ_\circ s) \restriction_\circ A = (r \restriction^r_\circ A) \circ_\circ (s \restriction^l_\circ A)$ ⟨*proof*⟩

lemma *vrestriction-vconverse*: $r^{-1}{}_\circ \restriction_\circ A = (r \restriction_\circ A)^{-1}{}_\circ$ ⟨*proof*⟩

Previous connections.

lemma *vrrestriction-vlrestriction*[*simp*]: $(r \restriction^r_\circ A) \restriction^l_\circ A = r \restriction_\circ A$ ⟨*proof*⟩

lemma *vlrestriction-vrrestriction*[*simp*]: $(r \restriction^l_\circ A) \restriction^r_\circ A = r \restriction_\circ A$ ⟨*proof*⟩

lemma *vrestriction-vlrestriction*[*simp*]: $(r \restriction_\circ A) \restriction^l_\circ A = r \restriction_\circ A$ ⟨*proof*⟩

lemma *vrestriction-vrrestriction*[*simp*]: $(r \restriction_\circ A) \restriction^r_\circ A = r \restriction_\circ A$ ⟨*proof*⟩

Special properties.

lemma *vrestriction-vsubset-vpairs*: $r \restriction_\circ A \subseteq_\circ \textit{vpairs}\ r$ ⟨*proof*⟩

lemma *vrestriction-vsubset-vtimes*: $r \restriction_\circ A \subseteq_\circ A \times_\circ A$ ⟨*proof*⟩

lemma *vrestriction-vsubset-rel*: $r \restriction_\circ A \subseteq_\circ r$ ⟨*proof*⟩

2.4.3 Properties

Domain

definition *vdomain* :: $V \Rightarrow V$
where $\textit{vdomain}\ D = (\lambda r \in_\circ D.\ \textit{set}\ \{a.\ \exists b.\ \langle a, b\rangle \in_\circ r\})$

abbreviation *app-vdomain* :: $V \Rightarrow V$ $(\langle\mathcal{D}_\circ\rangle)$
where $\mathcal{D}_\circ\ r \equiv \textit{vdomain}\ (\textit{set}\ \{r\})\ (\lVert r \rVert)$

lemma *app-vdomain-def*: $\mathcal{D}_\circ\ r = \textit{set}\ \{a.\ \exists b.\ \langle a, b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vdomain-small*[*simp*]: $\textit{small}\ \{a.\ \exists b.\ \langle a, b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *vdomainI*[*intro*!]:
assumes $r \in_\circ A$

shows $\langle r,\ \mathcal{D}_\circ\ r\rangle \in_\circ vdomain\ A$
⟨*proof*⟩

lemma *vdomainD*[*dest*]:
assumes $\langle r,\ s\rangle \in_\circ vdomain\ A$
shows $r \in_\circ A$ **and** $s = \mathcal{D}_\circ\ r$
⟨*proof*⟩

lemma *vdomainE*[*elim*]:
assumes $x \in_\circ vdomain\ A$
obtains r **where** $x = \langle r,\ \mathcal{D}_\circ\ r\rangle$ **and** $r \in_\circ A$
⟨*proof*⟩

lemma *app-vdomainI*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$
shows $a \in_\circ \mathcal{D}_\circ\ r$
⟨*proof*⟩

lemma *app-vdomainD*[*dest*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $\exists\, b.\ \langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *app-vdomainE*[*elim*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
obtains b **where** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vdomain-iff*: $a \in_\circ \mathcal{D}_\circ\ r \longleftrightarrow (\exists\, y.\ \langle a,\ y\rangle \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *vdomain-vempty*[*simp*]: $\mathcal{D}_\circ\ 0 = 0$ ⟨*proof*⟩

lemma *vdomain-vsingleton*[*simp*]: $\mathcal{D}_\circ\ (set\ \{\langle a,\ b\rangle\}) = set\ \{a\}$ ⟨*proof*⟩

lemma *vdomain-vdoubleton*[*simp*]: $\mathcal{D}_\circ\ (set\ \{\langle a,\ b\rangle,\ \langle c,\ d\rangle\}) = set\ \{a,\ c\}$
⟨*proof*⟩

lemma *vdomain-mono*:
assumes $r \subseteq_\circ s$
shows $\mathcal{D}_\circ\ r \subseteq_\circ \mathcal{D}_\circ\ s$
⟨*proof*⟩

lemma *vdomain-vinsert*[*simp*]: $\mathcal{D}_\circ\ (vinsert\ \langle a,\ b\rangle\ r) = vinsert\ a\ (\mathcal{D}_\circ\ r)$
⟨*proof*⟩

lemma *vdomain-vunion*: $\mathcal{D}_\circ\ (A \cup_\circ B) = \mathcal{D}_\circ\ A \cup_\circ \mathcal{D}_\circ\ B$
⟨*proof*⟩

lemma *vdomain-vintersection-vsubset*: $\mathcal{D}_\circ\ (A \cap_\circ B) \subseteq_\circ \mathcal{D}_\circ\ A \cap_\circ \mathcal{D}_\circ\ B$ ⟨*proof*⟩

lemma *vdomain-vdiff-vsubset*: $\mathcal{D}_\circ\ A -_\circ \mathcal{D}_\circ\ B \subseteq_\circ \mathcal{D}_\circ\ (A -_\circ B)$ ⟨*proof*⟩

Connections.

lemma *vdomain-vid-on*[*simp*]: $\mathcal{D}_\circ\ (vid\text{-}on\ A) = A$
⟨*proof*⟩

lemma *vdomain-vconst-on*[*simp*]: $\mathcal{D}_\circ\ (vconst\text{-}on\ A\ c) = A$

⟨*proof*⟩

lemma *vdomain-VLambda*[*simp*]: $\mathcal{D}_\circ\ (\lambda a \in_\circ A.\ f\ a) = A$
⟨*proof*⟩

lemma *vdomain-vlrestriction*: $\mathcal{D}_\circ\ (r \restriction^l_\circ A) = \mathcal{D}_\circ\ r \cap_\circ A$ ⟨*proof*⟩

lemma *vdomain-vlrestriction-vsubset*:
assumes $A \subseteq_\circ \mathcal{D}_\circ\ r$
shows $\mathcal{D}_\circ\ (r \restriction^l_\circ A) = A$
⟨*proof*⟩

Special properties.

lemma *vdomain-vsubset-vtimes*:
assumes *vpairs* $r \subseteq_\circ x \times_\circ y$
shows $\mathcal{D}_\circ\ r \subseteq_\circ x$
⟨*proof*⟩

Range

definition *vrange* :: $V \Rightarrow V$
where *vrange* $D = (\lambda r \in_\circ D.\ set\ \{b.\ \exists\, a.\ \langle a,\ b\rangle \in_\circ r\})$

abbreviation *app-vrange* :: $V \Rightarrow V$ $(\langle\mathcal{R}_\circ\rangle)$
where $\mathcal{R}_\circ\ r \equiv vrange\ (set\ \{r\})\ (\lfloor r \rfloor)$

lemma *app-vrange-def*: $\mathcal{R}_\circ\ r = set\ \{b.\ \exists\, a.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vrange-small*[*simp*]: *small* $\{b.\ \exists\, a.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *vrangeI*[*intro*]:
assumes $r \in_\circ A$
shows $\langle r,\ \mathcal{R}_\circ\ r\rangle \in_\circ vrange\ A$
⟨*proof*⟩

lemma *vrangeD*[*dest*]:
assumes $\langle r,\ s\rangle \in_\circ vrange\ A$
shows $r \in_\circ A$ **and** $s = \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vrangeE*[*elim*]:
assumes $x \in_\circ vrange\ A$
obtains r **where** $x = \langle r,\ \mathcal{R}_\circ\ r\rangle$ **and** $r \in_\circ A$
⟨*proof*⟩

lemma *app-vrangeI*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$
shows $b \in_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *app-vrangeD*[*dest*]:
assumes $b \in_\circ \mathcal{R}_\circ\ r$
shows $\exists\, a.\ \langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *app-vrangeE*[*elim*]:
assumes $b \in_\circ \mathcal{R}_\circ\ r$
obtains a **where** $\langle a,\ b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vrange-iff*: $b \in_\circ \mathcal{R}_\circ\ r \longleftrightarrow (\exists\, a.\ \langle a,\ b\rangle \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *vrange-vempty*[*simp*]: $\mathcal{R}_\circ\ 0 = 0$ ⟨*proof*⟩

lemma *vrange-vsingleton*[*simp*]: $\mathcal{R}_\circ\ (set\ \{\langle a,\ b\rangle\}) = set\ \{b\}$ ⟨*proof*⟩

lemma *vrange-vdoubleton*[*simp*]: $\mathcal{R}_\circ\ (set\ \{\langle a,\ b\rangle,\ \langle c,\ d\rangle\}) = set\ \{b,\ d\}$
⟨*proof*⟩

lemma *vrange-mono*:
assumes $r \subseteq_\circ s$
shows $\mathcal{R}_\circ\ r \subseteq_\circ \mathcal{R}_\circ\ s$
⟨*proof*⟩

lemma *vrange-vinsert*[*simp*]: $\mathcal{R}_\circ\ (vinsert\ \langle a,\ b\rangle\ r) = vinsert\ b\ (\mathcal{R}_\circ\ r)$
⟨*proof*⟩

lemma *vrange-vunion*: $\mathcal{R}_\circ\ (r \cup_\circ s) = \mathcal{R}_\circ\ r \cup_\circ \mathcal{R}_\circ\ s$
⟨*proof*⟩

lemma *vrange-vintersection-vsubset*: $\mathcal{R}_\circ\ (r \cap_\circ s) \subseteq_\circ \mathcal{R}_\circ\ r \cap_\circ \mathcal{R}_\circ\ s$ ⟨*proof*⟩

lemma *vrange-vdiff-vsubset*: $\mathcal{R}_\circ\ r -_\circ \mathcal{R}_\circ\ s \subseteq_\circ \mathcal{R}_\circ\ (r -_\circ s)$ ⟨*proof*⟩

Connections.

lemma *vrange-vid-on*[*simp*]: $\mathcal{R}_\circ\ (vid\text{-}on\ A) = A$ ⟨*proof*⟩

lemma *vrange-vconst-on-vempty*[*simp*]: $\mathcal{R}_\circ\ (vconst\text{-}on\ 0\ c) = 0$ ⟨*proof*⟩

lemma *vrange-vconst-on-ne*[*simp*]:
assumes $A \neq 0$
shows $\mathcal{R}_\circ\ (vconst\text{-}on\ A\ c) = set\ \{c\}$
⟨*proof*⟩

lemma *vrange-VLambda*: $\mathcal{R}_\circ\ (\lambda a{\in_\circ}A.\ f\ a) = set\ (f\ `\ elts\ A)$
⟨*proof*⟩

lemma *vrange-vrrestriction*: $\mathcal{R}_\circ\ (r \restriction^r_\circ A) = \mathcal{R}_\circ\ r \cap_\circ A$ ⟨*proof*⟩

Previous connections

lemma *vdomain-vconverse*[*simp*]: $\mathcal{D}_\circ\ ({r^{-1}}_\circ) = \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vrange-vconverse*[*simp*]: $\mathcal{R}_\circ\ ({r^{-1}}_\circ) = \mathcal{D}_\circ\ r$
⟨*proof*⟩

Special properties.

lemma *vrange-iff-vdomain*: $b \in_\circ \mathcal{R}_\circ\ r \longleftrightarrow (\exists\, a{\in_\circ}\mathcal{D}_\circ\ r.\ \langle a,\ b\rangle \in_\circ r)$ ⟨*proof*⟩

lemma *vrange-vsubset-vtimes*:
assumes $vpairs\ r \subseteq_\circ x \times_\circ y$

shows $\mathcal{R}_\circ\ r \subseteq_\circ y$
⟨*proof*⟩

lemma *vrange-VLambda-vsubset*:
assumes $\bigwedge x.\ x \in_\circ A \Longrightarrow f\ x \in_\circ B$
shows $\mathcal{R}_\circ\ (VLambda\ A\ f) \subseteq_\circ B$
⟨*proof*⟩

lemma *vpairs-vsubset-vdomain-vrange*[*simp*]: $vpairs\ r \subseteq_\circ \mathcal{D}_\circ\ r \times_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vrange-vsubset*:
assumes $\bigwedge x\ y.\ \langle x,\ y\rangle \in_\circ r \Longrightarrow y \in_\circ A$
shows $\mathcal{R}_\circ\ r \subseteq_\circ A$
⟨*proof*⟩

Field

definition *vfield* :: $V \Rightarrow V$
where $vfield\ D = (\lambda r \in_\circ D.\ \mathcal{D}_\circ\ r \cup_\circ \mathcal{R}_\circ\ r)$

abbreviation *app-vfield* :: $V \Rightarrow V$ $(\langle\mathcal{F}_\circ\rangle)$
where $\mathcal{F}_\circ\ r \equiv vfield\ (set\ \{r\})\ (\lfloor r \rfloor)$

lemma *app-vfield-def*: $\mathcal{F}_\circ\ r = \mathcal{D}_\circ\ r \cup_\circ \mathcal{R}_\circ\ r$ ⟨*proof*⟩

Rules.

lemma *vfieldI*[*intro*!]:
assumes $r \in_\circ A$
shows $\langle r,\ \mathcal{F}_\circ\ r\rangle \in_\circ vfield\ A$
⟨*proof*⟩

lemma *vfieldD*[*dest*]:
assumes $\langle r,\ s\rangle \in_\circ vfield\ A$
shows $r \in_\circ A$ **and** $s = \mathcal{F}_\circ\ r$
⟨*proof*⟩

lemma *vfieldE*[*elim*]:
assumes $x \in_\circ vfield\ A$
obtains r **where** $x = \langle r,\ \mathcal{F}_\circ\ r\rangle$ **and** $r \in_\circ A$
⟨*proof*⟩

lemma *app-vfieldI1*[*intro*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r \cup_\circ \mathcal{R}_\circ\ r$
shows $a \in_\circ \mathcal{F}_\circ\ r$
⟨*proof*⟩

lemma *app-vfieldI2*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$
shows $a \in_\circ \mathcal{F}_\circ\ r$
⟨*proof*⟩

lemma *app-vfieldI3*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$
shows $b \in_\circ \mathcal{F}_\circ\ r$
⟨*proof*⟩

lemma *app-vfieldD*[*dest*]:

assumes $a \in_\circ \mathcal{F}_\circ\ r$
shows $a \in_\circ \mathcal{D}_\circ\ r \cup_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *app-vfieldE*[*elim*]:
assumes $a \in_\circ \mathcal{F}_\circ\ r$ **and** $a \in_\circ \mathcal{D}_\circ\ r \cup_\circ \mathcal{R}_\circ\ r \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *app-vfield-vpairE*[*elim*]:
assumes $a \in_\circ \mathcal{F}_\circ\ r$
obtains b **where** $\langle a,\ b\rangle \in_\circ r \vee \langle b,\ a\rangle \in_\circ r$
⟨*proof*⟩

lemma *vfield-iff*: $a \in_\circ \mathcal{F}_\circ\ r \longleftrightarrow (\exists\, b.\ \langle a,\ b\rangle \in_\circ r \vee \langle b,\ a\rangle \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *vfield-vempty*[*simp*]: $\mathcal{F}_\circ\ 0 = 0$ ⟨*proof*⟩

lemma *vfield-vsingleton*[*simp*]: $\mathcal{F}_\circ\ (set\ \{\langle a,\ b\rangle\}) = set\ \{a,\ b\}$
⟨*proof*⟩

lemma *vfield-vdoubleton*[*simp*]: $\mathcal{F}_\circ\ (set\ \{\langle a,\ b\rangle,\ \langle c,\ d\rangle\}) = set\ \{a,\ b,\ c,\ d\}$
⟨*proof*⟩

lemma *vfield-mono*:
assumes $r \subseteq_\circ s$
shows $\mathcal{F}_\circ\ r \subseteq_\circ \mathcal{F}_\circ\ s$
⟨*proof*⟩

lemma *vfield-vinsert*[*simp*]: $\mathcal{F}_\circ\ (vinsert\ \langle a,\ b\rangle\ r) = set\ \{a,\ b\} \cup_\circ \mathcal{F}_\circ\ r$
⟨*proof*⟩

lemma *vfield-vunion*[*simp*]: $\mathcal{F}_\circ\ (r \cup_\circ s) = \mathcal{F}_\circ\ r \cup_\circ \mathcal{F}_\circ\ s$
⟨*proof*⟩

Connections.

lemma *vid-on-vfield*[*simp*]: $\mathcal{F}_\circ\ (vid\text{-}on\ A) = A$ ⟨*proof*⟩

lemma *vconst-on-vfield-ne*[*intro*, *simp*]:
assumes $A \neq 0$
shows $\mathcal{F}_\circ\ (vconst\text{-}on\ A\ c) = vinsert\ c\ A$
⟨*proof*⟩

lemma *vconst-on-vfield-vempty*[*simp*]: $\mathcal{F}_\circ\ (vconst\text{-}on\ 0\ c) = 0$ ⟨*proof*⟩

lemma *vfield-vconverse*[*simp*]: $\mathcal{F}_\circ\ ({r^{-1}}_\circ) = \mathcal{F}_\circ\ r$
⟨*proof*⟩

Image

definition *vimage* :: $V \Rightarrow V \Rightarrow V$
where $vimage\ D = VLambda\ D\ (\lambda\langle r,\ A\rangle.\ \mathcal{R}_\circ\ (r \restriction^l_\circ A))$

abbreviation *app-vimage* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹‘$_\circ$› 90)
where $r\ `_\circ\ A \equiv vimage\ (set\ \{\langle r,\ A\rangle\})\ (\!|\langle r,\ A\rangle|\!)$

lemma *app-vimage-def*: $r\ `_\circ\ A = \mathcal{R}_\circ\ (r \restriction^l_\circ A)$ ⟨*proof*⟩

lemma *vimage-small*[*simp*]: *small* $\{b.\ \exists\, a{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *app-vimage-set-def*: $r\ `_\circ\ A = set\ \{b.\ \exists\, a{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *vimageI*[*intro*!]:
assumes $\langle r,\ A\rangle \in_\circ D$
shows $\langle\langle r,\ A\rangle,\ r\ `_\circ\ A\rangle \in_\circ vimage\ D$
⟨*proof*⟩

lemma *vimageD*[*dest*]:
assumes $\langle\langle r,\ A\rangle,\ s\rangle \in_\circ vimage\ D$
shows $\langle r,\ A\rangle \in_\circ D$ **and** $s = r\ `_\circ\ A$
⟨*proof*⟩

lemma *vimageE*[*elim*]:
assumes $x \in_\circ vimage\ (R \times_\circ X)$
obtains $r\ A$ **where** $x = \langle\langle r,\ A\rangle,\ r\ `_\circ\ A\rangle$ **and** $r \in_\circ R$ **and** $A \in_\circ X$
⟨*proof*⟩

lemma *app-vimageI*1:
assumes $x \in_\circ \mathcal{R}_\circ\ (r\ |^l_\circ\ A)$
shows $x \in_\circ r\ `_\circ\ A$
⟨*proof*⟩

lemma *app-vimageI*2[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$ **and** $a \in_\circ A$
shows $b \in_\circ r\ `_\circ\ A$
⟨*proof*⟩

lemma *app-vimageD*[*dest*]:
assumes $x \in_\circ r\ `_\circ\ A$
shows $x \in_\circ \mathcal{R}_\circ\ (r\ |^l_\circ\ A)$
⟨*proof*⟩

lemma *app-vimageE*[*elim*]:
assumes $b \in_\circ r\ `_\circ\ A$
obtains a **where** $\langle a,\ b\rangle \in_\circ r$ **and** $a \in_\circ A$
⟨*proof*⟩

lemma *app-vimage-iff*: $b \in_\circ r\ `_\circ\ A \longleftrightarrow (\exists\, a{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *vimage-vempty*[*simp*]: $0\ `_\circ\ A = 0$ ⟨*proof*⟩

lemma *vimage-of-vempty*[*simp*]: $r\ `_\circ\ 0 = 0$ ⟨*proof*⟩

lemma *vimage-vsingleton*: $r\ `_\circ\ set\ \{a\} = set\ \{b.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vimage-vsingleton-in*[*intro*, *simp*]:
assumes $a \in_\circ A$
shows $set\ \{\langle a,\ b\rangle\}\ `_\circ\ A = set\ \{b\}$
⟨*proof*⟩

lemma *vimage-vsingleton-nin*[*intro*, *simp*]:
assumes $a \notin_\circ A$
shows $set\ \{\langle a, b\rangle\}\ `_\circ\ A = 0$
⟨*proof*⟩

lemma *vimage-vsingleton-vinsert*[*simp*]: $set\ \{\langle a, b\rangle\}\ `_\circ\ vinsert\ a\ A = set\ \{b\}$
⟨*proof*⟩

lemma *vimage-mono*:
assumes $r' \subseteq_\circ r$ **and** $A' \subseteq_\circ A$
shows $(r'\ `_\circ\ A') \subseteq_\circ (r\ `_\circ\ A)$
⟨*proof*⟩

lemma *vimage-vinsert*: $r\ `_\circ\ (vinsert\ a\ A) = r\ `_\circ\ set\ \{a\} \cup_\circ r\ `_\circ\ A$
⟨*proof*⟩

lemma *vimage-vunion-left*: $(r \cup_\circ s)\ `_\circ\ A = r\ `_\circ\ A \cup_\circ s\ `_\circ\ A$
⟨*proof*⟩

lemma *vimage-vunion-right*: $r\ `_\circ\ (A \cup_\circ B) = r\ `_\circ\ A \cup_\circ r\ `_\circ\ B$
⟨*proof*⟩

lemma *vimage-vintersection*: $r\ `_\circ\ (A \cap_\circ B) \subseteq_\circ r\ `_\circ\ A \cap_\circ r\ `_\circ\ B$ ⟨*proof*⟩

lemma *vimage-vdiff*: $r\ `_\circ\ A -_\circ r\ `_\circ\ B \subseteq_\circ r\ `_\circ\ (A -_\circ B)$ ⟨*proof*⟩

Previous set operations.

lemma *VPow-vinsert*:
$VPow\ (vinsert\ a\ A) = VPow\ A \cup_\circ ((\lambda x{\in_\circ} VPow\ A.\ vinsert\ a\ x)\ `_\circ\ VPow\ A)$
⟨*proof*⟩

Special properties.

lemma *vimage-vsingleton-iff*[*iff*]: $b \in_\circ r\ `_\circ\ set\ \{a\} \longleftrightarrow \langle a, b\rangle \in_\circ r$ ⟨*proof*⟩

lemma *vimage-is-vempty*[*iff*]: $r\ `_\circ\ A = 0 \longleftrightarrow vdisjnt\ (\mathcal{D}_\circ\ r)\ A$ ⟨*proof*⟩

lemma *vcomp-vimage-vtimes-right*:
assumes $r\ `_\circ\ Y = Z$
shows $r \circ_\circ (X \times_\circ Y) = X \times_\circ Z$
⟨*proof*⟩

Connections.

lemma *vid-on-vimage*[*simp*]: $vid\text{-}on\ A\ `_\circ\ B = A \cap_\circ B$
⟨*proof*⟩

lemma *vimage-vconst-on-ne*[*simp*]:
assumes $B \cap_\circ A \neq 0$
shows $vconst\text{-}on\ A\ c\ `_\circ\ B = set\ \{c\}$
⟨*proof*⟩

lemma *vimage-vconst-on-vempty*[*simp*]:
assumes $vdisjnt\ A\ B$
shows $vconst\text{-}on\ A\ c\ `_\circ\ B = 0$
⟨*proof*⟩

lemma *vimage-vconst-on-vsubset-vconst*: $vconst\text{-}on\ A\ c\ `_\circ\ B \subseteq_\circ set\ \{c\}$ ⟨*proof*⟩

lemma *vimage-VLambda-vrange*: $(\lambda a \in_\circ A.\ f\ a)\ `_\circ\ B = \mathcal{R}_\circ\ (\lambda a \in_\circ A \cap_\circ B.\ f\ a)$
⟨*proof*⟩

lemma *vimage-VLambda-vrange-rep*: $(\lambda a \in_\circ A.\ f\ a)\ `_\circ\ A = \mathcal{R}_\circ\ (\lambda a \in_\circ A.\ f\ a)$
⟨*proof*⟩

lemma *vcomp-vimage*: $(r \circ_\circ s)\ `_\circ\ A = r\ `_\circ\ (s\ `_\circ\ A)$
⟨*proof*⟩

lemma *vimage-vlrestriction*[*simp*]: $(r \restriction^l_\circ A)\ `_\circ\ B = r\ `_\circ\ (A \cap_\circ B)$
⟨*proof*⟩

lemma *vimage-vrrestriction*[*simp*]: $(r \restriction^r_\circ A)\ `_\circ\ B = A \cap_\circ r\ `_\circ\ B$ ⟨*proof*⟩

lemma *vimage-vrestriction*[*simp*]: $(r \restriction_\circ A)\ `_\circ\ B = A \cap_\circ (r\ `_\circ\ (A \cap_\circ B))$ ⟨*proof*⟩

lemma *vimage-vdomain*: $r\ `_\circ\ \mathcal{D}_\circ\ r = \mathcal{R}_\circ\ r$ ⟨*proof*⟩

lemma *vimage-eq-imp-vcomp*:
assumes $r\ `_\circ\ A = s\ `_\circ\ B$
shows $(t \circ_\circ r)\ `_\circ\ A = (t \circ_\circ s)\ `_\circ\ B$
⟨*proof*⟩

Previous connections.

lemma *vcomp-rel-vconst*: $r \circ_\circ (\mathit{vconst\text{-}on}\ A\ c) = A \times_\circ (r\ `_\circ\ \mathit{set}\ \{c\})$
⟨*proof*⟩

lemma *vcomp-VLambda*:
$(\lambda b \in_\circ ((\lambda a \in_\circ A.\ g\ a)\ `_\circ\ A).\ f\ b) \circ_\circ (\lambda a \in_\circ A.\ g\ a) = (\lambda a \in_\circ A.\ (f \circ g)\ a)$
⟨*proof*⟩

Further special properties.

lemma *vimage-vsubset*:
assumes $r \subseteq_\circ A \times_\circ B$
shows $r\ `_\circ\ C \subseteq_\circ B$
⟨*proof*⟩

lemma *vimage-vdomain-vsubset*: $r\ `_\circ\ A \subseteq_\circ r\ `_\circ\ \mathcal{D}_\circ\ r$ ⟨*proof*⟩

lemma *vdomain-vsubset-VUnion2*: $\mathcal{D}_\circ\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ r)$
⟨*proof*⟩

lemma *vrange-vsubset-VUnion2*: $\mathcal{R}_\circ\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ r)$
⟨*proof*⟩

lemma *vfield-vsubset-VUnion2*: $\mathcal{F}_\circ\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ r)$
⟨*proof*⟩

Inverse image

definition *invimage* :: $V \Rightarrow V$
where $\mathit{invimage}\ D = \mathit{VLambda}\ D\ (\lambda\langle r,\ A\rangle.\ r^{-1}_\circ\ `_\circ\ A)$

abbreviation *app-invimage* :: $V \Rightarrow V \Rightarrow V$ (**infixr** $\langle -`_\circ\rangle$ 90)
where $r -`_\circ A \equiv \mathit{invimage}\ (\mathit{set}\ \{\langle r,\ A\rangle\})\ (\!|\langle r,\ A\rangle|\!)$

lemma *app-invimage-def*: $r -`_\circ A = r^{-1}_\circ\ `_\circ\ A$ ⟨*proof*⟩

lemma *invimage-small*[*simp*]: *small* $\{a.\ \exists\, b{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *invimageI*[*intro*!]:
assumes $\langle r,\ A\rangle \in_\circ D$
shows $\langle\langle r,\ A\rangle,\ r -`_\circ A\rangle \in_\circ$ *invimage* D
⟨*proof*⟩

lemma *invimageD*[*dest*]:
assumes $\langle\langle r,\ A\rangle,\ s\rangle \in_\circ$ *invimage* D
shows $\langle r,\ A\rangle \in_\circ D$ **and** $s = r -`_\circ A$
⟨*proof*⟩

lemma *invimageE*[*elim*]:
assumes $x \in_\circ$ *invimage* D **and** $D \subseteq_\circ R \times_\circ X$
obtains $r\ A$ **where** $x = \langle\langle r,\ A\rangle,\ r -`_\circ A\rangle$ **and** $r \in_\circ R$ **and** $A \in_\circ X$
⟨*proof*⟩

lemma *app-invimageI*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ r$ **and** $b \in_\circ A$
shows $a \in_\circ r -`_\circ A$
⟨*proof*⟩

lemma *app-invimageD*[*dest*]:
assumes $a \in_\circ r -`_\circ A$
shows $a \in_\circ \mathcal{D}_\circ\ (r \restriction^r_\circ A)$
⟨*proof*⟩

lemma *app-invimageE*[*elim*]:
assumes $a \in_\circ r -`_\circ A$
obtains b **where** $\langle a,\ b\rangle \in_\circ r$ **and** $b \in_\circ A$
⟨*proof*⟩

lemma *app-invimageI*1:
assumes $a \in_\circ \mathcal{D}_\circ\ (r \restriction^r_\circ A)$
shows $a \in_\circ r -`_\circ A$
⟨*proof*⟩

lemma *app-invimageD*1:
assumes $a \in_\circ r -`_\circ A$
shows $a \in_\circ \mathcal{D}_\circ\ (r \restriction^r_\circ A)$
⟨*proof*⟩

lemma *app-invimageE*1:
assumes $a \in_\circ r -`_\circ A$ **and** $a \in_\circ \mathcal{D}_\circ\ (r \restriction^r_\circ A) \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *app-invimageI*2:
assumes $a \in_\circ r^{-1}{}_\circ\ `_\circ A$
shows $a \in_\circ r -`_\circ A$
⟨*proof*⟩

lemma *app-invimageD*2:
assumes $a \in_\circ r -`_\circ A$
shows $a \in_\circ r^{-1}{}_\circ\ `_\circ A$
⟨*proof*⟩

lemma *app-invimageE2*:
assumes $a \in_\circ r -`_\circ A$ **and** $a \in_\circ r^{-1}{}_\circ `_\circ A \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *invimage-iff*: $a \in_\circ r -`_\circ A \longleftrightarrow (\exists\, b{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r)$ ⟨*proof*⟩

lemma *invimage-iff1*: $a \in_\circ r -`_\circ A \longleftrightarrow a \in_\circ \mathcal{D}_\circ\ (r \restriction^r_\circ A)$ ⟨*proof*⟩

lemma *invimage-iff2*: $a \in_\circ r -`_\circ A \longleftrightarrow a \in_\circ r^{-1}{}_\circ `_\circ A$ ⟨*proof*⟩

Set operations.

lemma *invimage-vempty*[*simp*]: $0 -`_\circ A = 0$ ⟨*proof*⟩

lemma *invimage-of-vempty*[*simp*]: $r -`_\circ 0 = 0$ ⟨*proof*⟩

lemma *invimage-vsingleton-in*[*intro*, *simp*]:
assumes $b \in_\circ A$
shows $set\ \{\langle a,\ b\rangle\} -`_\circ A = set\ \{a\}$
⟨*proof*⟩

lemma *invimage-vsingleton-nin*[*intro*, *simp*]:
assumes $b \notin_\circ A$
shows $set\ \{\langle a,\ b\rangle\} -`_\circ A = 0$
⟨*proof*⟩

lemma *invimage-vsingleton-vinsert*[*intro*, *simp*]:
$set\ \{\langle a,\ b\rangle\} -`_\circ vinsert\ b\ A = set\ \{a\}$
⟨*proof*⟩

lemma *invimage-mono*:
assumes $r' \subseteq_\circ r$ **and** $A' \subseteq_\circ A$
shows $(r' -`_\circ A') \subseteq_\circ (r -`_\circ A)$
⟨*proof*⟩

lemma *invimage-vinsert*: $r -`_\circ (vinsert\ a\ A) = r -`_\circ set\ \{a\} \cup_\circ r -`_\circ A$
⟨*proof*⟩

lemma *invimage-vunion-left*: $(r \cup_\circ s) -`_\circ A = r -`_\circ A \cup_\circ s -`_\circ A$
⟨*proof*⟩

lemma *invimage-vunion-right*: $r -`_\circ (A \cup_\circ B) = r -`_\circ A \cup_\circ r -`_\circ B$
⟨*proof*⟩

lemma *invimage-vintersection*: $r -`_\circ (A \cap_\circ B) \subseteq_\circ r -`_\circ A \cap_\circ r -`_\circ B$ ⟨*proof*⟩

lemma *invimage-vdiff*: $r -`_\circ A -_\circ r -`_\circ B \subseteq_\circ r -`_\circ (A -_\circ B)$ ⟨*proof*⟩

Special properties.

lemma *invimage-set-def*: $r -`_\circ A = set\ \{a.\ \exists\, b{\in_\circ}A.\ \langle a,\ b\rangle \in_\circ r\}$ ⟨*proof*⟩

lemma *invimage-eq-vdomain-vrestriction*: $r -`_\circ A = \mathcal{D}_\circ\ (r \restriction^r_\circ A)$ ⟨*proof*⟩

lemma *invimage-vrange*[*simp*]: $r -`_\circ \mathcal{R}_\circ\ r = \mathcal{D}_\circ\ r$
⟨*proof*⟩

lemma *invimage-vrange-vsubset*[*simp*]:

assumes $\mathcal{R}_\circ\ r \subseteq_\circ B$
shows $r -`_\circ B = \mathcal{D}_\circ\ r$
⟨*proof*⟩

Connections.

lemma *invimage-vid-on*[*simp*]: *vid-on* $A -`_\circ B = A \cap_\circ B$
⟨*proof*⟩

lemma *invimage-vconst-on-vsubset-vdomain*[*simp*]: *vconst-on* $A\ c -`_\circ B \subseteq_\circ A$
⟨*proof*⟩

lemma *invimage-vconst-on-ne*[*simp*]:
assumes $c \in_\circ B$
shows *vconst-on* $A\ c -`_\circ B = A$
⟨*proof*⟩

lemma *invimage-vconst-on-vempty*[*simp*]:
assumes $c \notin_\circ B$
shows *vconst-on* $A\ c -`_\circ B = 0$
⟨*proof*⟩

lemma *invimage-vcomp*: $(r \circ_\circ s) -`_\circ x = s -`_\circ (r -`_\circ x)$
⟨*proof*⟩

lemma *invimage-vconverse*[*simp*]: $r^{-1}{}_\circ -`_\circ A = r\ `_\circ A$
⟨*proof*⟩

lemma *invimage-vlrestriction*[*simp*]: $(r \restriction^l_\circ A) -`_\circ B = A \cap_\circ r -`_\circ B$ ⟨*proof*⟩

lemma *invimage-vrrestriction*[*simp*]: $(r \restriction^r_\circ A) -`_\circ B = (r -`_\circ (A \cap_\circ B))$
⟨*proof*⟩

lemma *invimage-vrestriction*[*simp*]: $(r \restriction_\circ A) -`_\circ B = A \cap_\circ (r -`_\circ (A \cap_\circ B))$
⟨*proof*⟩

Previous connections.

lemma *vcomp-vconst-on-rel-vtimes*: *vconst-on* $A\ c \circ_\circ r = (r -`_\circ A) \times_\circ set\ \{c\}$
⟨*proof*⟩

lemma *vdomain-vcomp*[*simp*]: $\mathcal{D}_\circ\ (r \circ_\circ s) = s -`_\circ \mathcal{D}_\circ\ r$ ⟨*proof*⟩

lemma *vrange-vcomp*[*simp*]: $\mathcal{R}_\circ\ (r \circ_\circ s) = r\ `_\circ \mathcal{R}_\circ\ s$ ⟨*proof*⟩

lemma *vdomain-vcomp-vsubset*:
assumes $\mathcal{R}_\circ\ s \subseteq_\circ \mathcal{D}_\circ\ r$
shows $\mathcal{D}_\circ\ (r \circ_\circ s) = \mathcal{D}_\circ\ s$
⟨*proof*⟩

2.4.4 Classification of relations

Binary relation

locale *vbrelation* =
fixes $r :: V$
assumes *vbrelation*: *vpairs* $r = r$

Rules.

lemma *vpairs-eqI*[*intro*!]:

assumes $\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = \langle a,\ b\rangle$
shows *vpairs r* = *r*
⟨*proof*⟩

lemma *vpairs-eqD*[*dest*]:
assumes *vpairs r* = *r*
shows $\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = \langle a,\ b\rangle$
⟨*proof*⟩

lemma *vpairs-eqE*[*elim*!]:
assumes *vpairs r* = *r* **and** $(\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = \langle a,\ b\rangle) \Longrightarrow P$
shows *P*
⟨*proof*⟩

lemmas *vbrelationI*[*intro*!] = *vbrelation.intro*
lemmas *vbrelationD*[*dest*!] = *vbrelation.vbrelation*

lemma *vbrelationE*[*elim*!]:
assumes *vbrelation r* **and** (*vpairs r* = *r*) $\Longrightarrow P$
shows *P*
⟨*proof*⟩

lemma *vbrelationE*1[*elim*]:
assumes *vbrelation r* **and** $x \in_\circ r$
obtains *a b* **where** $x = \langle a,\ b\rangle$
⟨*proof*⟩

lemma *vbrelationD*1[*dest*]:
assumes *vbrelation r* **and** $x \in_\circ r$
shows $\exists\, a\ b.\ x = \langle a,\ b\rangle$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vinE*:
assumes $x \in_\circ r$
obtains *a b* **where** $x = \langle a,\ b\rangle$ **and** $a \in_\circ \mathcal{D}_\circ\ r$ **and** $b \in_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

Set operations.

lemma *vbrelation-vsubset*:
assumes *vbrelation s* **and** $r \subseteq_\circ s$
shows *vbrelation r*
⟨*proof*⟩

lemma *vbrelation-vinsert*[*simp*]: *vbrelation* (*vinsert* $\langle a,\ b\rangle$ *r*) $\longleftrightarrow$ *vbrelation r*
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vinsertI*[*intro*, *simp*]:
vbrelation (*vinsert* $\langle a,\ b\rangle$ *r*)
⟨*proof*⟩

lemma *vbrelation-vinsertD*[*dest*]:
assumes *vbrelation* (*vinsert* $\langle a,\ b\rangle$ *r*)
shows *vbrelation r*
⟨*proof*⟩

lemma *vbrelation-vunion*: *vbrelation* $(r \cup_\circ s) \longleftrightarrow$ *vbrelation r* $\wedge$ *vbrelation s*
⟨*proof*⟩

lemma *vbrelation-vunionI*:
assumes *vbrelation r* **and** *vbrelation s*
shows *vbrelation* $(r \cup_\circ s)$
⟨*proof*⟩

lemma *vbrelation-vunionD*[*dest*]:
assumes *vbrelation* $(r \cup_\circ s)$
shows *vbrelation r* **and** *vbrelation s*
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vintersectionI*: *vbrelation* $(r \cap_\circ s)$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vdiffI*: *vbrelation* $(r -_\circ s)$
⟨*proof*⟩

Connections.

lemma *vbrelation-vempty*: *vbrelation* 0 ⟨*proof*⟩

lemma *vbrelation-vsingleton*: *vbrelation* (*set* $\{\langle a, b\rangle\}$) ⟨*proof*⟩

lemma *vbrelation-vdoubleton*: *vbrelation* (*set* $\{\langle a, b\rangle, \langle c, d\rangle\}$) ⟨*proof*⟩

lemma *vbrelation-vid-on*[*simp*]: *vbrelation* (*vid-on A*) ⟨*proof*⟩

lemma *vbrelation-vconst-on*[*simp*]: *vbrelation* (*vconst-on A c*) ⟨*proof*⟩

lemma *vbrelation-VLambda*[*simp*]: *vbrelation* (*VLambda A f*)
⟨*proof*⟩

global-interpretation *rel-VLambda*: *vbrelation* ‹*VLambda U f*›
⟨*proof*⟩

lemma *vbrelation-vcomp*:
assumes *vbrelation r* **and** *vbrelation s*
shows *vbrelation* $(r \circ_\circ s)$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vconverse*: *vbrelation* $(r^{-1}{}_\circ)$
⟨*proof*⟩

lemma *vbrelation-vlrestriction*[*intro*, *simp*]: *vbrelation* $(r \restriction^l_\circ A)$ ⟨*proof*⟩

lemma *vbrelation-vrrestriction*[*intro*, *simp*]: *vbrelation* $(r \restriction^r_\circ A)$ ⟨*proof*⟩

lemma *vbrelation-vrestriction*[*intro*, *simp*]: *vbrelation* $(r \restriction_\circ A)$ ⟨*proof*⟩

Previous connections.

lemma (**in** *vbrelation*) *vconverse-vconverse*[*simp*]: $(r^{-1}{}_\circ)^{-1}{}_\circ = r$
⟨*proof*⟩

lemma *vconverse-mono*[*simp*]:
assumes *vbrelation r* **and** *vbrelation s*
shows $r^{-1}{}_\circ \subseteq_\circ s^{-1}{}_\circ \longleftrightarrow r \subseteq_\circ s$
⟨*proof*⟩

lemma *vconverse-inject*[*simp*]:
assumes *vbrelation r* **and** *vbrelation s*

shows $r^{-1}{}_\circ = s^{-1}{}_\circ \longleftrightarrow r = s$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vconverse-vsubset-swap-2*:
assumes $r^{-1}{}_\circ \subseteq_\circ s$
shows $r \subseteq_\circ s^{-1}{}_\circ$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vlrestriction-vdomain*[*simp*]: $r \upharpoonright^l_\circ \mathcal{D}_\circ\ r = r$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vrrestriction-vrange*[*simp*]: $r \upharpoonright^r_\circ \mathcal{R}_\circ\ r = r$
⟨*proof*⟩

Special properties.

lemma *brel-vsubset-vtimes*:
vbrelation $r \longleftrightarrow r \subseteq_\circ$ *set* (*vfst* ‘ *elts* r) $\times_\circ$ *set* (*vsnd* ‘ *elts* r)
⟨*proof*⟩

lemma *vsubset-vtimes-vbrelation*:
assumes $r \subseteq_\circ A \times_\circ B$
shows *vbrelation* r
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vintersection-vdomain*:
assumes *vdisjnt* ($\mathcal{D}_\circ$ r) ($\mathcal{D}_\circ$ s)
shows *vdisjnt* r s
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vintersection-vrange*:
assumes *vdisjnt* ($\mathcal{R}_\circ$ r) ($\mathcal{R}_\circ$ s)
shows *vdisjnt* r s
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vintersection-vfield*:
assumes *vdisjnt* (*vfield* r) (*vfield* s)
shows *vdisjnt* r s
⟨*proof*⟩

lemma (**in** *vbrelation*) *vdomain-vrange-vtimes*: $r \subseteq_\circ \mathcal{D}_\circ\ r \times_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vsubset-vtimes*:
assumes $\mathcal{D}_\circ\ r \subseteq_\circ A$ **and** $\mathcal{R}_\circ\ r \subseteq_\circ B$
shows $r \subseteq_\circ A \times_\circ B$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vlrestriction-vsubset-vrange*[*intro*, *simp*]:
assumes $\mathcal{D}_\circ\ r \subseteq_\circ A$
shows $r \upharpoonright^l_\circ A = r$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vrrestriction-vsubset-vrange*[*intro*, *simp*]:
assumes $\mathcal{R}_\circ\ r \subseteq_\circ B$
shows $r \upharpoonright^r_\circ B = r$
⟨*proof*⟩

lemma (**in** *vbrelation*) *vbrelation-vcomp-vid-on-left*[*simp*]:

assumes $\mathcal{R}_\circ\ r \subseteq_\circ A$
shows *vid-on* $A \circ_\circ r = r$
⟨*proof*⟩

lemma (in *vbrelation*) *vbrelation-vcomp-vid-on-right*[*simp*]:
assumes $\mathcal{D}_\circ\ r \subseteq_\circ A$
shows $r \circ_\circ$ *vid-on* $A = r$
⟨*proof*⟩

Alternative forms of existing results.

lemmas [*intro*, *simp*] = *vbrelation.vconverse-vconverse*
and [*intro*, *simp*] = *vbrelation.vlrestriction-vsubset-vrange*
and [*intro*, *simp*] = *vbrelation.vrrestriction-vsubset-vrange*

Simple single-valued relation

locale *vsv* = *vbrelation r* **for** *r* +
assumes *vsv*: $[\![\ \langle a, b\rangle \in_\circ r;\ \langle a, c\rangle \in_\circ r\ ]\!] \Longrightarrow b = c$

Rules.

lemmas (in *vsv*) [*intro*] = *vsv-axioms*

mk-ide rf *vsv-def*[*unfolded vsv-axioms-def*]
|*intro vsvI*[*intro*]|
|*dest vsvD*[*dest*]|
|*elim vsvE*[*elim*]|

Set operations.

lemma (in *vsv*) *vsv-vinsert*[*simp*]:
assumes $a \notin_\circ \mathcal{D}_\circ\ r$
shows *vsv* (*vinsert* $\langle a, b\rangle\ r$)
⟨*proof*⟩

lemma *vsv-vinsertD*:
assumes *vsv* (*vinsert x r*)
shows *vsv r*
⟨*proof*⟩

lemma *vsv-vunion*[*intro*, *simp*]:
assumes *vsv r* **and** *vsv s* **and** *vdisjnt* $(\mathcal{D}_\circ\ r)\ (\mathcal{D}_\circ\ s)$
shows *vsv* $(r \cup_\circ s)$
⟨*proof*⟩

lemma (in *vsv*) *vsv-vintersection*[*intro*, *simp*]: *vsv* $(r \cap_\circ s)$
⟨*proof*⟩

lemma (in *vsv*) *vsv-vdiff*[*intro*, *simp*]: *vsv* $(r -_\circ s)$ ⟨*proof*⟩

Connections.

lemma *vsv-vempty*[*simp*]: *vsv* 0 ⟨*proof*⟩

lemma *vsv-vsingleton*[*simp*]: *vsv* (*set* $\{\langle a, b\rangle\}$) ⟨*proof*⟩

global-interpretation *rel-vsingleton*: *vsv* ‹*set* $\{\langle a, b\rangle\}$›
⟨*proof*⟩

lemma *vsv-vdoubleton*:

assumes $a \neq c$
shows *vsv* (*set* $\{\langle a, b\rangle, \langle c, d\rangle\}$)
⟨*proof*⟩

lemma *vsv-vid-on*[*simp*]: *vsv* (*vid-on A*) ⟨*proof*⟩

lemma *vsv-vconst-on*[*simp*]: *vsv* (*vconst-on A c*) ⟨*proof*⟩

lemma *vsv-VLambda*[*simp*]: *vsv* ($\lambda a{\in_\circ} A.\ f\ a$) ⟨*proof*⟩

global-interpretation *rel-VLambda*: *vsv* ‹($\lambda a{\in_\circ} A.\ f\ a$)›
⟨*proof*⟩

lemma *vsv-vcomp*:
assumes *vsv r* **and** *vsv s*
shows *vsv* ($r \circ_\circ s$)
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-vlrestriction*[*intro*, *simp*]: *vsv* ($r \restriction^l_\circ A$)
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-vrrestriction*[*intro*, *simp*]: *vsv* ($r \restriction^r_\circ A$)
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-vrestriction*[*intro*, *simp*]: *vsv* ($r \restriction_\circ A$)
⟨*proof*⟩

Special properties.

lemma *small-vsv*[*simp*]: *small* $\{f.\ \text{vsv}\ f \wedge \mathcal{D}_\circ\ f = A \wedge \mathcal{R}_\circ\ f \subseteq_\circ B\}$
⟨*proof*⟩

context *vsv*
begin

lemma *vsv-ex*1:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $\exists!b.\ \langle a, b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vsv-ex*1-*app*1:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $b = r(\!|a|\!) \longleftrightarrow \langle a, b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vsv-ex*1-*app*2[*iff*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $r(\!|a|\!) = b \longleftrightarrow \langle a, b\rangle \in_\circ r$
⟨*proof*⟩

lemma *vsv-appI*[*intro*, *simp*]:
assumes $\langle a, b\rangle \in_\circ r$
shows $r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vsv-appE*:
assumes $r(\!|a|\!) = b$ **and** $a \in_\circ \mathcal{D}_\circ\ r$ **and** $\langle a, b\rangle \in_\circ r \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vdomain-vrange-is-vempty*: $\mathcal{D}_\circ\ r = 0 \longleftrightarrow \mathcal{R}_\circ\ r = 0$ ⟨*proof*⟩

lemma *vsv-vrange-vempty*:
assumes $\mathcal{R}_\circ\ r = 0$
shows $r = 0$
⟨*proof*⟩

lemma *vsv-vdomain-vempty-vrange-vempty*:
assumes $\mathcal{D}_\circ\ r \neq 0$
shows $\mathcal{R}_\circ\ r \neq 0$
⟨*proof*⟩

lemma *vsv-vdomain-vsingleton-vrange-vsingleton*:
assumes $\mathcal{D}_\circ\ r = set\ \{a\}$
obtains b **where** $\mathcal{R}_\circ\ r = set\ \{b\}$
⟨*proof*⟩

lemma *vsv-vsubset-vimageE*:
assumes $B \subseteq_\circ r\ `_\circ\ A$
obtains C **where** $C \subseteq_\circ A$ **and** $B = r\ `_\circ\ C$
⟨*proof*⟩

lemma *vsv-vimage-eqI*[*intro*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) = b$ **and** $a \in_\circ A$
shows $b \in_\circ r\ `_\circ\ A$
⟨*proof*⟩

lemma *vsv-vimageI1*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a \in_\circ A$
shows $r(\!|a|\!) \in_\circ r\ `_\circ\ A$
⟨*proof*⟩

lemma *vsv-vimageI2*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $r(\!|a|\!) \in_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vsv-vimageI2′*:
assumes $b = r(\!|a|\!)$ **and** $a \in_\circ \mathcal{D}_\circ\ r$
shows $b \in_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vsv-value*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
obtains b **where** $r(\!|a|\!) = b$ **and** $b \in_\circ \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemma *vsv-vimageE*:
assumes $b \in_\circ r\ `_\circ\ A$
obtains x **where** $r(\!|x|\!) = b$ **and** $x \in_\circ A$
⟨*proof*⟩

lemma *vsv-vimage-iff*: $b \in_\circ r\ `_\circ\ A \longleftrightarrow (\exists\, a.\ a \in_\circ A \wedge a \in_\circ \mathcal{D}_\circ\ r \wedge r(\!|a|\!) = b)$
⟨*proof*⟩

lemma *vsv-vimage-vsingleton*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$

shows r ‘$_\circ$ $set\ \{a\} = set\ \{r(\!|a|\!)\}$
⟨*proof*⟩

lemma *vsv-vimage-vsubsetI*:
assumes $\bigwedge a.\ [\![\ a \in_\circ A;\ a \in_\circ \mathcal{D}_\circ\ r\ ]\!] \Longrightarrow r(\!|a|\!) \in_\circ B$
shows r ‘$_\circ$ $A \subseteq_\circ B$
⟨*proof*⟩

lemma *vsv-image-vsubset-iff*:
r ‘$_\circ$ $A \subseteq_\circ B \longleftrightarrow (\forall\, a{\in_\circ}A.\ a \in_\circ \mathcal{D}_\circ\ r \longrightarrow r(\!|a|\!) \in_\circ B)$
⟨*proof*⟩

lemma *vsv-vimage-vinsert*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows r ‘$_\circ$ $vinsert\ a\ A = vinsert\ (r(\!|a|\!))\ (r$ ‘$_\circ$ $A)$
⟨*proof*⟩

lemma *vsv-vinsert-vimage*[*intro*, *simp*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a \in_\circ A$
shows $vinsert\ (r(\!|a|\!))\ (r$ ‘$_\circ$ $A) = r$ ‘$_\circ$ A
⟨*proof*⟩

lemma *vsv-is-VLambda*[*simp*]: $(\lambda x{\in_\circ}\mathcal{D}_\circ\ r.\ r(\!|x|\!)) = r$
⟨*proof*⟩

lemma *vsv-is-VLambda-on-vlrestriction*[*intro*, *simp*]:
assumes $A \subseteq_\circ \mathcal{D}_\circ\ r$
shows $(\lambda x{\in_\circ}A.\ r(\!|x|\!)) = r \restriction^l_\circ A$
⟨*proof*⟩

lemma *pairwise-vimageI*:
assumes $\bigwedge x\ y.$
$[\![\ x \in_\circ \mathcal{D}_\circ\ r;\ y \in_\circ \mathcal{D}_\circ\ r;\ x \neq y;\ r(\!|x|\!) \neq r(\!|y|\!)\ ]\!] \Longrightarrow P\ (r(\!|x|\!))\ (r(\!|y|\!))$
shows $vpairwise\ P\ (\mathcal{R}_\circ\ r)$
⟨*proof*⟩

lemma *vsv-vrange-vsubset*:
assumes $\bigwedge x.\ x \in_\circ \mathcal{D}_\circ\ r \Longrightarrow r(\!|x|\!) \in_\circ A$
shows $\mathcal{R}_\circ\ r \subseteq_\circ A$
⟨*proof*⟩

lemma *vsv-vlrestriction-vinsert*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $r \restriction^l_\circ vinsert\ a\ A = vinsert\ \langle a, r(\!|a|\!)\rangle\ (r \restriction^l_\circ A)$
⟨*proof*⟩

end

lemma *vsv-eqI*:
assumes $vsv\ r$
and $vsv\ s$
and $\mathcal{D}_\circ\ r = \mathcal{D}_\circ\ s$
and $\bigwedge a.\ a \in_\circ \mathcal{D}_\circ\ r \Longrightarrow r(\!|a|\!) = s(\!|a|\!)$
shows $r = s$
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-VLambda-cong*:
assumes $\bigwedge a.\ a \in_\circ \mathcal{D}_\circ\ r \Longrightarrow r(\!|a|\!) = f\ a$

shows $(\lambda a \in_\circ \mathcal{D}_\circ\ r.\ f\ a) = r$
⟨*proof*⟩

lemma *Axiom-of-Choice*:
obtains f **where** $\bigwedge x.\ x \in_\circ A \Longrightarrow x \neq 0 \Longrightarrow f(\!|x|\!) \in_\circ x$ **and** *vsv f*
⟨*proof*⟩

lemma *VLambda-eqI*:
assumes $X = Y$ **and** $\bigwedge x.\ x \in_\circ X \Longrightarrow f\ x = g\ x$
shows $(\lambda x \in_\circ X.\ f\ x) = (\lambda y \in_\circ Y.\ g\ y)$
⟨*proof*⟩

lemma *VLambda-vsingleton-def*: $(\lambda i \in_\circ set\ \{j\}.\ f\ i) = (\lambda i \in_\circ set\ \{j\}.\ f\ j)$ ⟨*proof*⟩

Alternative forms of the available results.

lemmas [*iff*] = *vsv.vsv-ex1-app2*
and [*intro*, *simp*] = *vsv.vsv-appI*
and [*elim*] = *vsv.vsv-appE*
and [*intro*] = *vsv.vsv-vimage-eqI*
and [*simp*] = *vsv.vsv-vinsert-vimage*
and [*intro*] = *vsv.vsv-is-VLambda-on-vlrestriction*
and [*simp*] = *vsv.vsv-is-VLambda*
and [*intro*, *simp*] = *vsv.vsv-vintersection*
and [*intro*, *simp*] = *vsv.vsv-vdiff*
and [*intro*, *simp*] = *vsv.vsv-vlrestriction*
and [*intro*, *simp*] = *vsv.vsv-vrrestriction*
and [*intro*, *simp*] = *vsv.vsv-vrestriction*

Specialization of existing properties to single-valued relations.

Identity relation.

lemma *vid-on-eq-atI*[*intro*, *simp*]:
assumes $a = b$ **and** $a \in_\circ A$
shows *vid-on A* $(\!|a|\!) = b$
⟨*proof*⟩

lemma *vid-on-atI*[*intro*, *simp*]:
assumes $a \in_\circ A$
shows *vid-on A* $(\!|a|\!) = a$
⟨*proof*⟩

lemma *vid-on-at-iff*[*intro*, *simp*]:
assumes $a \in_\circ A$
shows *vid-on A* $(\!|a|\!) = b \longleftrightarrow a = b$
⟨*proof*⟩

Constant function.

lemma *vconst-on-atI*[*simp*]:
assumes $a \in_\circ A$
shows *vconst-on A c* $(\!|a|\!) = c$
⟨*proof*⟩

Composition.

lemma *vcomp-atI*[*intro*, *simp*]:
assumes *vsv r*
and *vsv s*
and $a \in_\circ \mathcal{D}_\circ\ r$

and $b \in_\circ \mathcal{D}_\circ\ s$
and $s(\!|b|\!) = c$
and $r(\!|a|\!) = b$
shows $(s \circ_\circ r)(\!|a|\!) = c$
⟨*proof*⟩

lemma *vcomp-atD*[*dest*]:
assumes $(s \circ_\circ r)(\!|a|\!) = c$
and *vsv r*
and *vsv s*
and $a \in_\circ \mathcal{D}_\circ\ r$
and $r(\!|a|\!) \in_\circ \mathcal{D}_\circ\ s$
shows $\exists\, b.\ s(\!|b|\!) = c \wedge r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vcomp-atE*1:
assumes $(s \circ_\circ r)(\!|a|\!) = c$
and *vsv r*
and *vsv s*
and $a \in_\circ \mathcal{D}_\circ\ r$
and $r(\!|a|\!) \in_\circ \mathcal{D}_\circ\ s$
and $\exists\, b.\ s(\!|b|\!) = c \wedge r(\!|a|\!) = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vcomp-atE*[*elim*]:
assumes $(s \circ_\circ r)(\!|a|\!) = c$
and *vsv r*
and *vsv s*
and $a \in_\circ \mathcal{D}_\circ\ r$
and $r(\!|a|\!) \in_\circ \mathcal{D}_\circ\ s$
obtains b **where** $r(\!|a|\!) = b$ **and** $s(\!|b|\!) = c$
⟨*proof*⟩

lemma *vsv-vcomp-at*[*simp*]:
assumes *vsv r* **and** *vsv s* **and** $a \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) \in_\circ \mathcal{D}_\circ\ s$
shows $(s \circ_\circ r)(\!|a|\!) = s(\!|r(\!|a|\!)|\!)$
⟨*proof*⟩

context *vsv*
begin

Converse relation.

lemma *vconverse-atI*[*intro*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) = b$
shows $\langle b,\ a\rangle \in_\circ r^{-1}{}_\circ$
⟨*proof*⟩

lemma *vconverse-atD*[*dest*]:
assumes $\langle b,\ a\rangle \in_\circ r^{-1}{}_\circ$
shows $r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vconverse-atE*[*elim*]:
assumes $\langle b,\ a\rangle \in_\circ r^{-1}{}_\circ$ **and** $r(\!|a|\!) = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vconverse-iff*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $\langle b,\ a\rangle \in_\circ r^{-1}{}_\circ \longleftrightarrow r(\!|a|\!) = b$
⟨*proof*⟩

Left restriction.

interpretation *vlrestriction*: *vsv* ‹$r \restriction^l_\circ A$› ⟨*proof*⟩

lemma *vlrestriction-atI*[*intro*, *simp*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a \in_\circ A$ **and** $r(\!|a|\!) = b$
shows $(r \restriction^l_\circ A)(\!|a|\!) = b$
⟨*proof*⟩

lemma *vlrestriction-atD*[*dest*]:
assumes $(r \restriction^l_\circ A)(\!|a|\!) = b$ **and** $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a \in_\circ A$
shows $r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vlrestriction-atE*1[*elim*]:
assumes $(r \restriction^l_\circ A)(\!|a|\!) = b$
and $a \in_\circ \mathcal{D}_\circ\ r$
and $a \in_\circ A$
and $r(\!|a|\!) = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vlrestriction-atE*2[*elim*]:
assumes $x \in_\circ r \restriction^l_\circ A$
obtains $a\ b$ **where** $x = \langle a,\ b\rangle$ **and** $a \in_\circ A$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

Right restriction.

interpretation *vrrestriction*: *vsv* ‹$r \restriction^r_\circ A$› ⟨*proof*⟩

lemma *vrrestriction-atI*[*intro*, *simp*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $b \in_\circ A$ **and** $r(\!|a|\!) = b$
shows $(r \restriction^r_\circ A)(\!|a|\!) = b$
⟨*proof*⟩

lemma *vrrestriction-atD*[*dest*]:
assumes $(r \restriction^r_\circ A)(\!|a|\!) = b$ **and** $a \in_\circ r -`_\circ A$
shows $b \in_\circ A$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vrrestriction-atE*1[*elim*]:
assumes $(r \restriction^r_\circ A)(\!|a|\!) = b$ **and** $a \in_\circ r -`_\circ A$ **and** $r(\!|a|\!) = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vrrestriction-atE*2[*elim*]:
assumes $x \in_\circ r \restriction^r_\circ A$
obtains $a\ b$ **where** $x = \langle a,\ b\rangle$ **and** $b \in_\circ A$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

Restriction.

interpretation *vrestriction*: *vsv* ‹$r \restriction_\circ A$› ⟨*proof*⟩

lemma *vlrestriction-app*[*intro*, *simp*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a \in_\circ A$
shows $(r \restriction^l_\circ A)(\!|a|\!) = r(\!|a|\!)$
⟨*proof*⟩

lemma *vrestriction-atD*[*dest*]:
assumes $(r \restriction_\circ A)(\!|a|\!) = b$ **and** $a \in_\circ r -`_\circ A$ **and** $a \in_\circ A$
shows $b \in_\circ A$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vrestriction-atE*1[*elim*]:
assumes $(r \restriction_\circ A)(\!|a|\!) = b$
and $a \in_\circ r -`_\circ A$
and $a \in_\circ A$
and $r(\!|a|\!) = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vrestriction-atE*2[*elim*]:
assumes $x \in_\circ r \restriction_\circ A$
obtains $a\ b$ **where** $x = \langle a, b\rangle$ **and** $a \in_\circ A$ **and** $b \in_\circ A$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

Domain.

lemma *vdomain-atD*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows $\exists\, b{\in_\circ}\mathcal{R}_\circ\ r.\ r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vdomain-atE*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
obtains b **where** $b \in_\circ \mathcal{R}_\circ\ r$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

Range.

lemma *vrange-atD*:
assumes $b \in_\circ \mathcal{R}_\circ\ r$
shows $\exists\, a{\in_\circ}\mathcal{D}_\circ\ r.\ r(\!|a|\!) = b$
⟨*proof*⟩

lemma *vrange-atE*:
assumes $b \in_\circ \mathcal{R}_\circ\ r$
obtains a **where** $a \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) = b$
⟨*proof*⟩

Image.

lemma *vimage-set-eq-at*:
$\{b.\ \exists\, a{\in_\circ}A \cap_\circ \mathcal{D}_\circ\ r.\ r(\!|a|\!) = b\} = \{b.\ \exists\, a{\in_\circ}A.\ \langle a, b\rangle \in_\circ r\}$
⟨*proof*⟩

lemma *vimage-small*[*simp*]: *small* $\{b.\ \exists\, a{\in_\circ}A \cap_\circ \mathcal{D}_\circ\ r.\ r(\!|a|\!) = b\}$
⟨*proof*⟩

lemma *vimage-set-def*: $r\ `_\circ\ A = set\ \{b.\ \exists\, a{\in_\circ}A \cap_\circ \mathcal{D}_\circ\ r.\ r(\!|a|\!) = b\}$
⟨*proof*⟩

lemma *vimage-set-iff*: $b \in_\circ r\ `_\circ\ A \longleftrightarrow (\exists\, a{\in_\circ}A \cap_\circ \mathcal{D}_\circ\ r.\ r(\!|a|\!) = b)$

⟨*proof*⟩

Further derived results.

lemma *vimage-image*:
assumes $A \subseteq_\circ \mathcal{D}_\circ\ r$
shows *elts* $(r\ `_\circ\ A) = (\lambda x.\ r(\!|x|\!))\ `\ (\mathit{elts}\ A)$
⟨*proof*⟩

lemma *vsv-vinsert-match-appI*[*intro*, *simp*]:
assumes $a \notin_\circ \mathcal{D}_\circ\ r$
shows *vinsert* $\langle a, b\rangle\ r\ (\!|a|\!) = b$
⟨*proof*⟩

lemma *vsv-vinsert-no-match-appI*:
assumes $a \notin_\circ \mathcal{D}_\circ\ r$ **and** $c \in_\circ \mathcal{D}_\circ\ r$ **and** $r\ (\!|c|\!) = d$
shows *vinsert* $\langle a, b\rangle\ r\ (\!|c|\!) = d$
⟨*proof*⟩

lemma *vsv-is-vconst-onI*:
assumes $\mathcal{D}_\circ\ r = A$ **and** $\mathcal{R}_\circ\ r = \mathit{set}\ \{a\}$
shows $r = \mathit{vconst\text{-}on}\ A\ a$
⟨*proof*⟩

lemma *vsv-vdomain-vrange-vsingleton*:
assumes $\mathcal{D}_\circ\ r = \mathit{set}\ \{a\}$ **and** $\mathcal{R}_\circ\ r = \mathit{set}\{b\}$
shows $r = \mathit{set}\ \{\langle a, b\rangle\}$
⟨*proof*⟩

end

Alternative forms of existing results.

lemmas [*intro*] = *vsv.vconverse-atI*
and *vsv-vconverse-atD*[*dest*] = *vsv.vconverse-atD*[*rotated*]
and *vsv-vconverse-atE*[*elim*] = *vsv.vconverse-atE*[*rotated*]
and [*intro*, *simp*] = *vsv.vlrestriction-atI*
and *vsv-vlrestriction-atD*[*dest*] = *vsv.vlrestriction-atD*[*rotated*]
and *vsv-vlrestriction-atE*1[*elim*] = *vsv.vlrestriction-atE*1[*rotated*]
and *vsv-vlrestriction-atE*2[*elim*] = *vsv.vlrestriction-atE*2[*rotated*]
and [*intro*, *simp*] = *vsv.vrrestriction-atI*
and *vsv-vrrestriction-atD*[*dest*] = *vsv.vrrestriction-atD*[*rotated*]
and *vsv-vrrestriction-atE*1[*elim*] = *vsv.vrrestriction-atE*1[*rotated*]
and *vsv-vrrestriction-atE*2[*elim*] = *vsv.vrrestriction-atE*2[*rotated*]
and [*intro*, *simp*] = *vsv.vlrestriction-app*
and *vsv-vrestriction-atD*[*dest*] = *vsv.vrestriction-atD*[*rotated*]
and *vsv-vrestriction-atE*1[*elim*] = *vsv.vrestriction-atE*1[*rotated*]
and *vsv-vrestriction-atE*2[*elim*] = *vsv.vrestriction-atE*2[*rotated*]
and *vsv-vdomain-atD* = *vsv.vdomain-atD*[*rotated*]
and *vsv-vdomain-atE* = *vsv.vdomain-atE*[*rotated*]
and *vrange-atD* = *vsv.vrange-atD*[*rotated*]
and *vrange-atE* = *vsv.vrange-atE*[*rotated*]
and *vsv-vinsert-match-appI*[*intro*, *simp*] = *vsv.vsv-vinsert-match-appI*
and *vsv-vinsert-no-match-appI*[*intro*, *simp*] =
vsv.vsv-vinsert-no-match-appI[*rotated* 3]

Corollaries of the alternative forms of existing results.

lemma *vsv-vlrestriction-vrange*:
assumes *vsv s* **and** *vsv* $(r \restriction^l_\circ \mathcal{R}_\circ\ s)$

shows *vsv* $(r \circ_\circ s)$
⟨*proof*⟩

lemma *vsv-vunion-app-right*[*simp*]:
assumes *vsv* r **and** *vsv* s **and** *vdisjnt* $(\mathcal{D}_\circ\ r)$ $(\mathcal{D}_\circ\ s)$ **and** $x \in_\circ \mathcal{D}_\circ\ s$
shows $(r \cup_\circ s)(\!|x|\!) = s(\!|x|\!)$
⟨*proof*⟩

lemma *vsv-vunion-app-left*[*simp*]:
assumes *vsv* r **and** *vsv* s **and** *vdisjnt* $(\mathcal{D}_\circ\ r)$ $(\mathcal{D}_\circ\ s)$ **and** $x \in_\circ \mathcal{D}_\circ\ r$
shows $(r \cup_\circ s)(\!|x|\!) = r(\!|x|\!)$
⟨*proof*⟩

One-to-one relation

locale *v11* = *vsv* r **for** r +
assumes *vsv-vconverse*: *vsv* $(r^{-1}{}_\circ)$

Rules.

lemmas (in *v11*) [*intro*] = *v11-axioms*

mk-ide rf *v11-def*[*unfolded v11-axioms-def*]
|*intro v11I*[*intro*]|
|*dest v11D*[*dest*]|
|*elim v11E*[*elim*]|

Set operations.

lemma (in *v11*) *v11-vinsert*[*intro*, *simp*]:
assumes $a \notin_\circ \mathcal{D}_\circ\ r$ **and** $b \notin_\circ \mathcal{R}_\circ\ r$
shows *v11* (*vinsert* ⟨a, b⟩ r)
⟨*proof*⟩

lemma *v11-vinsertD*:
assumes *v11* (*vinsert* x r)
shows *v11* r
⟨*proof*⟩

lemma *v11-vunion*:
assumes *v11* r
and *v11* s
and *vdisjnt* $(\mathcal{D}_\circ\ r)$ $(\mathcal{D}_\circ\ s)$
and *vdisjnt* $(\mathcal{R}_\circ\ r)$ $(\mathcal{R}_\circ\ s)$
shows *v11* $(r \cup_\circ s)$
⟨*proof*⟩

lemma (in *v11*) *v11-vintersection*[*intro*, *simp*]: *v11* $(r \cap_\circ s)$
⟨*proof*⟩

lemma (in *v11*) *v11-vdiff*[*intro*, *simp*]: *v11* $(r -_\circ s)$
⟨*proof*⟩

Special properties.

lemma (in *vsv*) *vsv-valneq-v11I*:
assumes $\bigwedge x\ y.\ [\![\ x \in_\circ \mathcal{D}_\circ\ r;\ y \in_\circ \mathcal{D}_\circ\ r;\ x \neq y\ ]\!] \Longrightarrow r(\!|x|\!) \neq r(\!|y|\!)$
shows *v11* r
⟨*proof*⟩

lemma (in *vsv*) *vsv-valeq-v11I*:

assumes $\bigwedge x\ y.\ [\![\ x \in_\circ \mathcal{D}_\circ\ r;\ y \in_\circ \mathcal{D}_\circ\ r;\ r(\!|x|\!) = r(\!|y|\!)\ ]\!] \Longrightarrow x = y$
shows *v11* *r*
⟨*proof*⟩

Connections.

lemma *v11-vempty*[*simp*]: *v11* 0 ⟨*proof*⟩

lemma *v11-vsingleton*[*simp*]: *v11* (*set* {⟨*a*, *b*⟩}) ⟨*proof*⟩

lemma *v11-vdoubleton*:
assumes $a \neq c$ **and** $b \neq d$
shows *v11* (*set* {⟨*a*, *b*⟩, ⟨*c*, *d*⟩})
⟨*proof*⟩

lemma *v11-vid-on*[*simp*]: *v11* (*vid-on* *A*) ⟨*proof*⟩

lemma *v11-VLambda*[*intro*]:
assumes *inj-on* *f* (*elts* *A*)
shows *v11* (λ*a*∈∘*A*. *f* *a*)
⟨*proof*⟩

lemma *v11-vcomp*:
assumes *v11* *r* **and** *v11* *s*
shows *v11* (*r* ∘∘ *s*)
⟨*proof*⟩

context *v11*
begin

lemma *v11-vconverse*: *v11* (${r^{-1}}_\circ$) ⟨*proof*⟩

interpretation *v11* ‹${r^{-1}}_\circ$› ⟨*proof*⟩

lemma *v11-vlrestriction*[*intro*, *simp*]: *v11* (*r* ↾$^l_\circ$ *A*)
⟨*proof*⟩

lemma *v11-vrrestriction*[*intro*, *simp*]: *v11* (*r* ↾$^r_\circ$ *A*)
⟨*proof*⟩

lemma *v11-vrestriction*[*intro*, *simp*]: *v11* (*r* ↾∘ *A*)
⟨*proof*⟩

end

Further Special properties.

context *v11*
begin

lemma *v11-injective*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $b \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) = r(\!|b|\!)$
shows $a = b$
⟨*proof*⟩

lemma *v11-double-pair*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $a' \in_\circ \mathcal{D}_\circ\ r$ **and** $r(\!|a|\!) = b$ **and** $r(\!|a'|\!) = b'$
shows $a = a' \longleftrightarrow b = b'$
⟨*proof*⟩

lemma *v11-vrange-ex1-eq*: $b \in_\circ \mathcal{R}_\circ\ r \longleftrightarrow (\exists!a{\in_\circ}\mathcal{D}_\circ\ r.\ r(\!|a|\!) = b)$
⟨*proof*⟩

lemma *v11-VLambda-iff*: *inj-on f* (*elts A*) $\longleftrightarrow$ *v11* ($\lambda a{\in_\circ}A.\ f\ a$)
⟨*proof*⟩

lemma *v11-vimage-vpsubset-neq*:
assumes $A \subseteq_\circ \mathcal{D}_\circ\ r$ **and** $B \subseteq_\circ \mathcal{D}_\circ\ r$ **and** $A \neq B$
shows $r\ `_\circ\ A \neq r\ `_\circ\ B$
⟨*proof*⟩

lemma *v11-eq-iff*[*simp*]:
assumes $a \in_\circ \mathcal{D}_\circ\ r$ **and** $b \in_\circ \mathcal{D}_\circ\ r$
shows $r(\!|a|\!) = r(\!|b|\!) \longleftrightarrow a = b$
⟨*proof*⟩

lemma *v11-vcomp-vconverse*: ${r^{-1}}_\circ \circ_\circ r = \mathit{vid\text{-}on}\ (\mathcal{D}_\circ\ r)$
⟨*proof*⟩

lemma *v11-vcomp-vconverse'*: $r \circ_\circ {r^{-1}}_\circ = \mathit{vid\text{-}on}\ (\mathcal{R}_\circ\ r)$
⟨*proof*⟩

lemma *v11-vconverse-app*[*simp*]:
assumes $r(\!|a|\!) = b$ **and** $a \in_\circ \mathcal{D}_\circ\ r$
shows ${r^{-1}}_\circ(\!|b|\!) = a$
⟨*proof*⟩

lemma *v11-vconverse-app-in-vdomain*:
assumes $y \in_\circ \mathcal{R}_\circ\ r$
shows ${r^{-1}}_\circ(\!|y|\!) \in_\circ \mathcal{D}_\circ\ r$
⟨*proof*⟩

lemma *v11-app-if-vconverse-app*:
assumes $y \in_\circ \mathcal{R}_\circ\ r$ **and** ${r^{-1}}_\circ(\!|y|\!) = x$
shows $r(\!|x|\!) = y$
⟨*proof*⟩

lemma *v11-app-vconverse-app*:
assumes $a \in_\circ \mathcal{R}_\circ\ r$
shows $r(\!|{r^{-1}}_\circ(\!|a|\!)|\!) = a$
⟨*proof*⟩

lemma *v11-vconverse-app-app*:
assumes $a \in_\circ \mathcal{D}_\circ\ r$
shows ${r^{-1}}_\circ(\!|r(\!|a|\!)|\!) = a$
⟨*proof*⟩

end

lemma *v11-vlrestriction-vsubset*:
assumes *v11* ($f \restriction^l_\circ A$) **and** $B \subseteq_\circ A$
shows *v11* ($f \restriction^l_\circ B$)
⟨*proof*⟩

lemma *v11-vlrestriction-vrange*:
assumes *v11 s* **and** *v11* ($r \restriction^l_\circ \mathcal{R}_\circ\ s$)
shows *v11* ($r \circ_\circ s$)
⟨*proof*⟩

lemma *v11-vlrestriction-vcomp*:
assumes *v11* $(f \restriction_{\circ}^{l} A)$ **and** *v11* $(g \restriction_{\circ}^{l} (f \mathbin{`}_{\circ} A))$
shows *v11* $((g \circ_{\circ} f) \restriction_{\circ}^{l} A)$
⟨*proof*⟩

Alternative forms of existing results.

lemmas [*intro*, *simp*] = *v11.v11-vinsert*
and [*intro*, *simp*] = *v11.v11-vintersection*
and [*intro*, *simp*] = *v11.v11-vdiff*
and [*intro*, *simp*] = *v11.v11-vrrestriction*
and [*intro*, *simp*] = *v11.v11-vlrestriction*
and [*intro*, *simp*] = *v11.v11-vrestriction*
and [*intro*] = *v11.v11-vimage-vpsubset-neq*

2.4.5 Tools: *mk-VLambda*

ML‹

```
(* low level unfold *)
(*Designed based on an algorithm from HOL-Types_To_Sets/unoverload_def.ML*)
fun pure_unfold ctxt thms = ctxt
  |>
    (
      thms
      |> Conv.rewrs_conv
      |> Conv.try_conv
      |> K
      |> Conv.top_conv
    )
  |> Conv.fconv_rule;

val msg_args = "mk_VLambda: invalid arguments"

val vsv_VLambda_thm = @{thm vsv_VLambda};
val vsv_VLambda_match = vsv_VLambda_thm
  |> Thm.full_prop_of
  |> HOLogic.dest_Trueprop
  |> dest_comb
  |> #2;

val vdomain_VLambda_thm = @{thm vdomain_VLambda};
val vdomain_VLambda_match = vdomain_VLambda_thm
  |> Thm.full_prop_of
  |> HOLogic.dest_Trueprop
  |> HOLogic.dest_eq
  |> #1
  |> dest_comb
  |> #2;

val app_VLambda_thm = @{thm ZFC_Cardinals.beta};
val app_VLambda_match = app_VLambda_thm
  |> Thm.concl_of
  |> HOLogic.dest_Trueprop
  |> HOLogic.dest_eq
  |> #1
  |> strip_comb
  |> #2
```

```
  |> hd;

fun mk_VLabmda_thm match_t match_thm ctxt thm =
  let
    val thm_ct = Thm.cprop_of thm
    val (_, rhs_ct) = Thm.dest_equals thm_ct
      handle TERM ("dest_equals", _) => error msg_args
    val insts = Thm.match (Thm.cterm_of ctxt match_t, rhs_ct)
      handle Pattern.MATCH => error msg_args
  in
    match_thm
    |> Drule.instantiate_normalize insts
    |> pure_unfold ctxt (thm |> Thm.symmetric |> single)
  end;

val mk_VLambda_vsv =
  mk_VLabmda_thm vsv_VLambda_match vsv_VLambda_thm;
val mk_VLambda_vdomain =
  mk_VLabmda_thm vdomain_VLambda_match vdomain_VLambda_thm;
val mk_VLambda_app =
  mk_VLabmda_thm app_VLambda_match app_VLambda_thm;

val mk_VLambda_parser = Parse.thm --
  (
    Scan.repeat
      (
        (keyword‹|vsv› -- Parse_Spec.opt_thm_name "|") ||
        (keyword‹|app› -- Parse_Spec.opt_thm_name "|") ||
        (keyword‹|vdomain› -- Parse_Spec.opt_thm_name "|")
      )
  );

fun process_mk_VLambda_thm mk_VLambda_thm (b, thm) ctxt =
  let
    val thm' = mk_VLambda_thm ctxt thm
    val (res, ctxt') = ctxt
      |> Local_Theory.note (b ||> map (Attrib.check_src ctxt), single thm')
    val _ = Proof_Display.print_theorem (Position.thread_data ()) ctxt' res
  in ctxt' end;

fun folder_mk_VLambda (("|vsv", b), thm) ctxt =
      process_mk_VLambda_thm mk_VLambda_vsv (b, thm) ctxt
  | folder_mk_VLambda (("|app", b), thm) ctxt =
      process_mk_VLambda_thm mk_VLambda_app (b, thm) ctxt
  | folder_mk_VLambda (("|vdomain", b), thm) ctxt =
      process_mk_VLambda_thm mk_VLambda_vdomain (b, thm) ctxt
  | folder_mk_VLambda _ _ = error msg_args

fun process_mk_VLambda (thm, ins) ctxt =
  let
    val _ = ins |> map fst |> has_duplicates op= |> not orelse error msg_args
    val thm' = thm
      |> singleton (Attrib.eval_thms ctxt)
      |> Local_Defs.meta_rewrite_rule ctxt;
  in fold folder_mk_VLambda (map (fn x => (x, thm')) ins) ctxt end;

val _ =
  Outer_Syntax.local_theory
```

```
    command_keyword<mk_VLambda>
    "VLambda"
    (mk_VLambda_parser >> process_mk_VLambda);
```

›⟨*ML*⟩

2.5 Operations on indexed families of sets

2.5.1 Background

This section presents results about the fundamental operations on the indexed families of sets, such as unions and intersections of the indexed families of sets, disjoint unions and infinite Cartesian products.

Certain elements of the content of this section were inspired by elements of the content of [50]. However, as previously, many other results were ported (with amendments) from the main library of Isabelle/HOL.

abbreviation (*input*) *imVLambda* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *imVLambda A f* ≡ $(\lambda a{\in_\circ}A.\ f\ a)$ ‘$_\circ$ *A*

2.5.2 Intersection of an indexed family of sets

syntax -*VIFINTER* :: *pttrn* ⇒ *V* ⇒ *V* ⇒ *V* (‹(3$\bigcap_\circ$-$\in_\circ$-./ -)› [0, 0, 10] 10)

syntax-consts -*VIFINTER* ⇌ *VInter*

translations $\bigcap_\circ x{\in_\circ}A.\ f$ ⇌ *CONST VInter* (*CONST imVLambda A* ($\lambda x.\ f$))

Rules.

lemma *vifintersectionI*[*intro*]:
assumes $I \neq 0$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow a \in_\circ f\ i$
shows $a \in_\circ (\bigcap_\circ i{\in_\circ}I.\ f\ i)$
⟨*proof*⟩

lemma *vifintersectionD*[*dest*]:
assumes $a \in_\circ (\bigcap_\circ i{\in_\circ}I.\ f\ i)$ **and** $i \in_\circ I$
shows $a \in_\circ f\ i$
⟨*proof*⟩

lemma *vifintersectionE*1[*elim*]:
assumes $a \in_\circ (\bigcap_\circ i{\in_\circ}I.\ f\ i)$ **and** $a \in_\circ f\ i \Longrightarrow P$ **and** $i \notin_\circ I \Longrightarrow P$
shows *P*
⟨*proof*⟩

lemma *vifintersectionE*3[*elim*]:
assumes $a \in_\circ (\bigcap_\circ i{\in_\circ}I.\ f\ i)$
obtains $\bigwedge i.\ i{\in_\circ}I \Longrightarrow a \in_\circ f\ i$
⟨*proof*⟩

lemma *vifintersectionE*2[*elim*]:
assumes $a \in_\circ (\bigcap_\circ i{\in_\circ}I.\ f\ i)$
obtains *i* **where** $i \in_\circ I$ **and** $a \in_\circ f\ i$
⟨*proof*⟩

Set operations.

lemma *vifintersection-vempty-is*[*simp*]: $(\bigcap_\circ i{\in_\circ}0.\ f\ i) = 0$ ⟨*proof*⟩

lemma *vifintersection-vsingleton-is*[*simp*]: $(\bigcap_\circ i{\in_\circ}set\{i\}.\ f\ i) = f\ i$
⟨*proof*⟩

lemma *vifintersection-vdoubleton-is*[*simp*]: $(\bigcap_\circ i{\in_\circ}set\ \{i, j\}.\ f\ i) = f\ i \cap_\circ f\ j$
⟨*proof*⟩

lemma *vifintersection-antimono*1:

assumes $I \neq 0$ **and** $I \subseteq_\circ J$
shows $(\bigcap_\circ j \in_\circ J.\ f\ j) \subseteq_\circ (\bigcap_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifintersection-antimono2*:
assumes $I \neq 0$ **and** $I \subseteq_\circ J$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow f\ i \subseteq_\circ g\ i$
shows $(\bigcap_\circ j \in_\circ J.\ f\ j) \subseteq_\circ (\bigcap_\circ i \in_\circ I.\ g\ i)$
⟨*proof*⟩

lemma *vifintersection-vintersection*:
assumes $I \neq 0$ **and** $J \neq 0$
shows $(\bigcap_\circ i \in_\circ I.\ f\ i) \cap_\circ (\bigcap_\circ i \in_\circ J.\ f\ i) = (\bigcap_\circ i \in_\circ I \cup_\circ J.\ f\ i)$
⟨*proof*⟩

lemma *vifintersection-vintersection-family*:
assumes $I \neq 0$
shows $(\bigcap_\circ i \in_\circ I.\ A\ i) \cap_\circ (\bigcap_\circ i \in_\circ I.\ B\ i) = (\bigcap_\circ i \in_\circ I.\ A\ i \cap_\circ B\ i)$
⟨*proof*⟩

lemma *vifintersection-vunion*:
assumes $I \neq 0$ **and** $J \neq 0$
shows $(\bigcap_\circ i \in_\circ I.\ f\ i) \cup_\circ (\bigcap_\circ j \in_\circ J.\ g\ j) = (\bigcap_\circ i \in_\circ I.\ \bigcap_\circ j \in_\circ J.\ f\ i \cup_\circ g\ j)$
⟨*proof*⟩

lemma *vifintersection-vinsert-is*[*intro*, *simp*]:
assumes $I \neq 0$
shows $(\bigcap_\circ i \in_\circ vinsert\ j\ I.\ f\ i) = f\ j \cap_\circ (\bigcap_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifintersection-VPow*:
assumes $I \neq 0$
shows $VPow\ (\bigcap_\circ i \in_\circ I.\ f\ i) = (\bigcap_\circ i \in_\circ I.\ VPow\ (f\ i))$
⟨*proof*⟩

Elementary properties.

lemma *vifintersection-constant*[*intro*, *simp*]:
assumes $I \neq 0$
shows $(\bigcap_\circ y \in_\circ I.\ c) = c$
⟨*proof*⟩

lemma *vifintersection-vsubset-iff*:
assumes $I \neq 0$
shows $A \subseteq_\circ (\bigcap_\circ i \in_\circ I.\ f\ i) \longleftrightarrow (\forall\, i \in_\circ I.\ A \subseteq_\circ f\ i)$
⟨*proof*⟩

lemma *vifintersection-vsubset-lower*:
assumes $i \in_\circ I$
shows $(\bigcap_\circ i \in_\circ I.\ f\ i) \subseteq_\circ f\ i$
⟨*proof*⟩

lemma *vifintersection-vsubset-greatest*:
assumes $I \neq 0$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A \subseteq_\circ f\ i$
shows $A \subseteq_\circ (\bigcap_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifintersection-vintersection-value*:
assumes $i \in_\circ I$
shows $f\ i \cap_\circ (\bigcap_\circ i \in_\circ I.\ f\ i) = (\bigcap_\circ i \in_\circ I.\ f\ i)$

⟨*proof*⟩

lemma *vifintersection-vintersection-single*:
assumes $I \neq 0$
shows $B \cup_\circ (\bigcap_\circ i \in_\circ I.\ A\ i) = (\bigcap_\circ i \in_\circ I.\ B \cup_\circ A\ i)$
⟨*proof*⟩

Connections.

lemma *vifintersection-vrange-VLambda*: $(\bigcap_\circ i \in_\circ I.\ f\ i) = \bigcap_\circ (\mathcal{R}_\circ\ (\lambda a \in_\circ I.\ f\ a))$
⟨*proof*⟩

2.5.3 Union of an indexed family of sets

syntax -*VIFUNION* :: *pttrn* ⇒ *V* ⇒ *V* ⇒ *V* (‹(3⋃∘-∈∘-./ -)› [0, 0, 10] 10)

syntax-consts -*VIFUNION* ⇌ *VUnion*

translations $\bigcup_\circ x \in_\circ A.\ f$ ⇌ *CONST VUnion* (*CONST imVLambda A* (λx. *f*))

Rules.

lemma *vifunion-iff*: $b \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i) \longleftrightarrow (\exists\, i \in_\circ I.\ b \in_\circ f\ i)$ ⟨*proof*⟩

lemma *vifunionI*[*intro*]:
assumes $i \in_\circ I$ **and** $a \in_\circ f\ i$
shows $a \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifunionD*[*dest*]:
assumes $a \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i)$
shows $\exists\, i \in_\circ I.\ a \in_\circ f\ i$
⟨*proof*⟩

lemma *vifunionE*[*elim*!]:
assumes $a \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i)$ **and** $\bigwedge i.\ [\![\ i \in_\circ I;\ a \in_\circ f\ i\ ]\!] \Longrightarrow R$
shows *R*
⟨*proof*⟩

Set operations.

lemma *vifunion-vempty-family*[*simp*]: $(\bigcup_\circ i \in_\circ I.\ 0) = 0$ ⟨*proof*⟩

lemma *vifunion-vsingleton-is*[*simp*]: $(\bigcup_\circ i \in_\circ set\ \{i\}.\ f\ i) = f\ i$ ⟨*proof*⟩

lemma *vifunion-vsingleton-family*[*simp*]: $(\bigcup_\circ i \in_\circ I.\ set\ \{i\}) = I$ ⟨*proof*⟩

lemma *vifunion-vdoubleton*: $(\bigcup_\circ i \in_\circ set\ \{i, j\}.\ f\ i) = f\ i \cup_\circ f\ j$
⟨*proof*⟩

lemma *vifunion-mono*:
assumes $I \subseteq_\circ J$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow f\ i \subseteq_\circ g\ i$
shows $(\bigcup_\circ i \in_\circ I.\ f\ i) \subseteq_\circ (\bigcup_\circ j \in_\circ J.\ g\ j)$
⟨*proof*⟩

lemma *vifunion-vunion-is*: $(\bigcup_\circ i \in_\circ I.\ f\ i) \cup_\circ (\bigcup_\circ j \in_\circ J.\ f\ j) = (\bigcup_\circ i \in_\circ I \cup_\circ J.\ f\ i)$
⟨*proof*⟩

lemma *vifunion-vunion-family*:
$(\bigcup_\circ i \in_\circ I.\ f\ i) \cup_\circ (\bigcup_\circ i \in_\circ I.\ g\ i) = (\bigcup_\circ i \in_\circ I.\ f\ i \cup_\circ g\ i)$
⟨*proof*⟩

lemma *vifunion-vintersection*:
$(\bigcup_\circ i \in_\circ I.\ f\ i) \cap_\circ (\bigcup_\circ j \in_\circ J.\ g\ j) = (\bigcup_\circ i \in_\circ I.\ \bigcup_\circ j \in_\circ J.\ f\ i \cap_\circ g\ j)$
⟨*proof*⟩

lemma *vifunion-vinsert-is*:
$(\bigcup_\circ i \in_\circ vinsert\ j\ I.\ f\ i) = f\ j \cup_\circ (\bigcup_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifunion-VPow*: $(\bigcup_\circ i \in_\circ I.\ VPow\ (f\ i)) \subseteq_\circ VPow\ (\bigcup_\circ i \in_\circ I.\ f\ i)$ ⟨*proof*⟩

Elementary properties.

lemma *vifunion-vempty-conv*:
$0 = (\bigcup_\circ i \in_\circ I.\ f\ i) \longleftrightarrow (\forall\, i \in_\circ I.\ f\ i = 0)$
$(\bigcup_\circ i \in_\circ I.\ f\ i) = 0 \longleftrightarrow (\forall\, i \in_\circ I.\ f\ i = 0)$
⟨*proof*⟩

lemma *vifunion-constant*[*simp*]: $(\bigcup_\circ i \in_\circ I.\ c) = (if\ I = 0\ then\ 0\ else\ c)$
⟨*proof*⟩

lemma *vifunion-upper*:
assumes $i \in_\circ I$
shows $f\ i \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifunion-least*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow f\ i \subseteq_\circ C$
shows $(\bigcup_\circ i \in_\circ I.\ f\ i) \subseteq_\circ C$
⟨*proof*⟩

lemma *vifunion-absorb*:
assumes $j \in_\circ I$
shows $f\ j \cup_\circ (\bigcup_\circ i \in_\circ I.\ f\ i) = (\bigcup_\circ i \in_\circ I.\ f\ i)$
⟨*proof*⟩

lemma *vifunion-vifunion-flatten*:
$(\bigcup_\circ j \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i).\ g\ j) = (\bigcup_\circ i \in_\circ I.\ \bigcup_\circ j \in_\circ f\ i.\ g\ j)$
⟨*proof*⟩

lemma *vifunion-vsubset-iff*: $((\bigcup_\circ i \in_\circ I.\ f\ i) \subseteq_\circ B) = (\forall\, i \in_\circ I.\ f\ i \subseteq_\circ B)$ ⟨*proof*⟩

lemma *vifunion-vsingleton-eq-vrange*: $(\bigcup_\circ i \in_\circ I.\ set\ \{f\ i\}) = \mathcal{R}_\circ\ (\lambda a \in_\circ I.\ f\ a)$
⟨*proof*⟩

lemma *vball-vifunion*[*simp*]: $(\forall\, z \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i).\ P\ z) \longleftrightarrow (\forall\, x \in_\circ I.\ \forall\, z \in_\circ f\ x.\ P\ z)$
⟨*proof*⟩

lemma *vbex-vifunion*[*simp*]: $(\exists\, z \in_\circ (\bigcup_\circ i \in_\circ I.\ f\ i).\ P\ z) \longleftrightarrow (\exists\, x \in_\circ I.\ \exists\, z \in_\circ f\ x.\ P\ z)$
⟨*proof*⟩

lemma *vifunion-vintersection-index-right*[*simp*]: $(\bigcup_\circ C \in_\circ B.\ A \cap_\circ C) = A \cap_\circ \bigcup_\circ B$
⟨*proof*⟩

lemma *vifunion-vintersection-index-left*[*simp*]: $(\bigcup_\circ C \in_\circ B.\ C \cap_\circ A) = \bigcup_\circ B \cap_\circ A$
⟨*proof*⟩

lemma *vifunion-vunion-index*[*intro*, *simp*]:
assumes $B \neq 0$

shows $(\bigcap_\circ C \in_\circ B.\ A \cup_\circ C) = A \cup_\circ \bigcap_\circ B$
⟨*proof*⟩

lemma *vifunion-vintersection-single*: $B \cap_\circ (\bigcup_\circ i \in_\circ I.\ f\ i) = (\bigcup_\circ i \in_\circ I.\ B \cap_\circ f\ i)$
⟨*proof*⟩

lemma *vifunion-vifunion-flip*:
$(\bigcup_\circ b \in_\circ B.\ \bigcup_\circ a \in_\circ A.\ f\ b\ a) = (\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ f\ b\ a)$
⟨*proof*⟩

Connections.

lemma *vifunion-disjoint*: $(\bigcup_\circ C \cap_\circ A = 0) \longleftrightarrow (\forall\ B \in_\circ C.\ vdisjnt\ B\ A)$
⟨*proof*⟩

lemma *vdisjnt-vifunion-iff*:
$vdisjnt\ A\ (\bigcup_\circ i \in_\circ I.\ f\ i) \longleftrightarrow (\forall\ i \in_\circ I.\ vdisjnt\ A\ (f\ i))$
⟨*proof*⟩

lemma *vifunion-VLambda*: $(\bigcup_\circ i \in_\circ A.\ set\ \{\langle i, f\ i\rangle\}) = (\lambda a \in_\circ A.\ f\ a)$
⟨*proof*⟩

lemma *vifunion-vrange-VLambda*: $(\bigcup_\circ i \in_\circ I.\ f\ i) = \bigcup_\circ(\mathcal{R}_\circ\ (\lambda a \in_\circ I.\ f\ a))$
⟨*proof*⟩

lemma (**in** *vsv*) *vsv-vrange-vsubset-vifunion-app*:
assumes $\mathcal{D}_\circ\ r = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow r(\!|i|\!) \in_\circ A\ i$
shows $\mathcal{R}_\circ\ r \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma *v11-vlrestriction-vifintersection*:
assumes $I \neq 0$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow v11\ (f \restriction^l_\circ (A\ i))$
shows $v11\ (f \restriction^l_\circ (\bigcap_\circ i \in_\circ I.\ A\ i))$
⟨*proof*⟩

2.5.4 Additional simplification rules for indexed families of sets.

Union.

lemma *vifunion-simps*[*simp*]:
$\bigwedge a\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ vinsert\ a\ (B\ i)) =$
$\quad (if\ I{=}0\ then\ 0\ else\ vinsert\ a\ (\bigcup_\circ i \in_\circ I.\ B\ i))$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A\ i \cup_\circ B) = ((if\ I{=}0\ then\ 0\ else\ (\bigcup_\circ i \in_\circ I.\ A\ i) \cup_\circ B))$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A \cup_\circ B\ i) = ((if\ I{=}0\ then\ 0\ else\ A \cup_\circ (\bigcup_\circ i \in_\circ I.\ B\ i)))$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A\ i \cap_\circ B) = ((\bigcup_\circ i \in_\circ I.\ A\ i) \cap_\circ B)$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A \cap_\circ B\ i) = (A \cap_\circ (\bigcup_\circ i \in_\circ I.\ B\ i))$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A\ i -_\circ B) = ((\bigcup_\circ i \in_\circ I.\ A\ i) -_\circ B)$
$\bigwedge A\ B.\ (\bigcup_\circ i \in_\circ \bigcup_\circ A.\ B\ i) = (\bigcup_\circ y \in_\circ A.\ \bigcup_\circ i \in_\circ y.\ B\ i)$
⟨*proof*⟩

lemma *vifunion-simps-ext*:
$\bigwedge a\ B\ I.\ vinsert\ a\ (\bigcup_\circ i \in_\circ I.\ B\ i) =$
$\quad (if\ I{=}0\ then\ set\ \{a\}\ else\ (\bigcup_\circ i \in_\circ I.\ vinsert\ a\ (B\ i)))$
$\bigwedge A\ B\ I.\ (\bigcup_\circ i \in_\circ I.\ A\ i) \cup_\circ B = (if\ I{=}0\ then\ B\ else\ (\bigcup_\circ i \in_\circ I.\ A\ i \cup_\circ B))$
$\bigwedge A\ B\ I.\ A \cup_\circ (\bigcup_\circ i \in_\circ I.\ B\ i) = (if\ I{=}0\ then\ A\ else\ (\bigcup_\circ i \in_\circ I.\ A \cup_\circ B\ i))$
$\bigwedge A\ B\ I.\ ((\bigcup_\circ i \in_\circ I.\ A\ i) \cap_\circ B) = (\bigcup_\circ i \in_\circ I.\ A\ i \cap_\circ B)$
$\bigwedge A\ B\ I.\ ((\bigcup_\circ i \in_\circ I.\ A\ i) -_\circ B) = (\bigcup_\circ i \in_\circ I.\ A\ i -_\circ B)$
$\bigwedge A\ B.\ (\bigcup_\circ y \in_\circ A.\ \bigcup_\circ i \in_\circ y.\ B\ i) = (\bigcup_\circ i \in_\circ \bigcup_\circ A.\ B\ i)$
⟨*proof*⟩

lemma *vifunion-vball-vbex-simps*[*simp*]:
$\bigwedge A\ P.\ (\forall\, a \in_\circ \bigcup_\circ A.\ P\ a) \longleftrightarrow (\forall\, y \in_\circ A.\ \forall\, a \in_\circ y.\ P\ a)$
$\bigwedge A\ P.\ (\exists\, a \in_\circ \bigcup_\circ A.\ P\ a) \longleftrightarrow (\exists\, y \in_\circ A.\ \exists\, a \in_\circ y.\ P\ a)$
⟨*proof*⟩

Intersection.

lemma *vifintersection-simps*[*simp*]:
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A\ i \cap_\circ B) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (\bigcap_\circ i \in_\circ I.\ A\ i) \cap_\circ B)$
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A \cap_\circ B\ i) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } A \cap_\circ (\bigcap_\circ i \in_\circ I.\ B\ i))$
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A\ i -_\circ B) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (\bigcap_\circ i \in_\circ I.\ A\ i) -_\circ B)$
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A -_\circ B\ i) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } A -_\circ (\bigcup_\circ i \in_\circ I.\ B\ i))$
$\bigwedge I\ a\ B.$
$(\bigcap_\circ i \in_\circ I.\ \textit{vinsert } a\ (B\ i)) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else vinsert } a\ (\bigcap_\circ i \in_\circ I.\ B\ i))$
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A\ i \cup_\circ B) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } ((\bigcap_\circ i \in_\circ I.\ A\ i) \cup_\circ B))$
$\bigwedge I\ A\ B.\ (\bigcap_\circ i \in_\circ I.\ A \cup_\circ B\ i) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (A \cup_\circ (\bigcap_\circ i \in_\circ I.\ B\ i)))$
⟨*proof*⟩

lemma *vifintersection-simps-ext*:
$\bigwedge A\ B\ I.\ (\bigcap_\circ i \in_\circ I.\ A\ i) \cap_\circ B = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (\bigcap_\circ i \in_\circ I.\ A\ i \cap_\circ B))$
$\bigwedge A\ B\ I.\ A \cap_\circ (\bigcap_\circ i \in_\circ I.\ B\ i) = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (\bigcap_\circ i \in_\circ I.\ A \cap_\circ B\ i))$
$\bigwedge A\ B\ I.\ (\bigcap_\circ i \in_\circ I.\ A\ i) -_\circ B = (\textit{if } I = 0 \textit{ then } 0 \textit{ else } (\bigcap_\circ i \in_\circ I.\ A\ i -_\circ B))$
$\bigwedge A\ B\ I.\ A -_\circ (\bigcup_\circ i \in_\circ I.\ B\ i) = (\textit{if } I = 0 \textit{ then } A \textit{ else } (\bigcap_\circ i \in_\circ I.\ A -_\circ B\ i))$
$\bigwedge a\ B\ I.\ \textit{vinsert } a\ (\bigcap_\circ i \in_\circ I.\ B\ i) =$
$(\textit{if } I = 0 \textit{ then set } \{a\} \textit{ else } (\bigcap_\circ i \in_\circ I.\ \textit{vinsert } a\ (B\ i)))$
$\bigwedge A\ B\ I.\ ((\bigcap_\circ i \in_\circ I.\ A\ i) \cup_\circ B) = (\textit{if } I = 0 \textit{ then } B \textit{ else } (\bigcap_\circ i \in_\circ I.\ A\ i \cup_\circ B))$
$\bigwedge A\ B\ I.\ A \cup_\circ (\bigcap_\circ i \in_\circ I.\ B\ i) = (\textit{if } I = 0 \textit{ then } A \textit{ else } (\bigcap_\circ i \in_\circ I.\ A \cup_\circ B\ i))$
⟨*proof*⟩

2.5.5 Knowledge transfer: union and intersection of indexed families

lemma *SUP-vifunion*: $(\textit{SUP } \xi \in \textit{elts } \alpha.\ A\ \xi) = (\bigcup_\circ \xi \in_\circ \alpha.\ A\ \xi)$
⟨*proof*⟩

lemma *INF-vifintersection*: $(\textit{INF } \xi \in \textit{elts } \alpha.\ A\ \xi) = (\bigcap_\circ \xi \in_\circ \alpha.\ A\ \xi)$
⟨*proof*⟩

lemmas *Ord-induct3'*[*consumes* 1, *case-names* 0 *succ Limit*, *induct type*: *V*] =
Ord-induct3[*unfolded SUP-vifunion*]

lemma *Limit-vifunion-def*[*simp*]:
assumes *Limit* α
shows $(\bigcup_\circ \xi \in_\circ \alpha.\ \xi) = \alpha$
⟨*proof*⟩

lemmas-with[*unfolded SUP-vifunion INF-vifintersection*]:
TC = *ZFC-Cardinals.TC*
and *rank-Sup* = *ZFC-Cardinals.rank-Sup*
and *TC-def* = *ZFC-Cardinals.TC-def*
and *Ord-equality* = *ZFC-in-HOL.Ord-equality*
and *Aleph-Limit* = *ZFC-Cardinals.Aleph-Limit*
and *rank* = *ZFC-Cardinals.rank*
and *Vset* = *ZFC-in-HOL.Vset*
and *mult* = *Kirby.mult*
and *Aleph-def* = *ZFC-Cardinals.Aleph-def*
and *times-V-def* = *Kirby.times-V-def*
and *mult-Limit* = *Kirby.mult-Limit*
and *Vfrom* = *ZFC-in-HOL.Vfrom*

and *Vfrom-def* = *ZFC-in-HOL.Vfrom-def*
and *rank-def* = *ZFC-Cardinals.rank-def*
and *add-Limit* = *Kirby.add-Limit*
and *Limit-Vfrom-eq* = *ZFC-in-HOL.Limit-Vfrom-eq*
and *VSigma-def* = *ZFC-Cardinals.VSigma-def*
and *add-Sup-distrib-id* = *Kirby.add-Sup-distrib-id*
and *Limit-add-Sup-distrib* = *Kirby.Limit-add-Sup-distrib*
and *TC-mult* = *Kirby.TC-mult*
and *add-Sup-distrib* = *Kirby.add-Sup-distrib*

2.5.6 Disjoint union

See the main library of *ZFC in HOL* for further information and elementary properties.

syntax -*VSIGMA* :: *pttrn* ⇒ *V* ⇒ *V* ⇒ *V* (‹(3∐∘-∈∘-./ -)› [0, 0, 10] 10)

syntax-consts -*VSIGMA* ⇌ *VSigma*

translations ∐∘*i*∈∘*I*. *A* ⇌ *CONST VSigma I* (λ*i*. *A*)

Further rules.

lemma *vdunion-def*: (∐∘*i*∈∘*I*. *A i*) = (⋃∘*i*∈∘*I*. ⋃∘*x*∈∘*A i*. *set* {⟨*i*, *x*⟩})
⟨*proof*⟩

lemma *vdunionI*:
assumes *ix* = ⟨*i*, *x*⟩ **and** *i* ∈∘ *I* **and** *x* ∈∘ *A i*
shows *ix* ∈∘ (∐∘*i*∈∘*I*. *A i*)
⟨*proof*⟩

lemmas *vdunionD* = *VSigmaD*1 *VSigmaD*2
and *vdunionE* = *VSigmaE*

Set operations.

lemma *vdunion-vsingleton*: (∐∘*i*∈∘*set*{*c*}. *A i*) = *set* {*c*} ×∘ *A c* ⟨*proof*⟩

lemma *vdunion-vdoubleton*:
(∐∘*i*∈∘*set*{*a*, *b*}. *A i*) = *set* {*a*} ×∘ *A a* ∪∘ *set* {*b*} ×∘ *A b*
⟨*proof*⟩

Connections.

lemma *vdunion-vsum*: (∐∘*i*∈∘*set*{0, 1}. *if i*=0 *then A else B*) = *A* ⊎ *B*
⟨*proof*⟩

2.5.7 Canonical injection

definition *vcinjection* :: (*V* ⇒ *V*) ⇒ *V* ⇒ *V*
where *vcinjection A i* = (λ*x*∈∘*A i*. ⟨*i*, *x*⟩)

Rules.

mk-VLambda *vcinjection-def*
|*vsv vcinjection-vsv*[*intro*]|
|*vdomain vcinjection-vdomain*[*simp*]|
|*app vcinjection-app*[*simp*, *intro*]|

Elementary results.

lemma *vcinjection-vrange-vsubset*:
assumes *i* ∈∘ *I*

shows $\mathcal{R}_\circ\ (\mathit{vcinjection}\ A\ i) \subseteq_\circ (\coprod_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma *vcinjection-vrange*:
assumes $i \in_\circ I$ **and** $\bigwedge j.\ j \in_\circ I \Longrightarrow A\ j \neq 0$
shows $\mathcal{R}_\circ\ (\mathit{vcinjection}\ A\ i) = (\bigcup_\circ x \in_\circ A\ i.\ \mathit{set}\ \{\langle i, x\rangle\})$
⟨*proof*⟩

2.5.8 Infinite Cartesian product

definition $\mathit{vproduct} :: V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where $\mathit{vproduct}\ I\ A = \mathit{set}\ \{f.\ \mathit{vsv}\ f \wedge \mathcal{D}_\circ\ f = I \wedge (\forall i \in_\circ I.\ f(\!|i|\!) \in_\circ A\ i)\}$

syntax $\text{-}\mathit{VPRODUCT} :: \mathit{pttrn} \Rightarrow V \Rightarrow V \Rightarrow V$ $(\langle(3\prod_\circ \text{-} \in_\circ \text{-}./\ \text{-})\rangle\ [0, 0, 10]\ 10)$

syntax-consts $\text{-}\mathit{VPRODUCT} \rightleftharpoons \mathit{vproduct}$

translations $\prod_\circ i \in_\circ I.\ A \rightleftharpoons \mathit{CONST}\ \mathit{vproduct}\ I\ (\lambda i.\ A)$

lemma *small-vproduct*[*simp*]:
$\mathit{small}\ \{f.\ \mathit{vsv}\ f \wedge \mathcal{D}_\circ\ f = I \wedge (\forall i \in_\circ I.\ f(\!|i|\!) \in_\circ A\ i)\}$
(**is** ‹*small ?A*›)
⟨*proof*⟩

Rules.

lemma *vproductI*[*intro*!]:
assumes $\mathit{vsv}\ f$ **and** $\mathcal{D}_\circ\ f = I$ **and** $\forall i \in_\circ I.\ f(\!|i|\!) \in_\circ A\ i$
shows $f \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma *vproductD*[*dest*]:
assumes $f \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$
shows $\mathit{vsv}\ f$
and $\mathcal{D}_\circ\ f = I$
and $\forall i \in_\circ I.\ f(\!|i|\!) \in_\circ A\ i$
⟨*proof*⟩

lemma *vproductE*[*elim*!]:
assumes $f \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$
obtains $\mathit{vsv}\ f$ **and** $\mathcal{D}_\circ\ f = I$ **and** $\forall i \in_\circ I.\ f(\!|i|\!) \in_\circ A\ i$
⟨*proof*⟩

Set operations.

lemma *vproduct-index-vempty*[*simp*]: $(\prod_\circ i \in_\circ 0.\ A\ i) = \mathit{set}\ \{0\}$
⟨*proof*⟩

lemma *vproduct-vsingletonI*:
assumes $f(\!|c|\!) \in_\circ A\ c$ **and** $f = \mathit{set}\ \{\langle c, f(\!|c|\!)\rangle\}$
shows $f \in_\circ (\prod_\circ i \in_\circ \mathit{set}\{c\}.\ A\ i)$
⟨*proof*⟩

lemma *vproduct-vsingletonD*:
assumes $f \in_\circ (\prod_\circ i \in_\circ \mathit{set}\{c\}.\ A\ i)$
shows $\mathit{vsv}\ f$ **and** $f(\!|c|\!) \in_\circ A\ c$ **and** $f = \mathit{set}\ \{\langle c, f(\!|c|\!)\rangle\}$
⟨*proof*⟩

lemma *vproduct-vsingletonE*:
assumes $f \in_\circ (\prod_\circ i \in_\circ \mathit{set}\{c\}.\ A\ i)$

obtains *vsv f* **and** $f(\!|c|\!) \in_\circ A\ c$ **and** $f = set\ \{\langle c, f(\!|c|\!)\rangle\}$
⟨*proof*⟩

lemma *vproduct-vsingleton-iff*:
$f \in_\circ (\prod_\circ i \in_\circ set\{c\}.\ A\ i) \longleftrightarrow f(\!|c|\!) \in_\circ A\ c \wedge f = set\ \{\langle c, f(\!|c|\!)\rangle\}$
⟨*proof*⟩

lemma *vproduct-vdoubletonI*[*intro*]:
assumes *vsv f*
and $f(\!|a|\!) \in_\circ A\ a$
and $f(\!|b|\!) \in_\circ A\ b$
and $\mathcal{D}_\circ\ f = set\ \{a, b\}$
and $\mathcal{R}_\circ\ f \subseteq_\circ A\ a \cup_\circ A\ b$
shows $f \in_\circ (\prod_\circ i \in_\circ set\ \{a, b\}.\ A\ i)$
⟨*proof*⟩

lemma *vproduct-vdoubletonD*[*dest*]:
assumes $f \in_\circ (\prod_\circ i \in_\circ set\{a, b\}.\ A\ i)$
shows *vsv f*
and $f(\!|a|\!) \in_\circ A\ a$
and $f(\!|b|\!) \in_\circ A\ b$
and $\mathcal{D}_\circ\ f = set\ \{a, b\}$
and $f = set\ \{\langle a, f(\!|a|\!)\rangle, \langle b, f(\!|b|\!)\rangle\}$
⟨*proof*⟩

lemma *vproduct-vdoubletonE*:
assumes $f \in_\circ (\prod_\circ i \in_\circ set\{a, b\}.\ A\ i)$
obtains *vsv f*
and $f(\!|a|\!) \in_\circ A\ a$
and $f(\!|b|\!) \in_\circ A\ b$
and $\mathcal{D}_\circ\ f = set\ \{a, b\}$
and $f = set\ \{\langle a, f(\!|a|\!)\rangle, \langle b, f(\!|b|\!)\rangle\}$
⟨*proof*⟩

lemma *vproduct-vdoubleton-iff*:
$f \in_\circ (\prod_\circ i \in_\circ set\ \{a, b\}.\ A\ i) \longleftrightarrow$
vsv f $\wedge$
$f(\!|a|\!) \in_\circ A\ a \wedge$
$f(\!|b|\!) \in_\circ A\ b \wedge$
$\mathcal{D}_\circ\ f = set\ \{a, b\} \wedge$
$f = set\ \{\langle a, f(\!|a|\!)\rangle, \langle b, f(\!|b|\!)\rangle\}$
⟨*proof*⟩

Elementary properties.

lemma *vproduct-eq-vemptyI*:
assumes $i \in_\circ I$ **and** $A\ i = 0$
shows $(\prod_\circ i \in_\circ I.\ A\ i) = 0$
⟨*proof*⟩

lemma *vproduct-eq-vemptyD*:
assumes $(\prod_\circ i \in_\circ I.\ A\ i) \neq 0$
shows $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \neq 0$
⟨*proof*⟩

lemma *vproduct-vrange*:
assumes $f \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$
shows $\mathcal{R}_\circ\ f \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma *vproduct-vsubset-VPow*: $(\prod_\circ i \in_\circ I.\ A\ i) \subseteq_\circ VPow\ (I \times_\circ (\bigcup_\circ i \in_\circ I.\ A\ i))$
⟨*proof*⟩

lemma *VLambda-in-vproduct*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow f\ i \in_\circ A\ i$
shows $(\lambda i \in_\circ I.\ f\ i) \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$
⟨*proof*⟩

lemma *vproduct-cong*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow f\ i = g\ i$
shows $(\prod_\circ i \in_\circ I.\ f\ i) = (\prod_\circ i \in_\circ I.\ g\ i)$
⟨*proof*⟩

lemma *vproduct-ex-in-vproduct*:
assumes $x \in_\circ (\prod_\circ i \in_\circ J.\ A\ i)$ **and** $J \subseteq_\circ I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \neq 0$
obtains X **where** $X \in_\circ (\prod_\circ i \in_\circ I.\ A\ i)$ **and** $x = X \restriction^l_\circ J$
⟨*proof*⟩

lemma *vproduct-vsingleton-def*: $(\prod_\circ i \in_\circ set\ \{j\}.\ A\ i) = (\prod_\circ i \in_\circ set\ \{j\}.\ A\ j)$
⟨*proof*⟩

lemma *vprojection-in-VUnionI*:
assumes $A \subseteq_\circ (\prod_\circ i \in_\circ I.\ F\ i)$ **and** $f \in_\circ A$ **and** $i \in_\circ I$
shows $f(\!|i|\!) \in_\circ \bigcup_\circ(\bigcup_\circ(\bigcup_\circ A))$
⟨*proof*⟩

2.5.9 Projection

definition *vprojection* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$
where $vprojection\ I\ A\ i = (\lambda f \in_\circ (\prod_\circ i \in_\circ I.\ A\ i).\ f(\!|i|\!))$

Rules.

mk-VLambda *vprojection-def*
|*vsv vprojection-vsv*[*intro*]|
|*vdomain vprojection-vdomain*[*simp*]|
|*app vprojection-app*[*simp*, *intro*]|

Elementary results.

lemma *vprojection-vrange-vsubset*:
assumes $i \in_\circ I$
shows $\mathcal{R}_\circ\ (vprojection\ I\ A\ i) \subseteq_\circ A\ i$
⟨*proof*⟩

lemma *vprojection-vrange*:
assumes $i \in_\circ I$ **and** $\bigwedge j.\ j \in_\circ I \Longrightarrow A\ j \neq 0$
shows $\mathcal{R}_\circ\ (vprojection\ I\ A\ i) = A\ i$
⟨*proof*⟩

2.5.10 Cartesian power of a set

definition *vcpower* :: $V \Rightarrow V \Rightarrow V$ (**infixr** $\langle \widehat{}_\times \rangle$ 80)
where $A\ \widehat{}_\times\ n = (\prod_\circ i \in_\circ n.\ A)$

Rules.

lemma *vcpowerI*[*intro*]:
assumes $f \in_\circ (\prod_\circ i \in_\circ n.\ A)$
shows $f \in_\circ (A\ \widehat{}_\times\ n)$

⟨*proof*⟩

lemma *vcpowerD*[*dest*]:
assumes $f \in_\circ (A \mathbin{\widehat{\times}} n)$
shows $f \in_\circ (\prod_\circ i \in_\circ n.\ A)$
⟨*proof*⟩

lemma *vcpowerE*[*elim*]:
assumes $f \in_\circ (A \mathbin{\widehat{\times}} n)$ **and** $f \in_\circ (\prod_\circ i \in_\circ n.\ A) \Longrightarrow P$
shows P
⟨*proof*⟩

Set operations.

lemma *vcpower-index-vempty*[*simp*]: $A \mathbin{\widehat{\times}} 0 = set\ \{0\}$
⟨*proof*⟩

lemma *vcpower-of-vempty*:
assumes $n \neq 0$
shows $0 \mathbin{\widehat{\times}} n = 0$
⟨*proof*⟩

lemma *vcpower-vsubset-mono*:
assumes $A \subseteq_\circ B$
shows $A \mathbin{\widehat{\times}} n \subseteq_\circ B \mathbin{\widehat{\times}} n$
⟨*proof*⟩

Connections.

lemma *vcpower-vdomain*:
assumes $f \in_\circ (A \mathbin{\widehat{\times}} n)$
shows $\mathcal{D}_\circ f = n$
⟨*proof*⟩

lemma *vcpower-vrange*:
assumes $f \in_\circ (A \mathbin{\widehat{\times}} n)$
shows $\mathcal{R}_\circ f \subseteq_\circ A$
⟨*proof*⟩

2.6 Equipollence

2.6.1 Background

The section presents an adaption of the existing framework *Equipollence* in the main library of Isabelle/HOL to the type V.

Some of content of this theory was ported directly (with amendments) from the theory *HOL–Library.Equipollence* in the main library of Isabelle/HOL.

2.6.2 *veqpoll*

abbreviation *veqpoll* :: $V \Rightarrow V \Rightarrow$ *bool* (**infixl** ‹$\approx_\circ$› 50)
  where $A \approx_\circ B \equiv$ *elts* $A \approx$ *elts* B

Rules

lemma (in $v11$**)** $v11$-*veqpollI*[*intro*]:
  assumes $\mathcal{D}_\circ\ r = A$ **and** $\mathcal{R}_\circ\ r = B$
  shows $A \approx_\circ B$
  ⟨*proof*⟩

lemmas $v11$-*veqpollI*[*intro*] = $v11$.$v11$-*veqpollI*

lemma $v11$-*veqpollE*[*elim*]:
  assumes $A \approx_\circ B$
  obtains f **where** $v11\ f$ **and** $\mathcal{D}_\circ\ f = A$ **and** $\mathcal{R}_\circ\ f = B$
⟨*proof*⟩

Set operations.

lemma *veqpoll-vsingleton*: *set* $\{x\} \approx_\circ$ *set* $\{y\}$
  ⟨*proof*⟩

lemma *veqpoll-vinsert*:
  assumes $A \approx_\circ B$ **and** $a \notin_\circ A$ **and** $b \notin_\circ B$
  shows *vinsert* $a\ A \approx_\circ$ *vinsert* $b\ B$
  ⟨*proof*⟩

lemma *veqpoll-pair*:
  assumes $a \neq b$ **and** $c \neq d$
  shows *set* $\{a, b\} \approx_\circ$ *set* $\{c, d\}$
  ⟨*proof*⟩

lemma *veqpoll-vpair*:
  assumes $a \neq b$ **and** $c \neq d$
  shows $\langle a, b\rangle \approx_\circ \langle c, d\rangle$
  ⟨*proof*⟩

2.6.3 *vlepoll*

abbreviation *vlepoll* :: $V \Rightarrow V \Rightarrow$ *bool* (**infixl** ‹$\lesssim_\circ$› 50)
  where $A \lesssim_\circ B \equiv$ *elts* $A \lesssim$ *elts* B

Set operations.

lemma *vlepoll-vsubset*:
  assumes $A \subseteq_\circ B$
  shows $A \lesssim_\circ B$
  ⟨*proof*⟩

Special properties.

lemma *vlepoll-singleton-vinsert*: *set* $\{x\} \lesssim_\circ$ *vinsert* y A
⟨*proof*⟩

lemma *vlepoll-vempty-iff*[*simp*]: $A \lesssim_\circ 0 \longleftrightarrow A = 0$ ⟨*proof*⟩

2.6.4 *vlespoll*

abbreviation *vlesspoll* :: $V \Rightarrow V \Rightarrow$ *bool* (**infixl** ‹$\prec_\circ$› 50)
where $A \prec_\circ B \equiv$ *elts* $A \prec$ *elts* B

lemma *vlesspoll-def*: $A \prec_\circ B = (A \lesssim_\circ B \wedge \neg(A \approx_\circ B))$ ⟨*proof*⟩

Rules.

lemmas *vlesspollI*[*intro*] = *vlesspoll-def*[*THEN iffD2*]

lemmas *vlesspollD*[*dest*] = *vlesspoll-def*[*THEN iffD1*]

lemma *vlesspollE*[*elim*]:
assumes $A \prec_\circ B$ **and** $A \lesssim_\circ B \Longrightarrow \neg(A \approx_\circ B) \Longrightarrow P$
shows P
⟨*proof*⟩

lemma (in *v*11) *v*11-*vlepollI*[*intro*]:
assumes $\mathcal{D}_\circ\ r = A$ **and** $\mathcal{R}_\circ\ r \subseteq_\circ B$
shows $A \lesssim_\circ B$
⟨*proof*⟩

lemmas *v*11-*vlepollI*[*intro*] = *v*11.*v*11-*vlepollI*

lemma *v*11-*vlepollE*[*elim*]:
assumes $A \lesssim_\circ B$
obtains f **where** *v*11 f **and** $\mathcal{D}_\circ\ f = A$ **and** $\mathcal{R}_\circ\ f \subseteq_\circ B$
⟨*proof*⟩

2.7 Cardinality

2.7.1 Background

The section presents further results about the cardinality of terms of the type V. The emphasis of this work, however, is on the development of a theory of finite sets internalized in the type V.

Many of the results that are presented in this section were carried over (with amendments) from the theory *Finite* in the main library of Isabelle/HOL.

declare *One-nat-def*[*simp del*]

2.7.2 Cardinality of an arbitrary set

Elementary properties.

lemma *vcard-veqpoll*: $vcard\ A = vcard\ B \longleftrightarrow A \approx_\circ B$
⟨*proof*⟩

lemma *vcard-vlepoll*: $vcard\ A \leq vcard\ B \longleftrightarrow A \lesssim_\circ B$
⟨*proof*⟩

lemma *vcard-vempty*: $vcard\ A = 0 \longleftrightarrow A = 0$
⟨*proof*⟩

lemmas *vcard-vemptyD* = *vcard-vempty*[*THEN iffD*1]
and *vcard-vemptyI* = *vcard-vempty*[*THEN iffD*2]

lemma *vcard-neq-vempty*: $vcard\ A \neq 0_{\mathbf{N}} \longleftrightarrow A \neq 0_{\mathbf{N}}$
⟨*proof*⟩

lemmas *vcard-neq-vemptyD* = *vcard-neq-vempty*[*THEN iffD*1]
and *vcard-neq-vemptyI* = *vcard-neq-vempty*[*THEN iffD*2]

Set operations.

lemma *vcard-mono*:
assumes $A \subseteq_\circ B$
shows $vcard\ A \leq vcard\ B$
⟨*proof*⟩

lemma *vcard-vinsert-in*[*simp*]:
assumes $a \in_\circ A$
shows $vcard\ (vinsert\ a\ A) = vcard\ A$
⟨*proof*⟩

lemma *vcard-vintersection-left*: $vcard\ (A \cap_\circ B) \leq vcard\ A$
⟨*proof*⟩

lemma *vcard-vintersection-right*: $vcard\ (A \cap_\circ B) \leq vcard\ B$
⟨*proof*⟩

lemma *vcard-vunion*:
assumes *vdisjnt A B*
shows $vcard\ (A \cup_\circ B) = vcard\ A \oplus vcard\ B$
⟨*proof*⟩

lemma *vcard-vdiff*[*simp*]: $vcard\ (A -_\circ B) \oplus vcard\ (A \cap_\circ B) = vcard\ A$
⟨*proof*⟩

lemma *vcard-vdiff-vsubset*:
assumes $B \subseteq_\circ A$
shows $vcard\ (A -_\circ B) \oplus vcard\ B = vcard\ A$
⟨*proof*⟩

Connections.

lemma (in *vsv*) *vsv-vcard-vdomain*: $vcard\ (\mathcal{D}_\circ\ r) = vcard\ r$
⟨*proof*⟩

Special properties.

lemma *vcard-vunion-vintersection*:
$vcard\ (A \cup_\circ B) \oplus vcard\ (A \cap_\circ B) = vcard\ A \oplus vcard\ B$
⟨*proof*⟩

2.7.3 Finite sets

abbreviation *vfinite* :: $V \Rightarrow bool$
where $vfinite\ A \equiv finite\ (elts\ A)$

lemma *vfinite-def*: $vfinite\ A \longleftrightarrow (\exists\, n{\in_\circ}\omega.\ n \approx_\circ A)$
⟨*proof*⟩

Rules.

lemmas *vfiniteI*[*intro*!] = *vfinite-def*[*THEN iffD2*]

lemmas *vfiniteD*[*dest*!] = *vfinite-def*[*THEN iffD1*]

lemma *vfiniteE1*[*elim*!]:
assumes $vfinite\ A$ **and** $\exists\, n{\in_\circ}\omega.\ n \approx_\circ A \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vfiniteE2*[*elim*]:
assumes $vfinite\ A$
obtains n **where** $n \in_\circ \omega$ **and** $n \approx_\circ A$
⟨*proof*⟩

Elementary properties.

lemma *veqpoll-omega-vcard*[*intro*, *simp*]:
assumes $n \in_\circ \omega$ **and** $n \approx_\circ A$
shows $vcard\ A = n$
⟨*proof*⟩

lemma (in *vsv*) *vfinite-vimage*[*intro*]:
assumes $vfinite\ A$
shows $vfinite\ (r\ `_\circ\ A)$
⟨*proof*⟩

lemmas [*intro*] = *vsv.vfinite-vimage*

lemma *vfinite-veqpoll-trans*:
assumes $vfinite\ A$ **and** $A \approx_\circ B$
shows $vfinite\ B$
⟨*proof*⟩

lemma *vfinite-vlepoll-trans*:

assumes *vfinite A* **and** $B \lesssim_\circ A$
shows *vfinite B*
⟨*proof*⟩

lemma *vfinite-vlesspoll-trans*:
assumes *vfinite A* **and** $B \prec_\circ A$
shows *vfinite B*
⟨*proof*⟩

Induction.

lemma *vfinite-induct*[*consumes* 1, *case-names vempty vinsert*]:
assumes *vfinite F*
and *P* 0
and $\bigwedge x\ F.\ [\![\mathit{vfinite}\ F;\ x \notin_\circ F;\ P\ F]\!] \Longrightarrow P\ (\mathit{vinsert}\ x\ F)$
shows *P F*
⟨*proof*⟩

Set operations.

lemma *vfinite-vempty*[*simp*]: *vfinite* $(0_{\mathbb{N}})$ ⟨*proof*⟩

lemma *vfinite-vsingleton*[*simp*]: *vfinite* (*set* $\{x\}$) ⟨*proof*⟩

lemma *vfinite-vdoubleton*[*simp*]: *vfinite* (*set* $\{x, y\}$) ⟨*proof*⟩

lemma *vfinite-vinsert*:
assumes *vfinite F*
shows *vfinite* (*vinsert x F*)
⟨*proof*⟩

lemma *vfinite-vinsertD*:
assumes *vfinite* (*vinsert x F*)
shows *vfinite F*
⟨*proof*⟩

lemma *vfinite-vsubset*:
assumes *vfinite B* **and** $A \subseteq_\circ B$
shows *vfinite A*
⟨*proof*⟩

lemma *vfinite-vunion*: *vfinite* $(A \cup_\circ B) \longleftrightarrow$ *vfinite A* $\wedge$ *vfinite B*
⟨*proof*⟩

lemma *vfinite-vunionI*:
assumes *vfinite A* **and** *vfinite B*
shows *vfinite* $(A \cup_\circ B)$
⟨*proof*⟩

lemma *vfinite-vunionD*:
assumes *vfinite* $(A \cup_\circ B)$
shows *vfinite A* **and** *vfinite B*
⟨*proof*⟩

lemma *vfinite-vintersectionI*:
assumes *vfinite A* **and** *vfinite B*
shows *vfinite* $(A \cap_\circ B)$
⟨*proof*⟩

lemma *vfinite-VPowI*:

assumes *vfinite A*
shows *vfinite* (*VPow A*)
⟨*proof*⟩

Connections.

lemma *vfinite-vcard-vfinite*: *vfinite* (*vcard A*) = *vfinite A*
⟨*proof*⟩

lemma *vfinite-vcard-omega-iff*: *vfinite A* ⟷ *vcard A* $\in_\circ$ ω
⟨*proof*⟩

lemmas *vcard-vfinite-omega* = *vfinite-vcard-omega-iff*[*THEN iffD*2]
and *vfinite-vcard-omega* = *vfinite-vcard-omega-iff*[*THEN iffD*1]

lemma *vfinite-csucc*[*intro*, *simp*]:
assumes *vfinite A*
shows *csucc* (*vcard A*) = *succ* (*vcard A*)
⟨*proof*⟩

lemmas [*intro*, *simp*] = *finite-csucc*

Previous connections.

lemma *vcard-vsingleton*[*simp*]: *vcard* (*set* {*a*}) = $1_{\mathbb{N}}$ ⟨*proof*⟩

lemma *vfinite-vcard-vinsert-nin*[*simp*]:
assumes *vfinite A* **and** *a* $\notin_\circ$ *A*
shows *vcard* (*vinsert a A*) = *csucc* (*vcard A*)
⟨*proof*⟩

2.8 Further results about ordinal numbers

2.8.1 Background

The subsection presents several results about ordinal numbers. The primary general reference for this section is [59].

lemmas [*intro*] = *Limit-is-Ord Ord-in-Ord*

2.8.2 Further ordinal arithmetic and inequalities

lemma *Ord-succ-mono*:
assumes *Ord* β **and** $\alpha \in_\circ \beta$
shows *succ* $\alpha \in_\circ$ *succ* β
⟨*proof*⟩

lemma *Limit-right-Limit-mult*:
— Based on Theorem 8.23 in [59].
assumes *Ord* α **and** *Limit* β **and** $0 \in_\circ \alpha$
shows *Limit* $(\alpha * \beta)$
⟨*proof*⟩

lemma *Limit-left-Limit-mult*:
assumes *Limit* α **and** *Ord* β **and** $0 \in_\circ \beta$
shows *Limit* $(\alpha * \beta)$
⟨*proof*⟩

lemma *zero-if-Limit-eq-Limit-plus-vnat*:
assumes *Limit* α **and** *Limit* β **and** $\alpha = \beta + n$ **and** $n \in_\circ \omega$
shows $n = 0$
⟨*proof*⟩

lemma *Ord-vsubset-closed*:
assumes *Ord* α **and** *Ord* γ **and** $\alpha \subseteq_\circ \beta$ **and** $\beta \in_\circ \gamma$
shows $\alpha \in_\circ \gamma$
⟨*proof*⟩

lemma
— Based on Exercise 1, page 53 in [59].
assumes *Ord* α **and** *Ord* γ **and** $\alpha + \beta \in_\circ \gamma$
shows *Ord-plus-Ord-closed-augend*: $\alpha \in_\circ \gamma$
and *Ord-plus-Ord-closed-addend*: $\beta \in_\circ \gamma$
⟨*proof*⟩

lemma *Ord-ex*1*-Limit-plus-in-omega*:
— Based on Theorem 8.13 in [59].
assumes *Ord* α **and** $\omega \subseteq_\circ \alpha$
shows $\exists!\beta.\ \exists!n.\ n \in_\circ \omega \wedge$ *Limit* $\beta \wedge \alpha = \beta + n$
⟨*proof*⟩

lemma *not-Limit-if-in-Limit-plus-omega*:
assumes *Limit* α **and** $\alpha \in_\circ \beta$ **and** $\beta \in_\circ \alpha + \omega$
shows ~*Limit* β
⟨*proof*⟩

lemma *Limit-plus-omega-vsubset-Limit*:
assumes *Limit* α **and** *Limit* β **and** $\alpha \in_\circ \beta$
shows $\alpha + \omega \subseteq_\circ \beta$
⟨*proof*⟩

lemma *Limit-plus-nat-in-Limit*:
assumes *Limit* α **and** *Limit* β **and** $\alpha \in_\circ \beta$
shows $\alpha + a_{\mathbf{N}} \in_\circ \beta$
⟨*proof*⟩

lemma *omega2-vsubset-Limit*:
assumes *Limit* α **and** $\omega \in_\circ \alpha$
shows $\omega + \omega \subseteq_\circ \alpha$
⟨*proof*⟩

2.9 Finite sequences

2.9.1 Background

The section presents a theory of finite sequences internalized in the type V. The content of this subsection was inspired by and draws on many ideas from the content of the theory *List* in the main library of Isabelle/HOL.

2.9.2 Definition and common properties

A finite sequence is defined as a single-valued binary relation whose domain is an initial segment of the set of natural numbers.

locale *vfsequence* = *vsv xs* **for** *xs* +
  assumes *vfsequence-vdomain-in-omega*: $\mathcal{D}_\circ\ xs \in_\circ \omega$

locale *vfsequence-pair* = r_1: *vfsequence* xs_1 + r_2: *vfsequence* xs_2 **for** xs_1 xs_2

Rules.

lemmas [*intro*] = *vfsequence.axioms*(1)

lemma *vfsequenceI*[*intro*]:
  assumes *vsv xs* **and** $\mathcal{D}_\circ\ xs \in_\circ \omega$
  shows *vfsequence xs*
  ⟨*proof*⟩

lemma *vfsequenceD*[*dest*]:
  assumes *vfsequence xs*
  shows $\mathcal{D}_\circ\ xs \in_\circ \omega$
  ⟨*proof*⟩

lemma *vfsequenceE*[*elim*]:
  assumes *vfsequence xs* **and** $\mathcal{D}_\circ\ xs \in_\circ \omega \Longrightarrow P$
  shows P
  ⟨*proof*⟩

lemma *vfsequence-iff*: *vfsequence xs* $\longleftrightarrow$ *vsv xs* $\wedge$ $\mathcal{D}_\circ\ xs \in_\circ \omega$
  ⟨*proof*⟩

Elementary properties.

lemma (**in** *vfsequence*) *vfsequence-vdomain*: $\mathcal{D}_\circ\ xs$ = *vcard xs*
  ⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vcard-in-omega*[*simp*]: *vcard xs* $\in_\circ \omega$
  ⟨*proof*⟩

Set operations.

lemma *vfsequence-vempty*[*intro*, *simp*]: *vfsequence* 0 ⟨*proof*⟩

lemma *vfsequence-vsingleton*[*intro*, *simp*]: *vfsequence* (*set* {⟨0, *a*⟩})
  ⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vinsert*:
  vfsequence (*vinsert* ⟨*vcard xs*, *a*⟩ *xs*)
  ⟨*proof*⟩

Connections.

lemma (in *vfsequence*) *vfsequence-vfinite*[*simp*]: *vfinite xs*
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vlrestriction*[*intro*, *simp*]:
assumes $k \in_\circ \omega$
shows *vfsequence* $(xs \restriction_\circ^l k)$
⟨*proof*⟩

lemma *vfsequence-vproduct*:
assumes $n \in_\circ \omega$ **and** $xs \in_\circ (\prod_\circ i \in_\circ n.\ A\ i)$
shows *vfsequence xs*
⟨*proof*⟩

lemma *vfsequence-vcpower*:
assumes $n \in_\circ \omega$ **and** $xs \in_\circ A\ \widehat{}_\times\ n$
shows *vfsequence xs*
⟨*proof*⟩

lemma *vfsequence-vcomp-vsv-vfsequence*:
assumes *vsv f* **and** *vfsequence xs* **and** $\mathcal{R}_\circ\ xs \subseteq_\circ \mathcal{D}_\circ\ f$
shows *vfsequence* $(f \circ_\circ xs)$
⟨*proof*⟩

Special properties.

lemma (in *vfsequence*) *vfsequence-vdomain-vlrestriction*[*intro*, *simp*]:
assumes $k \in_\circ$ *vcard xs*
shows $\mathcal{D}_\circ\ (xs \restriction_\circ^l k) = k$
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vlrestriction-vcard*[*simp*]:
$xs \restriction_\circ^l$ (*vcard xs*) = *xs*
⟨*proof*⟩

lemma *vfsequence-vfinite-vcardI*:
assumes *vsv xs* **and** *vfinite xs* **and** $\mathcal{D}_\circ\ xs$ = *vcard xs*
shows *vfsequence xs*
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vrangeE*:
assumes $a \in_\circ \mathcal{R}_\circ\ xs$
obtains *n* **where** $n \in_\circ$ *vcard xs* **and** $xs(\lfloor n \rfloor) = a$
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vrange-vproduct*:
assumes $\bigwedge i.\ i \in_\circ$ *vcard xs* $\Longrightarrow xs(\lfloor i \rfloor) \in_\circ A\ i$
shows $xs \in_\circ (\prod_\circ i \in_\circ$ *vcard xs*. $A\ i)$
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vrange-vcpower*:
assumes $\mathcal{R}_\circ\ xs \subseteq_\circ A$
shows $xs \in_\circ A\ \widehat{}_\times$ (*vcard xs*)
⟨*proof*⟩

Alternative forms of existing results.

lemmas [*intro*, *simp*] = *vfsequence.vfsequence-vcard-in-omega*
and [*intro*, *simp*] = *vfsequence.vfsequence-vfinite*
and [*intro*, *simp*] = *vfsequence.vfsequence-vlrestriction*
and [*intro*, *simp*] = *vfsequence.vfsequence-vdomain-vlrestriction*

and [*intro*, *simp*] = *vfsequence.vfsequence-vlrestriction-vcard*

2.9.3 Appending an element to a finite sequence: *vcons*

Definition and common properties

definition *vcons* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\#_\circ$› 65)
where $xs \#_\circ x$ = *vinsert* ⟨*vcard xs*, *x*⟩ *xs*

Syntax.

abbreviation *vempty-vfsequence* (‹$[]_\circ$›) **where**
vempty-vfsequence ≡ 0::*V*

notation *vempty-vfsequence* (‹$[]_\circ$›)

nonterminal *fsfields*
nonterminal *vlist*

syntax
:: $V \Rightarrow$ *fsfields* (‹-›)
-*fsfields* :: *fsfields* ⇒ $V \Rightarrow$ *fsfields* (‹-,/ -›)
-*vlist* :: *fsfields* ⇒ V (‹$[(-)]_\circ$›)
-*vapp* :: $V \Rightarrow$ *fsfields* ⇒ V (‹- ($|(-)|_\bullet$› [100, 100] 100)

syntax-consts
-*vlist* == *vcons* **and**
-*vapp* == *app*

translations
$[xs, x]_\circ$ == $[xs]_\circ \#_\circ x$
$[x]_\circ$ == $[]_\circ \#_\circ x$

translations
$f(|xs, x|)_\bullet$ == $f(|[xs, x]_\circ|)$
$f(|x|)_\bullet$ == $f(|[x]_\circ|)$

Rules.

lemma *vconsI*[*intro*!]:
assumes $a \in_\circ$ *vinsert* ⟨*vcard xs*, *x*⟩ *xs*
shows $a \in_\circ xs \#_\circ x$
⟨*proof*⟩

lemma *vconsD*[*dest*!]:
assumes $a \in_\circ xs \#_\circ x$
shows $a \in_\circ$ *vinsert* ⟨*vcard xs*, *x*⟩ *xs*
⟨*proof*⟩

lemma *vconsE*[*elim*!]:
assumes $a \in_\circ xs \#_\circ x$
obtains *a* **where** $a \in_\circ$ *vinsert* ⟨*vcard xs*, *x*⟩ *xs*
⟨*proof*⟩

Elementary properties.

lemma *vcons-neq-vempty*[*simp*]: $ys \#_\circ y \neq []_\circ$ ⟨*proof*⟩

Set operations.

lemma *vcons-vsingleton*: $[a]_\circ$ = *set* {⟨$0_\mathbb{N}$, *a*⟩} ⟨*proof*⟩

lemma *vcons-vdoubleton*: $[a,\ b]_{\circ} = set\ \{\langle 0_{\mathbb{N}},\ a\rangle,\ \langle 1_{\mathbb{N}},\ b\rangle\}$
⟨*proof*⟩

lemma *vcons-vsubset*: $xs \subseteq_{\circ} xs \#_{\circ} x$ ⟨*proof*⟩

lemma *vcons-vsubset′*:
assumes $vcons\ xs\ x \subseteq_{\circ} ys$
shows $vcons\ xs\ x \subseteq_{\circ} vcons\ ys\ y$
⟨*proof*⟩

Connections.

lemma (**in** *vfsequence*) *vfsequence-vcons*[*intro*, *simp*]: $vfsequence\ (xs \#_{\circ} x)$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vcons-vdomain*[*simp*]:
$\mathcal{D}_{\circ}\ (xs \#_{\circ} x) = succ\ (vcard\ xs)$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vcons-vrange*[*simp*]:
$\mathcal{R}_{\circ}\ (xs \#_{\circ} x) = vinsert\ x\ (\mathcal{R}_{\circ}\ xs)$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vrange-vconsI*:
assumes $\mathcal{R}_{\circ}\ xs \subseteq_{\circ} X$ **and** $x \in_{\circ} X$
shows $\mathcal{R}_{\circ}\ (xs \#_{\circ} x) \subseteq_{\circ} X$
⟨*proof*⟩

lemmas *vfsequence-vrange-vconsI* = *vfsequence.vfsequence-vrange-vconsI*[*rotated* 1]

Special properties.

lemma *vcons-vrange-mono*:
assumes $xs \subseteq_{\circ} ys$
shows $\mathcal{R}_{\circ}\ (xs \#_{\circ} x) \subseteq_{\circ} \mathcal{R}_{\circ}\ (ys \#_{\circ} x)$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vlrestriction-succ*:
assumes [*simp*]: $k \in_{\circ} vcard\ xs$
shows $xs \restriction_{\circ}^{l} succ\ k = xs \restriction_{\circ}^{l} k \#_{\circ} (xs(\lfloor k \rfloor))$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vremove-vcons-vfsequence*:
assumes $xs = xs' \#_{\circ} x$
shows $vfsequence\ xs'$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vcons-ex*:
assumes $xs \neq []_{\circ}$
obtains $xs'\ x$ **where** $xs = xs' \#_{\circ} x$ **and** $vfsequence\ xs'$
⟨*proof*⟩

Induction and case analysis

lemma *vfsequence-induct*[*consumes* 1, *case-names* 0 *vcons*]:
assumes $vfsequence\ xs$
and $P\ []_{\circ}$
and $\bigwedge xs\ x.\ [\![vfsequence\ xs;\ P\ xs]\!] \Longrightarrow P\ (xs \#_{\circ} x)$
shows $P\ xs$
⟨*proof*⟩

lemma *vfsequence-cases*[*consumes* 1, *case-names* 0 *vcons*]:
assumes *vfsequence xs*
and $xs = []_\circ \Longrightarrow P$
and $\bigwedge xs'\ x.\ [\![xs = xs' \#_\circ x;\ vfsequence\ xs']\!] \Longrightarrow P$
shows *P*
⟨*proof*⟩

Evaluation

lemma (**in** *vfsequence*) *vfsequence-vcard-vcons*[*simp*]:
$vcard\ (xs \#_\circ x) = succ\ (vcard\ xs)$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-at-last*[*intro*, *simp*]:
assumes $i = vcard\ xs$
shows $(xs \#_\circ x)(\!|i|\!) = x$
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-at-not-last*[*intro*, *simp*]:
assumes $i \in_\circ vcard\ xs$
shows $(xs \#_\circ x)(\!|i|\!) = xs(\!|i|\!)$
⟨*proof*⟩

Alternative forms of existing results.

lemmas [*intro*, *simp*] = *vfsequence.vfsequence-vcons*
and [*intro*, *simp*] = *vfsequence.vfsequence-vcard-vcons*
and [*intro*, *simp*] = *vfsequence.vfsequence-at-last*
and [*intro*, *simp*] = *vfsequence.vfsequence-at-not-last*
and [*intro*, *simp*] = *vfsequence.vfsequence-vcons-vdomain*
and [*intro*, *simp*] = *vfsequence.vfsequence-vcons-vrange*

Congruence-like properties

context *vfsequence-pair*
begin

lemma *vcons-eq-vcard-eq*:
assumes $xs_1 \#_\circ x_1 = xs_2 \#_\circ x_2$
shows $vcard\ xs_1 = vcard\ xs_2$
⟨*proof*⟩

lemma *vcons-eqD*[*dest*]:
assumes $xs_1 \#_\circ x_1 = xs_2 \#_\circ x_2$
shows $xs_1 = xs_2$ **and** $x_1 = x_2$
⟨*proof*⟩

lemma *vcons-eqI*:
assumes $xs_1 = xs_2$ **and** $x_1 = x_2$
shows $xs_1 \#_\circ x_1 = xs_2 \#_\circ x_2$
⟨*proof*⟩

lemma *vcons-eq-iff*[*simp*]: $(xs_1 \#_\circ x_1 = xs_2 \#_\circ x_2) \longleftrightarrow (xs_1 = xs_2 \wedge x_1 = x_2)$
⟨*proof*⟩

end

Alternative forms of existing results.

context
fixes xs_1 xs_2
assumes xs_1: *vfsequence* xs_1
and xs_2: *vfsequence* xs_2
begin

lemmas-with[*OF vfsequence-pair.intro*[*OF* xs_1 xs_2]]:
vcons-eqD′ = *vfsequence-pair.vcons-eqD*
and *vcons-eq-iff*[*intro*, *simp*] = *vfsequence-pair.vcons-eq-iff*

end

lemmas *vcons-eqD*[*dest*] = *vcons-eqD′*[*rotated* −1]

2.9.4 Transfer between the type *V list* and finite sequences

Initialization

primrec *vfsequence-of-vlist* :: *V list* ⇒ *V*
where
vfsequence-of-vlist [] = 0
| *vfsequence-of-vlist* (*x* # *xs*) = *vfsequence-of-vlist xs* #∘ *x*

definition *vlist-of-vfsequence* :: *V* ⇒ *V list*
where *vlist-of-vfsequence* = *inv-into UNIV vfsequence-of-vlist*

lemma *vfsequence-vfsequence-of-vlist*: *vfsequence* (*vfsequence-of-vlist xs*)
⟨*proof*⟩

lemma *inj-vfsequence-of-vlist*: *inj vfsequence-of-vlist*
⟨*proof*⟩

lemma *range-vfsequence-of-vlist*:
range vfsequence-of-vlist = {*xs*. *vfsequence xs*}
⟨*proof*⟩

lemma *vlist-of-vfsequence-vfsequence-of-vlist*[*simp*]:
vlist-of-vfsequence (*vfsequence-of-vlist xs*) = *xs*
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-of-vlist-vlist-of-vfsequence*[*simp*]:
vfsequence-of-vlist (*vlist-of-vfsequence xs*) = *xs*
⟨*proof*⟩

lemmas *vfsequence-of-vlist-vlist-of-vfsequence*[*intro*, *simp*] =
vfsequence.vfsequence-of-vlist-vlist-of-vfsequence

lemma *vlist-of-vfsequence-vempty*[*simp*]: *vlist-of-vfsequence* []∘ = []
⟨*proof*⟩

Transfer relation 1.

definition *cr-vfsequence* :: *V* ⇒ *V list* ⇒ *bool*
where *cr-vfsequence a b* ⟷ (*a* = *vfsequence-of-vlist b*)

lemma *cr-vfsequence-right-total*[*transfer-rule*]: *right-total cr-vfsequence*
⟨*proof*⟩

lemma *cr-vfsequence-bi-unqie*[*transfer-rule*]: *bi-unique cr-vfsequence*

⟨*proof*⟩

lemma *cr-vfsequence-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-vfsequence = (λ*xs*. *vfsequence xs*)
⟨*proof*⟩

lemma *cr-vfsequence-vconsD*:
assumes *cr-vfsequence* (*xs* #∘ *x*) (*y* # *ys*)
shows *cr-vfsequence xs ys* **and** *x* = *y*
⟨*proof*⟩

Transfer relation 2.

definition *cr-cr-vfsequence* :: *V* ⇒ *V list list* ⇒ *bool*
where *cr-cr-vfsequence a b* ⟷
(*a* = *vfsequence-of-vlist* (*map vfsequence-of-vlist b*))

lemma *cr-cr-vfsequence-right-total*[*transfer-rule*]:
right-total cr-cr-vfsequence
⟨*proof*⟩

lemma *cr-cr-vfsequence-bi-unqie*[*transfer-rule*]: *bi-unique cr-cr-vfsequence*
⟨*proof*⟩

Transfer relation for scalars.

definition *cr-scalar* :: (*V* ⇒ 'a ⇒ *bool*) ⇒ *V* ⇒ 'a ⇒ *bool*
where *cr-scalar R x y* = (∃ *a*. *x* = [*a*]∘ ∧ *R a y*)

lemma *cr-scalar-bi-unique*[*transfer-rule*]:
assumes *bi-unique R*
shows *bi-unique* (*cr-scalar R*)
⟨*proof*⟩

lemma *cr-scalar-right-total*[*transfer-rule*]:
assumes *right-total R*
shows *right-total* (*cr-scalar R*)
⟨*proof*⟩

lemma *cr-scalar-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp (*cr-scalar R*) = (λ*x*. ∃ *a*. *x* = [*a*]∘ ∧ *Domainp R a*)
⟨*proof*⟩

Transfer rules for previously defined entities

context
includes *lifting-syntax*
begin

lemma *vfsequence-vempty-transfer*[*transfer-rule*]: *cr-vfsequence* []∘ []
⟨*proof*⟩

lemma *vfsequence-vempty-ll-transfer*[*transfer-rule*]:
cr-cr-vfsequence [[]∘]∘ [[]]
⟨*proof*⟩

lemma *vcons-transfer*[*transfer-rule*]:
((=) ===> *cr-vfsequence* ===> *cr-vfsequence*) (λ*x xs*. *xs* #∘ *x*) (λ*x xs*. *x* # *xs*)
⟨*proof*⟩

lemma *vcons-ll-transfer*[*transfer-rule*]:
(*cr-vfsequence* ===> *cr-cr-vfsequence* ===> *cr-cr-vfsequence*)
(λ*x xs*. *xs* #∘ *x*) (λ*x xs*. *x* # *xs*)
⟨*proof*⟩

lemma *vfsequence-vrange-transfer*[*transfer-rule*]:
(*cr-vfsequence* ===> (=)) (λ*xs*. *elts* (ℛ∘ *xs*)) *list.set*
⟨*proof*⟩

lemma *vcard-transfer*[*transfer-rule*]:
(*cr-vfsequence* ===> *cr-omega*) *vcard length*
⟨*proof*⟩

lemma *vcard-ll-transfer*[*transfer-rule*]:
(*cr-cr-vfsequence* ===> *cr-omega*) *vcard length*
⟨*proof*⟩

end

Corollaries.

lemma *vdomain-vfsequence-of-vlist*: 𝒟∘ (*vfsequence-of-vlist xs*) = *length xs*
⟨*proof*⟩

lemma *vrange-vfsequence-of-vlist*:
ℛ∘ (*vfsequence-of-vlist xs*) = *set* (*list.set xs*)
⟨*proof*⟩

lemma *cr-cr-vfsequence-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-cr-vfsequence =
(λ*xss*. *vfsequence xss* ∧ (∀ *xs*∈∘ℛ∘ *xss*. *vfsequence xs*))
⟨*proof*⟩

Appending elements

definition *vappend* :: *V* ⇒ *V* ⇒ *V* (**infixr** ‹@∘› 65)
where *xs* @∘ *ys* =
vfsequence-of-vlist (*vlist-of-vfsequence ys* @ *vlist-of-vfsequence xs*)

Transfer.

lemma *vappend-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vfsequence* ===> *cr-vfsequence* ===> *cr-vfsequence*)
(λ*xs ys*. *vappend ys xs*) *append*
⟨*proof*⟩

lemma *vappend-ll-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-cr-vfsequence* ===> *cr-cr-vfsequence* ===> *cr-cr-vfsequence*)
(λ*xs ys*. *vappend ys xs*) *append*
⟨*proof*⟩

Elementary properties.

lemma (**in** *vfsequence*) *vfsequence-vappend-vempty-vfsequence*[*simp*]:
[]∘ @∘ *xs* = *xs*
⟨*proof*⟩

lemmas *vfsequence-vappend-vempty-vfsequence*[*simp*] =
vfsequence.vfsequence-vappend-vempty-vfsequence

lemma (in *vfsequence*) *vfsequence-vappend-vfsequence-vempty*[*simp*]:
xs @$_\circ$ []$_\circ$ = *xs*
⟨*proof*⟩

lemmas *vfsequence-vappend-vfsequence-vempty*[*simp*] =
vfsequence.vfsequence-vappend-vfsequence-vempty

lemma *vappend-vcons*[*simp*]:
assumes *vfsequence xs* **and** *vfsequence ys*
shows *xs* @$_\circ$ (*ys* #$_\circ$ *y*) = (*xs* @$_\circ$ *ys*) #$_\circ$ *y*
⟨*proof*⟩

Distinct elements

definition *vdistinct* :: *V* ⇒ *bool*
where *vdistinct xs* = *distinct* (*vlist-of-vfsequence xs*)

Transfer.

lemma *vdistinct-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vfsequence* ===> (=)) *vdistinct distinct*
⟨*proof*⟩

lemma *vdistinct-ll-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-cr-vfsequence* ===> (=)) *vdistinct distinct*
⟨*proof*⟩

Elementary properties.

lemma (in *vfsequence*) *vfsequence-vdistinct-if-vcard-vrange-eq-vcard*:
assumes *vcard* (R$_\circ$ *xs*) = *vcard xs*
shows *vdistinct xs*
⟨*proof*⟩

lemma *vdistinct-vempty*[*intro*, *simp*]: *vdistinct* []$_\circ$
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vcons-vdistinct*:
assumes *vdistinct* (*xs* #$_\circ$ *x*)
shows *vdistinct xs*
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vcons-nin-vrange*:
assumes *vdistinct* (*xs* #$_\circ$ *x*)
shows *x* ∉$_\circ$ R$_\circ$ *xs*
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-v11I*[*intro*]:
assumes *vdistinct xs*
shows *v*11 *xs*
⟨*proof*⟩

lemma (in *vfsequence*) *vfsequence-vcons-vdistinctI*:
assumes *vdistinct xs* **and** *x* ∉$_\circ$ R$_\circ$ *xs*
shows *vdistinct* (*xs* #$_\circ$ *x*)
⟨*proof*⟩

lemmas *vfsequence-vcons-vdistinctI*[*intro*] =
vfsequence.vfsequence-vcons-vdistinctI

lemma (**in** *vfsequence*) *vfsequence-nin-vrange-vcons*:
assumes $y \notin_{\circ} \mathcal{R}_{\circ}\ xs$ **and** $y \neq x$
shows $y \notin_{\circ} \mathcal{R}_{\circ}\ (xs\ \#_{\circ}\ x)$
⟨*proof*⟩

lemmas *vfsequence-nin-vrange-vcons*[*intro*] =
vfsequence.vfsequence-nin-vrange-vcons

Concatenation of sequences

definition *vconcat* :: $V \Rightarrow V$
where *vconcat xss* =
vfsequence-of-vlist(
concat (*map vlist-of-vfsequence* (*vlist-of-vfsequence xss*))
)

Transfer.

lemma *vconcat-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-cr-vfsequence* ===> *cr-vfsequence*) *vconcat concat*
⟨*proof*⟩

Elementary properties.

lemma *vconcat-vempty*[*simp*]: *vconcat* $[]_{\circ}$ = $[]_{\circ}$
⟨*proof*⟩

lemma *vconcat-append*[*simp*]:
assumes *vfsequence xss*
and $\forall\, xs \in_{\circ} \mathcal{R}_{\circ}\ xss.$ *vfsequence xs*
and *vfsequence yss*
and $\forall\, xs \in_{\circ} \mathcal{R}_{\circ}\ yss.$ *vfsequence xs*
shows *vconcat* (*xss* $@_{\circ}$ *yss*) = *vconcat xss* $@_{\circ}$ *vconcat yss*
⟨*proof*⟩

lemma *vconcat-vcons*[*simp*]:
assumes *vfsequence xs* **and** *vfsequence xss* **and** $\forall\, xs \in_{\circ} \mathcal{R}_{\circ}\ xss.$ *vfsequence xs*
shows *vconcat* (*xss* $\#_{\circ}$ *xs*) = *vconcat xss* $@_{\circ}$ *xs*
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-vconcat-fsingleton*[*simp*]: *vconcat* $[xs]_{\circ}$ = *xs*
⟨*proof*⟩

lemmas *vfsequence-vconcat-fsingleton*[*simp*] =
vfsequence.vfsequence-vconcat-fsingleton

2.9.5 Finite sequences and the Cartesian product

lemma *vfsequence-vcons-vproductI*[*intro*!]:
assumes $n \in_{\circ} \omega$
and $xs \in_{\circ} (\prod_{\circ} i \in_{\circ} vcard\ xs.\ A\ i)$
and $x \in_{\circ} A\ (vcard\ xs)$
and $n = vcard\ (xs\ \#_{\circ}\ x)$
shows $xs\ \#_{\circ}\ x \in_{\circ} (\prod_{\circ} i \in_{\circ} n.\ A\ i)$
⟨*proof*⟩

lemma *vfsequence-vcons-vproductD*[*dest*]:
assumes $xs \#_\circ x \in_\circ (\prod_\circ i \in_\circ n.\ A\ i)$ **and** $n \in_\circ \omega$
shows $xs \in_\circ (\prod_\circ i \in_\circ vcard\ xs.\ A\ i)$
and $x \in_\circ A\ (vcard\ xs)$
and $n = vcard\ (xs \#_\circ x)$
⟨*proof*⟩

lemma *vfsequence-vcons-vproductE*[*elim*!]:
assumes $xs \#_\circ x \in_\circ (\prod_\circ i \in_\circ n.\ A\ i)$ **and** $n \in_\circ \omega$
obtains $xs \in_\circ (\prod_\circ i \in_\circ vcard\ xs.\ A\ i)$
and $x \in_\circ A\ (vcard\ xs)$
and $n = vcard\ (xs \#_\circ x)$
⟨*proof*⟩

2.9.6 Binary Cartesian product based on finite sequences: *ftimes*

definition *ftimes* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\times_\bullet$› 80)
where $ftimes\ a\ b \equiv (\prod_\circ i \in_\circ 2_{\mathbb{N}}.\ if\ i = 0\ then\ a\ else\ b)$

lemma *small-fpairs*[*simp*]: $small\ \{[a,\ b]_\circ \mid a\ b.\ [a,\ b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *ftimesI*1[*intro*]:
assumes $x = [a,\ b]_\circ$ **and** $a \in_\circ A$ **and** $b \in_\circ B$
shows $x \in_\circ A \times_\bullet B$
⟨*proof*⟩

lemma *ftimesI*2[*intro*!]:
assumes $a \in_\circ A$ **and** $b \in_\circ B$
shows $[a,\ b]_\circ \in_\circ A \times_\bullet B$
⟨*proof*⟩

lemma *fproductE*1[*elim*!]:
assumes $x \in_\circ A \times_\bullet B$
obtains $a\ b$ **where** $x = [a,\ b]_\circ$ **and** $a \in_\circ A$ **and** $b \in_\circ B$
⟨*proof*⟩

lemma *fproductE*2[*elim*!]:
assumes $[a,\ b]_\circ \in_\circ A \times_\bullet B$ **obtains** $a \in_\circ A$ **and** $b \in_\circ B$
⟨*proof*⟩

Set operations.

lemma *vfinite-0-left*[*simp*]: $0 \times_\bullet b = 0$
⟨*proof*⟩

lemma *vfinite-0-right*[*simp*]: $a \times_\bullet 0 = 0$
⟨*proof*⟩

lemma *fproduct-vintersection*: $(a \cap_\circ b) \times_\bullet (c \cap_\circ d) = (a \times_\bullet c) \cap_\circ (b \times_\bullet d)$
⟨*proof*⟩

lemma *fproduct-vdiff*: $a \times_\bullet (b -_\circ c) = (a \times_\bullet b) -_\circ (a \times_\bullet c)$ ⟨*proof*⟩

lemma *vfinite-ftimesI*[*intro*!]:
assumes *vfinite a* **and** *vfinite b*
shows *vfinite* $(a \times_\bullet b)$
⟨*proof*⟩

ftimes and *vcpower*

lemma *vproduct-vpair*: $[a,\ b]_\circ \in_\circ (\prod_\circ i \in_\circ 2_\mathbb{N}.\ f\ i) \longleftrightarrow \langle a,\ b\rangle \in_\circ f\ (0_\mathbb{N}) \times_\circ f\ (1_\mathbb{N})$
⟨*proof*⟩

Connections.

lemma *vcpower-two-ftimes*: $A\ \widehat{\ }_\times\ 2_\mathbb{N} = A \times_\bullet A$
⟨*proof*⟩

lemma *vcpower-two-ftimesI*[*intro*]:
assumes $x \in_\circ A \times_\bullet A$
shows $x \in_\circ A\ \widehat{\ }_\times\ 2_\mathbb{N}$
⟨*proof*⟩

lemma *vcpower-two-ftimesD*[*dest*]:
assumes $x \in_\circ A\ \widehat{\ }_\times\ 2_\mathbb{N}$
shows $x \in_\circ A \times_\bullet A$
⟨*proof*⟩

lemma *vcpower-two-ftimesE*[*elim*]:
assumes $x \in_\circ A\ \widehat{\ }_\times\ 2_\mathbb{N}$ **and** $x \in_\circ A \times_\bullet A \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *vfsequence-vcpower-two-vpair*: $[a,\ b]_\circ \in_\circ A\ \widehat{\ }_\times\ 2_\mathbb{N} \longleftrightarrow \langle a,\ b\rangle \in_\circ A \times_\circ A$
⟨*proof*⟩

lemma *vsv-vfsequence-two*:
assumes *vsv gf* **and** $\mathcal{D}_\circ\ gf = 2_\mathbb{N}$
shows $[vpfst\ gf,\ vpsnd\ gf]_\circ = gf$
⟨*proof*⟩

lemma *vsv-vfsequence-three*:
assumes *vsv hgf* **and** $\mathcal{D}_\circ\ hgf = 3_\mathbb{N}$
shows $[vpfst\ hgf,\ vpsnd\ hgf,\ vpthrd\ hgf]_\circ = hgf$
⟨*proof*⟩

2.9.7 Sequence as an element of a Cartesian power of a set

lemma *vcons-in-vcpowerI*[*intro*!]:
assumes $n \in_\circ \omega$
and $xs \in_\circ A\ \widehat{\ }_\times\ vcard\ xs$
and $x \in_\circ A$
and $n = vcard\ (xs\ \#_\circ\ x)$
shows $xs\ \#_\circ\ x \in_\circ A\ \widehat{\ }_\times\ n$
⟨*proof*⟩

lemma *vcons-in-vcpowerD*[*dest*]:
assumes $xs\ \#_\circ\ x \in_\circ A\ \widehat{\ }_\times\ n$ **and** $n \in_\circ \omega$
shows $xs \in_\circ A\ \widehat{\ }_\times\ vcard\ xs$
and $x \in_\circ A$
and $n = vcard\ (xs\ \#_\circ\ x)$
⟨*proof*⟩

lemma *vcons-in-vcpowerE1*[*elim*!]:
assumes $xs\ \#_\circ\ x \in_\circ A\ \widehat{\ }_\times\ n$ **and** $n \in_\circ \omega$
obtains $xs \in_\circ A\ \widehat{\ }_\times\ vcard\ xs$ **and** $x \in_\circ A$ **and** $n = vcard\ (xs\ \#_\circ\ x)$
⟨*proof*⟩

lemma *vcons-in-vcpowerE2*:
assumes $xs \in_\circ A \mathbin{\widehat{}}_\times n$ **and** $n \in_\circ \omega$ **and** $0 \in_\circ n$
obtains $x\ xs'$ **where** $xs = xs' \#_\circ x$
and $xs' \in_\circ A \mathbin{\widehat{}}_\times vcard\ xs'$
and $x \in_\circ A$
and $n = vcard\ (xs' \#_\circ x)$
⟨*proof*⟩

lemma *vcons-vcpower1E*:
assumes $xs \in_\circ A \mathbin{\widehat{}}_\times 1_{\mathbb{N}}$
obtains x **where** $xs = [x]_\circ$ **and** $x \in_\circ A$
⟨*proof*⟩

2.9.8 The set of all finite sequences on a set

Definition and elementary properties

definition *vfsequences-on* :: $V \Rightarrow V$
where $vfsequences\text{-}on\ X = set\ \{x.\ vfsequence\ x \wedge (\forall\, i \in_\circ \mathcal{D}_\circ\ x.\ x(\lfloor i \rfloor) \in_\circ X)\}$

lemma *vfsequences-on-subset-ω-set*:
$\{x.\ vfsequence\ x \wedge (\forall\, i \in elts\ (\mathcal{D}_\circ\ x).\ x(\lfloor i \rfloor) \in_\circ X)\} \subseteq elts\ (VPow\ (\omega \times_\circ X))$
⟨*proof*⟩

lemma *small-vfsequences-on*[*simp*]:
$small\ \{x.\ vfsequence\ x \wedge (\forall\, i \in_\circ \mathcal{D}_\circ\ x.\ x(\lfloor i \rfloor) \in_\circ X)\}$
⟨*proof*⟩

Rules.

lemma *vfsequences-onI*:
assumes *vfsequence xs* **and** $\bigwedge i.\ i \in_\circ \mathcal{D}_\circ\ xs \Longrightarrow xs(\lfloor i \rfloor) \in_\circ X$
shows $xs \in_\circ vfsequences\text{-}on\ X$
⟨*proof*⟩

lemma *vfsequences-onD*[*dest*]:
assumes $xs \in_\circ vfsequences\text{-}on\ X$
shows *vfsequence xs* **and** $\bigwedge i.\ i \in_\circ \mathcal{D}_\circ\ xs \Longrightarrow xs(\lfloor i \rfloor) \in_\circ X$
⟨*proof*⟩

lemma *vfsequences-onE*[*elim*]:
assumes $xs \in_\circ vfsequences\text{-}on\ X$
obtains *vfsequence xs* **and** $\bigwedge i.\ i \in_\circ \mathcal{D}_\circ\ xs \Longrightarrow xs(\lfloor i \rfloor) \in_\circ X$
⟨*proof*⟩

Further properties

lemma *vfsequences-on-vsubset-mono*:
assumes $A \subseteq_\circ B$
shows $vfsequences\text{-}on\ A \subseteq_\circ vfsequences\text{-}on\ B$
⟨*proof*⟩

2.10 Binary relation as a finite sequence

2.10.1 Background

This section exposes the theory of binary relations that are represented by a two element finite sequence $[a,\ b]_\circ$ (as opposed to a pair $\langle a,\ b\rangle$). Many results were adapted from the theory *CZH-Sets-BRelations*.

As previously, many of the results that are presented in this section can be assumed to have been adapted (with amendments) from the theory *Relation* in the main library.

lemma *fpair-iff*[*simp*]: $([a,\ b]_\circ = [a',\ b']_\circ) = (a = a' \wedge b = b')$ ⟨*proof*⟩

lemmas *fpair-inject*[*elim*!] = *fpair-iff*[*THEN iffD*1, *THEN conjE*]

2.10.2 *fpairs*

definition *fpairs* :: $V \Rightarrow V$ **where**
fpairs r = *set* $\{x.\ x \in_\circ r \wedge (\exists\, a\ b.\ x = [a,\ b]_\circ)\}$

lemma *small-fpairs*[*simp*]: *small* $\{x.\ x \in_\circ r \wedge (\exists\, a\ b.\ x = [a,\ b]_\circ)\}$
⟨*proof*⟩

Rules.

lemma *fpairsI*[*intro*]:
assumes $x \in_\circ r$ **and** $x = [a,\ b]_\circ$
shows $x \in_\circ$ *fpairs r*
⟨*proof*⟩

lemma *fpairsD*[*dest*]:
assumes $x \in_\circ$ *fpairs r*
shows $x \in_\circ r$ **and** $\exists\, a\ b.\ x = [a,\ b]_\circ$
⟨*proof*⟩

lemma *fpairsE*[*elim*]:
assumes $x \in_\circ$ *fpairs r*
obtains *a b* **where** $x = [a,\ b]_\circ$ **and** $[a,\ b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *fpairs-iff*: $x \in_\circ$ *fpairs* $r \longleftrightarrow x \in_\circ r \wedge (\exists\, a\ b.\ x = [a,\ b]_\circ)$ ⟨*proof*⟩

Elementary properties.

lemma *fpairs-iff-elts*: $[a,\ b]_\circ \in_\circ$ *fpairs* $r \longleftrightarrow [a,\ b]_\circ \in_\circ r$ ⟨*proof*⟩

Set operations.

lemma *fpairs-vempty*[*simp*]: *fpairs* $0 = 0$ ⟨*proof*⟩

lemma *fpairs-vsingleton*[*simp*]: *fpairs* (*set* $\{[a,\ b]_\circ\}$) = *set* $\{[a,\ b]_\circ\}$ ⟨*proof*⟩

lemma *fpairs-vinsert*: *fpairs* (*vinsert* $[a,\ b]_\circ$ *A*) = *set* $\{[a,\ b]_\circ\} \cup_\circ$ *fpairs A*
⟨*proof*⟩

lemma *fpairs-mono*:
assumes $r \subseteq_\circ s$
shows *fpairs r* $\subseteq_\circ$ *fpairs s*
⟨*proof*⟩

lemma *fpairs-vunion*: *fpairs* $(A \cup_\circ B)$ = *fpairs A* $\cup_\circ$ *fpairs B* ⟨*proof*⟩

lemma *fpairs-vintersection*: *fpairs* $(A \cap_\circ B)$ = *fpairs* $A \cap_\circ$ *fpairs* B ⟨*proof*⟩

lemma *fpairs-vdiff*: *fpairs* $(A -_\circ B)$ = *fpairs* $A -_\circ$ *fpairs* B ⟨*proof*⟩

Special properties.

lemma *fpairs-ex-vfst*:
assumes $x \in_\circ$ *fpairs* r
shows $\exists\, b.\ [x(\!|0_{\mathbb{N}}|\!),\ b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *fpairs-ex-vsnd*:
assumes $x \in_\circ$ *fpairs* r
shows $\exists\, a.\ [a,\ x(\!|1_{\mathbb{N}}|\!)]_\circ \in_\circ r$
⟨*proof*⟩

lemma *fpair-vcpower2I*[*intro*]:
assumes $a \in_\circ A\ \widehat{}_\times\ 1_{\mathbb{N}}$ **and** $b \in_\circ A\ \widehat{}_\times\ 1_{\mathbb{N}}$
shows *vconcat* $[a,\ b]_\circ \in_\circ A\ \widehat{}_\times\ 2_{\mathbb{N}}$
⟨*proof*⟩

2.10.3 Constructors

Identity relation

definition *fid-on* :: $V \Rightarrow V$
where *fid-on* A = *set* $\{[a,\ a]_\circ \mid a.\ a \in_\circ A\}$

lemma *fid-on-small*[*simp*]: *small* $\{[a,\ a]_\circ \mid a.\ a \in_\circ A\}$
⟨*proof*⟩

Rules.

lemma *fid-on-eqI*:
assumes $a = b$ **and** $a \in_\circ A$
shows $[a,\ b]_\circ \in_\circ$ *fid-on* A
⟨*proof*⟩

lemma *fid-onI*[*intro*!]:
assumes $a \in_\circ A$
shows $[a,\ a]_\circ \in_\circ$ *fid-on* A
⟨*proof*⟩

lemma *fid-onD*[*dest*!]:
assumes $[a,\ a]_\circ \in_\circ$ *fid-on* A
shows $a \in_\circ A$
⟨*proof*⟩

lemma *fid-onE*[*elim*!]:
assumes $x \in_\circ$ *fid-on* A **and** $\exists\, a{\in_\circ}A.\ x = [a,\ a]_\circ \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *fid-on-iff*: $[a,\ b]_\circ \in_\circ$ *fid-on* $A \longleftrightarrow a = b \wedge a \in_\circ A$ ⟨*proof*⟩

Set operations.

lemma *fid-on-vempty*[*simp*]: *fid-on* $0 = 0$ ⟨*proof*⟩

lemma *fid-on-vsingleton*[*simp*]: *fid-on* (*set* $\{a\}$) = *set* $\{[a,\ a]_\circ\}$ ⟨*proof*⟩

lemma *fid-on-vdoubleton*: *fid-on* (*set* {*a*, *b*}) = *set* {[*a*, *a*]∘, [*b*, *b*]∘} ⟨*proof*⟩

lemma *fid-on-mono*:
assumes $A \subseteq_\circ B$
shows *fid-on* $A \subseteq_\circ$ *fid-on* B
⟨*proof*⟩

lemma *fid-on-vinsert*: *vinsert* [*a*, *a*]∘ (*fid-on* *A*) = *fid-on* (*vinsert* *a* *A*)
⟨*proof*⟩

lemma *fid-on-vintersection*: *fid-on* ($A \cap_\circ B$) = *fid-on* $A \cap_\circ$ *fid-on* B ⟨*proof*⟩

lemma *fid-on-vunion*: *fid-on* ($A \cup_\circ B$) = *fid-on* $A \cup_\circ$ *fid-on* B ⟨*proof*⟩

lemma *fid-on-vdiff*: *fid-on* ($A -_\circ B$) = *fid-on* $A -_\circ$ *fid-on* B ⟨*proof*⟩

Special properties.

lemma *fid-on-vsubset-vcpower*: *fid-on* $A \subseteq_\circ A \mathbin{\widehat{\times}} 2_{\mathbf{N}}$ ⟨*proof*⟩

Constant function

definition *fconst-on* :: $V \Rightarrow V \Rightarrow V$
where *fconst-on* *A* *c* = *set* {[*a*, *c*]∘ | *a*. *a* ∈∘ *A*}

lemma *small-fconst-on*[*simp*]: *small* {[*a*, *c*]∘ | *a*. *a* ∈∘ *A*}
⟨*proof*⟩

Rules.

lemma *fconst-onI*[*intro*!]:
assumes $a \in_\circ A$
shows [*a*, *c*]∘ ∈∘ *fconst-on* *A* *c*
⟨*proof*⟩

lemma *fconst-onD*[*dest*!]:
assumes [*a*, *c*]∘ ∈∘ *fconst-on* *A* *c*
shows $a \in_\circ A$
⟨*proof*⟩

lemma *fconst-onE*[*elim*!]:
assumes $x \in_\circ$ *fconst-on* *A* *c*
obtains *a* **where** $a \in_\circ A$ **and** x = [*a*, *c*]∘
⟨*proof*⟩

lemma *fconst-on-iff*: [*a*, *c*]∘ ∈∘ *fconst-on* *A* *c* ⟷ $a \in_\circ A$ ⟨*proof*⟩

Set operations.

lemma *fconst-on-vempty*[*simp*]: *fconst-on* 0 *c* = 0
⟨*proof*⟩

lemma *fconst-on-vsingleton*[*simp*]: *fconst-on* (*set* {*a*}) *c* = *set* {[*a*, *c*]∘}
⟨*proof*⟩

lemma *fconst-on-vdoubleton*: *fconst-on* (*set* {*a*, *b*}) *c* = *set* {[*a*, *c*]∘, [*b*, *c*]∘}
⟨*proof*⟩

lemma *fconst-on-mono*:
assumes $A \subseteq_\circ B$

shows *fconst-on A c* $\subseteq_\circ$ *fconst-on B c*
⟨*proof*⟩

lemma *fconst-on-vinsert*:
(*vinsert* $[a, c]_\circ$ (*fconst-on A c*)) = (*fconst-on* (*vinsert a A*) *c*)
⟨*proof*⟩

lemma *fconst-on-vintersection*:
fconst-on ($A \cap_\circ B$) *c* = *fconst-on A c* $\cap_\circ$ *fconst-on B c*
⟨*proof*⟩

lemma *fconst-on-vunion*: *fconst-on* ($A \cup_\circ B$) *c* = *fconst-on A c* $\cup_\circ$ *fconst-on B c*
⟨*proof*⟩

lemma *fconst-on-vdiff*: *fconst-on* ($A -_\circ B$) *c* = *fconst-on A c* $-_\circ$ *fconst-on B c*
⟨*proof*⟩

Special properties.

lemma *fconst-on-eq-ftimes*: *fconst-on A c* = $A \times_\bullet$ *set* {*c*} ⟨*proof*⟩

Composition

definition *fcomp* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\circ_\bullet$› 75)
where $r \circ_\bullet s$ = *set* $\{[a, c]_\circ \mid a\ c.\ \exists b.\ [a, b]_\circ \in_\circ s \wedge [b, c]_\circ \in_\circ r\}$
notation *fcomp* (**infixr** ‹$\circ_\bullet$› 75)

lemma *fcomp-small*[*simp*]: *small* $\{[a, c]_\circ \mid a\ c.\ \exists b.\ [a, b]_\circ \in_\circ s \wedge [b, c]_\circ \in_\circ r\}$
(**is** ‹*small ?s*›)
⟨*proof*⟩

Rules.

lemma *fcompI*[*intro*]:
assumes $[b, c]_\circ \in_\circ r$ **and** $[a, b]_\circ \in_\circ s$
shows $[a, c]_\circ \in_\circ r \circ_\bullet s$
⟨*proof*⟩

lemma *fcompD*[*dest*]:
assumes $[a, c]_\circ \in_\circ r \circ_\bullet s$
shows $\exists b.\ [b, c]_\circ \in_\circ r \wedge [a, b]_\circ \in_\circ s$
⟨*proof*⟩

lemma *fcompE*[*elim*]:
assumes $ac \in_\circ r \circ_\bullet s$
obtains *a b c* **where** $ac = [a, c]_\circ$ **and** $[a, b]_\circ \in_\circ s$ **and** $[b, c]_\circ \in_\circ r$
⟨*proof*⟩

Elementary properties.

lemma *fcomp-assoc*: $(r \circ_\bullet s) \circ_\bullet t = r \circ_\bullet (s \circ_\bullet t)$ ⟨*proof*⟩

Set operations.

lemma *fcomp-vempty-left*[*simp*]: $0 \circ_\bullet r = 0$ ⟨*proof*⟩

lemma *fcomp-vempty-right*[*simp*]: $r \circ_\bullet 0 = 0$ ⟨*proof*⟩

lemma *fcomp-mono*:
assumes $r' \subseteq_\circ r$ **and** $s' \subseteq_\circ s$
shows $r' \circ_\bullet s' \subseteq_\circ r \circ_\bullet s$

⟨*proof*⟩

lemma *fcomp-vinsert-left*[*simp*]:
vinsert ([*a*, *b*]$_\circ$) *s* $\circ_\bullet$ *r* = (*set* {[*a*, *b*]$_\circ$} $\circ_\bullet$ *r*) $\cup_\circ$ (*s* $\circ_\bullet$ *r*)
⟨*proof*⟩

lemma *fcomp-vinsert-right*[*simp*]:
r $\circ_\bullet$ *vinsert* [*a*, *b*]$_\circ$ *s* = (*r* $\circ_\bullet$ *set* {[*a*, *b*]$_\circ$}) $\cup_\circ$ (*r* $\circ_\bullet$ *s*)
⟨*proof*⟩

lemma *fcomp-vunion-left*[*simp*]: (*s* $\cup_\circ$ *t*) $\circ_\bullet$ *r* = (*s* $\circ_\bullet$ *r*) $\cup_\circ$ (*t* $\circ_\bullet$ *r*) ⟨*proof*⟩

lemma *fcomp-vunion-right*[*simp*]: *r* $\circ_\bullet$ (*s* $\cup_\circ$ *t*) = (*r* $\circ_\bullet$ *s*) $\cup_\circ$ (*r* $\circ_\bullet$ *t*) ⟨*proof*⟩

Connections.

lemma *fcomp-fid-on-idem*[*simp*]: *fid-on A* $\circ_\bullet$ *fid-on A* = *fid-on A* ⟨*proof*⟩

lemma *fcomp-fid-on*[*simp*]: *fid-on A* $\circ_\bullet$ *fid-on B* = *fid-on* (*A* $\cap_\circ$ *B*) ⟨*proof*⟩

lemma *fcomp-fconst-on-fid-on*[*simp*]: *fconst-on A c* $\circ_\bullet$ *fid-on A* = *fconst-on A c*
⟨*proof*⟩

Special properties.

lemma *fcomp-vsubset-vtimes*:
assumes *r* $\subseteq_\circ$ *B* $\times_\bullet$ *C* **and** *s* $\subseteq_\circ$ *A* $\times_\bullet$ *B*
shows *r* $\circ_\bullet$ *s* $\subseteq_\circ$ *A* $\times_\bullet$ *C*
⟨*proof*⟩

lemma *fcomp-obtain-middle*[*elim*]:
assumes [*a*, *c*]$_\circ$ $\in_\circ$ *f* $\circ_\bullet$ *g*
obtains *b* **where** [*a*, *b*]$_\circ$ $\in_\circ$ *g* **and** [*b*, *c*]$_\circ$ $\in_\circ$ *f*
⟨*proof*⟩

Converse relation

definition *fconverse* :: *V* ⇒ *V* (‹(-$^{-1}_\bullet$)› [1000] 999)
where $r^{-1}_\bullet$ = *set* {[*b*, *a*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ *r*}

lemma *fconverse-small*[*simp*]: *small* {[*b*, *a*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ *r*}
⟨*proof*⟩

Rules.

lemma *fconverseI*[*sym*, *intro*!]:
assumes [*a*, *b*]$_\circ$ $\in_\circ$ *r*
shows [*b*, *a*]$_\circ$ $\in_\circ$ $r^{-1}_\bullet$
⟨*proof*⟩

lemma *fconverseD*[*sym*, *dest*]:
assumes [*a*, *b*]$_\circ$ $\in_\circ$ $r^{-1}_\bullet$
shows [*b*, *a*]$_\circ$ $\in_\circ$ *r*
⟨*proof*⟩

lemma *fconverseE*[*elim*!]:
assumes *x* $\in_\circ$ $r^{-1}_\bullet$
obtains *a b* **where** *x* = [*b*, *a*]$_\circ$ **and** [*a*, *b*]$_\circ$ $\in_\circ$ *r*
⟨*proof*⟩

lemma *fconverse-iff*: [*b*, *a*]$_\circ$ $\in_\circ$ $r^{-1}_\bullet$ ⟷ [*a*, *b*]$_\circ$ $\in_\circ$ *r* ⟨*proof*⟩

Set operations.

lemma *fconverse-vempty*[*simp*]: $0^{-1}_\bullet = 0$ ⟨*proof*⟩

lemma *fconverse-vsingleton*: $(set\ \{[a,\ b]_\circ\})^{-1}_\bullet = set\ \{[b,\ a]_\circ\}$ ⟨*proof*⟩

lemma *fconverse-vdoubleton*: $(set\ \{[a,\ b]_\circ,\ [c,\ d]_\circ\})^{-1}_\bullet = set\ \{[b,\ a]_\circ,\ [d,\ c]_\circ\}$
⟨*proof*⟩

lemma *fconverse-vinsert*: $(vinsert\ [a,\ b]_\circ\ r)^{-1}_\bullet = vinsert\ [b,\ a]_\circ\ (r^{-1}_\bullet)$ ⟨*proof*⟩

lemma *fconverse-vintersection*: $(r \cap_\circ s)^{-1}_\bullet = r^{-1}_\bullet \cap_\circ s^{-1}_\bullet$ ⟨*proof*⟩

lemma *fconverse-vunion*: $(r \cup_\circ s)^{-1}_\bullet = r^{-1}_\bullet \cup_\circ s^{-1}_\bullet$ ⟨*proof*⟩

Connections.

lemma *fconverse-fid-on*[*simp*]: $(fid\text{-}on\ A)^{-1}_\bullet = fid\text{-}on\ A$ ⟨*proof*⟩

lemma *fconverse-fconst-on*[*simp*]: $(fconst\text{-}on\ A\ c)^{-1}_\bullet = set\ \{c\} \times_\bullet A$ ⟨*proof*⟩

lemma *fconverse-fcomp*: $(r \circ_\bullet s)^{-1}_\bullet = s^{-1}_\bullet \circ_\bullet r^{-1}_\bullet$ ⟨*proof*⟩

lemma *fconverse-ftimes*: $(A \times_\bullet B)^{-1}_\bullet = (B \times_\bullet A)$ ⟨*proof*⟩

Special properties.

lemma *fconverse-pred*:
assumes *small* $\{[a,\ b]_\circ \mid a\ b.\ P\ a\ b\}$
shows $(set\ \{[a,\ b]_\circ \mid a\ b.\ P\ a\ b\})^{-1}_\bullet = set\ \{[b,\ a]_\circ \mid a\ b.\ P\ a\ b\}$
⟨*proof*⟩

Left restriction

definition *flrestriction* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\restriction^l_\bullet$› 80)
where $r \restriction^l_\bullet A = set\ \{[a,\ b]_\circ \mid a\ b.\ a \in_\circ A \wedge [a,\ b]_\circ \in_\circ r\}$

lemma *flrestriction-small*[*simp*]: *small* $\{[a,\ b]_\circ \mid a\ b.\ a \in_\circ A \wedge [a,\ b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *flrestrictionI*[*intro*!]:
assumes $a \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
shows $[a,\ b]_\circ \in_\circ r \restriction^l_\bullet A$
⟨*proof*⟩

lemma *flrestrictionD*[*dest*]:
assumes $[a,\ b]_\circ \in_\circ r \restriction^l_\bullet A$
shows $a \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *flrestrictionE*[*elim*!]:
assumes $x \in_\circ r \restriction^l_\bullet A$
obtains $a\ b$ **where** $x = [a,\ b]_\circ$ **and** $a \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *flrestriction-on-vempty*[*simp*]: $r \restriction^l_\bullet 0 = 0$ ⟨*proof*⟩

lemma *flrestriction-vempty*[*simp*]: $0 \restriction^l_\bullet A = 0$ ⟨*proof*⟩

lemma *ftrestriction-vsingleton-in*[*simp*]:
assumes $a \in_\circ A$
shows *set* $\{[a, b]_\circ\} \upharpoonright^l_\bullet A$ = *set* $\{[a, b]_\circ\}$
⟨*proof*⟩

lemma *ftrestriction-vsingleton-nin*[*simp*]:
assumes $a \notin_\circ A$
shows *set* $\{[a, b]_\circ\} \upharpoonright^l_\bullet A = 0$
⟨*proof*⟩

lemma *ftrestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \upharpoonright^l_\bullet A \subseteq_\circ r \upharpoonright^l_\bullet B$
⟨*proof*⟩

lemma *ftrestriction-vinsert-nin*[*simp*]:
assumes $a \notin_\circ A$
shows (*vinsert* $[a, b]_\circ$ r) $\upharpoonright^l_\bullet A = r \upharpoonright^l_\bullet A$
⟨*proof*⟩

lemma *ftrestriction-vinsert-in*:
assumes $a \in_\circ A$
shows (*vinsert* $[a, b]_\circ$ r) $\upharpoonright^l_\bullet A$ = *vinsert* $[a, b]_\circ$ $(r \upharpoonright^l_\bullet A)$
⟨*proof*⟩

lemma *ftrestriction-vintersection*: $(r \cap_\circ s) \upharpoonright^l_\bullet A = r \upharpoonright^l_\bullet A \cap_\circ s \upharpoonright^l_\bullet A$ ⟨*proof*⟩

lemma *ftrestriction-vunion*: $(r \cup_\circ s) \upharpoonright^l_\bullet A = r \upharpoonright^l_\bullet A \cup_\circ s \upharpoonright^l_\bullet A$ ⟨*proof*⟩

lemma *ftrestriction-vdiff*: $(r -_\circ s) \upharpoonright^l_\bullet A = r \upharpoonright^l_\bullet A -_\circ s \upharpoonright^l_\bullet A$ ⟨*proof*⟩

Connections.

lemma *ftrestriction-fid-on*[*simp*]: (*fid-on* A) $\upharpoonright^l_\bullet B$ = *fid-on* $(A \cap_\circ B)$ ⟨*proof*⟩

lemma *ftrestriction-fconst-on*: (*fconst-on* A c) $\upharpoonright^l_\bullet B$ = (*fconst-on* B c) $\upharpoonright^l_\bullet A$
⟨*proof*⟩

lemma *ftrestriction-fconst-on-commute*:
assumes $x \in_\circ$ *fconst-on* A c $\upharpoonright^l_\bullet B$
shows $x \in_\circ$ *fconst-on* B c $\upharpoonright^l_\bullet A$
⟨*proof*⟩

lemma *ftrestriction-fcomp*[*simp*]: $(r \circ_\bullet s) \upharpoonright^l_\bullet A = r \circ_\bullet (s \upharpoonright^l_\bullet A)$ ⟨*proof*⟩

Previous connections.

lemma *fcomp-rel-fid-on*[*simp*]: $r \circ_\bullet$ *fid-on* $A = r \upharpoonright^l_\bullet A$ ⟨*proof*⟩

lemma *fcomp-fconst-on*:
$r \circ_\bullet$ (*fconst-on* A c) = $(r \upharpoonright^l_\bullet$ *set* $\{c\}) \circ_\bullet$ (*fconst-on* A c)
⟨*proof*⟩

Special properties.

lemma *ftrestriction-vsubset-fpairs*: $r \upharpoonright^l_\bullet A \subseteq_\circ$ *fpairs* r
⟨*proof*⟩

lemma *ftrestriction-vsubset-frel*: $r \upharpoonright^l_\bullet A \subseteq_\circ r$ ⟨*proof*⟩

Right restriction

definition *frrestriction* :: $V \Rightarrow V \Rightarrow V$ (**infixr** $\langle\restriction^r_\bullet\rangle$ 80)
where $r \restriction^r_\bullet A = set\ \{[a,\ b]_\circ \mid a\ b.\ b \in_\circ A \wedge [a,\ b]_\circ \in_\circ r\}$

lemma *frrestriction-small*[*simp*]: *small* $\{[a,\ b]_\circ \mid a\ b.\ b \in_\circ A \wedge [a,\ b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *frrestrictionI*[*intro*!]:
assumes $b \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
shows $[a,\ b]_\circ \in_\circ r \restriction^r_\bullet A$
⟨*proof*⟩

lemma *frrestrictionD*[*dest*]:
assumes $[a,\ b]_\circ \in_\circ r \restriction^r_\bullet A$
shows $b \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *frrestrictionE*[*elim*!]:
assumes $x \in_\circ r \restriction^r_\bullet A$
obtains $a\ b$ **where** $x = [a,\ b]_\circ$ **and** $b \in_\circ A$ **and** $[a,\ b]_\circ \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *frrestriction-on-vempty*[*simp*]: $r \restriction^r_\bullet 0 = 0$ ⟨*proof*⟩

lemma *frrestriction-vempty*[*simp*]: $0 \restriction^r_\bullet A = 0$ ⟨*proof*⟩

lemma *frrestriction-vsingleton-in*[*simp*]:
assumes $b \in_\circ A$
shows $set\ \{[a,\ b]_\circ\} \restriction^r_\bullet A = set\ \{[a,\ b]_\circ\}$
⟨*proof*⟩

lemma *frrestriction-vsingleton-nin*[*simp*]:
assumes $b \notin_\circ A$
shows $set\ \{[a,\ b]_\circ\} \restriction^r_\bullet A = 0$
⟨*proof*⟩

lemma *frrestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \restriction^r_\bullet A \subseteq_\circ r \restriction^r_\bullet B$
⟨*proof*⟩

lemma *frrestriction-vinsert-nin*[*simp*]:
assumes $b \notin_\circ A$
shows $(vinsert\ [a,\ b]_\circ\ r) \restriction^r_\bullet A = r \restriction^r_\bullet A$
⟨*proof*⟩

lemma *frrestriction-vinsert-in*:
assumes $b \in_\circ A$
shows $(vinsert\ [a,\ b]_\circ\ r) \restriction^r_\bullet A = vinsert\ [a,\ b]_\circ\ (r \restriction^r_\bullet A)$
⟨*proof*⟩

lemma *frrestriction-vintersection*: $(r \cap_\circ s) \restriction^r_\bullet A = r \restriction^r_\bullet A \cap_\circ s \restriction^r_\bullet A$ ⟨*proof*⟩

lemma *frrestriction-vunion*: $(r \cup_\circ s) \restriction^r_\bullet A = r \restriction^r_\bullet A \cup_\circ s \restriction^r_\bullet A$ ⟨*proof*⟩

lemma *frrestriction-vdiff*: $(r -_\circ s) \restriction^r_\bullet A = r \restriction^r_\bullet A -_\circ s \restriction^r_\bullet A$ ⟨*proof*⟩

Connections.

lemma *frrestriction-fid-on*[*simp*]: $(\mathit{fid\text{-}on}\ A) \restriction^r_\bullet B = \mathit{fid\text{-}on}\ (A \cap_\circ B)$ ⟨*proof*⟩

lemma *frrestriction-fconst-on*:
assumes $c \in_\circ B$
shows $(\mathit{fconst\text{-}on}\ A\ c) \restriction^r_\bullet B = \mathit{fconst\text{-}on}\ A\ c$
⟨*proof*⟩

lemma *frrestriction-fcomp*[*simp*]: $(r \circ_\bullet s) \restriction^r_\bullet A = (r \restriction^r_\bullet A) \circ_\bullet s$ ⟨*proof*⟩

Previous connections.

lemma *fcomp-fid-on-rel*[*simp*]: $\mathit{fid\text{-}on}\ A \circ_\bullet r = r \restriction^r_\bullet A$ ⟨*proof*⟩

lemma *fcomp-fconst-on-rel*: $(\mathit{fconst\text{-}on}\ A\ c) \circ_\bullet r = (\mathit{fconst\text{-}on}\ A\ c) \circ_\bullet (r \restriction^r_\bullet A)$
⟨*proof*⟩

lemma *flrestriction-fconverse*: $r^{-1}_\bullet \restriction^l_\bullet A = (r \restriction^r_\bullet A)^{-1}_\bullet$ ⟨*proof*⟩

lemma *frrestriction-fconverse*: $r^{-1}_\bullet \restriction^r_\bullet A = (r \restriction^l_\bullet A)^{-1}_\bullet$ ⟨*proof*⟩

Special properties.

lemma *frrestriction-vsubset-rel*: $r \restriction^r_\bullet A \subseteq_\circ r$ ⟨*proof*⟩

lemma *frrestriction-vsubset-vpairs*: $r \restriction^r_\bullet A \subseteq_\circ \mathit{fpairs}\ r$ ⟨*proof*⟩

Restriction

definition *frestriction* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\restriction_\bullet$› 80)
where $r \restriction_\bullet A = \mathit{set}\ \{[a, b]_\circ \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge [a, b]_\circ \in_\circ r\}$

lemma *frestriction-small*[*simp*]:
$\mathit{small}\ \{[a, b]_\circ \mid a\ b.\ a \in_\circ A \wedge b \in_\circ A \wedge [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *frestrictionI*[*intro*!]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$ **and** $[a, b]_\circ \in_\circ r$
shows $[a, b]_\circ \in_\circ r \restriction_\bullet A$
⟨*proof*⟩

lemma *frestrictionD*[*dest*]:
assumes $[a, b]_\circ \in_\circ r \restriction_\bullet A$
shows $a \in_\circ A$ **and** $b \in_\circ A$ **and** $[a, b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *frestrictionE*[*elim*!]:
assumes $x \in_\circ r \restriction_\bullet A$
obtains $a\ b$ **where** $x = [a, b]_\circ$ **and** $a \in_\circ A$ **and** $b \in_\circ A$ **and** $[a, b]_\circ \in_\circ r$
⟨*proof*⟩

Set operations.

lemma *frestriction-on-vempty*[*simp*]: $r \restriction_\bullet 0 = 0$ ⟨*proof*⟩

lemma *frestriction-vempty*[*simp*]: $0 \restriction_\bullet A = 0$ ⟨*proof*⟩

lemma *frestriction-vsingleton-in*[*simp*]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows *set* $\{[a, b]_\circ\} \upharpoonright_\bullet A$ = *set* $\{[a, b]_\circ\}$
⟨*proof*⟩

lemma *frestriction-vsingleton-nin-left*[*simp*]:
assumes $a \notin_\circ A$
shows *set* $\{[a, b]_\circ\} \upharpoonright_\bullet A = 0$
⟨*proof*⟩

lemma *frestriction-vsingleton-nin-right*[*simp*]:
assumes $b \notin_\circ A$
shows *set* $\{[a, b]_\circ\} \upharpoonright_\bullet A = 0$
⟨*proof*⟩

lemma *frestriction-mono*:
assumes $A \subseteq_\circ B$
shows $r \upharpoonright_\bullet A \subseteq_\circ r \upharpoonright_\bullet B$
⟨*proof*⟩

lemma *frestriction-vinsert-nin*[*simp*]:
assumes $a \notin_\circ A$ **and** $b \notin_\circ A$
shows (*vinsert* $[a, b]_\circ$ r) $\upharpoonright_\bullet A = r \upharpoonright_\bullet A$
⟨*proof*⟩

lemma *frestriction-vinsert-in*:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows (*vinsert* $[a, b]_\circ$ r) $\upharpoonright_\bullet A$ = *vinsert* $[a, b]_\circ$ $(r \upharpoonright_\bullet A)$
⟨*proof*⟩

lemma *frestriction-vintersection*: $(r \cap_\circ s) \upharpoonright_\bullet A = r \upharpoonright_\bullet A \cap_\circ s \upharpoonright_\bullet A$ ⟨*proof*⟩

lemma *frestriction-vunion*: $(r \cup_\circ s) \upharpoonright_\bullet A = r \upharpoonright_\bullet A \cup_\circ s \upharpoonright_\bullet A$ ⟨*proof*⟩

lemma *frestriction-vdiff*: $(r -_\circ s) \upharpoonright_\bullet A = r \upharpoonright_\bullet A -_\circ s \upharpoonright_\bullet A$ ⟨*proof*⟩

Connections.

lemma *fid-on-frestriction*[*simp*]: (*fid-on* A) $\upharpoonright_\bullet B$ = *fid-on* $(A \cap_\circ B)$ ⟨*proof*⟩

lemma *frestriction-fconst-on-ex*:
assumes $c \in_\circ B$
shows (*fconst-on* A c) $\upharpoonright_\bullet B$ = *fconst-on* $(A \cap_\circ B)$ c
⟨*proof*⟩

lemma *frestriction-fconst-on-nex*:
assumes $c \notin_\circ B$
shows (*fconst-on* A c) $\upharpoonright_\bullet B = 0$
⟨*proof*⟩

lemma *frestriction-fcomp*[*simp*]: $(r \circ_\bullet s) \upharpoonright_\bullet A = (r \upharpoonright^r_\bullet A) \circ_\bullet (s \upharpoonright^l_\bullet A)$ ⟨*proof*⟩

lemma *frestriction-fconverse*: $r^{-1}_\bullet \upharpoonright_\bullet A = (r \upharpoonright_\bullet A)^{-1}_\bullet$ ⟨*proof*⟩

Previous connections.

lemma *frrestriction-flrestriction*[*simp*]: $(r \upharpoonright^r_\bullet A) \upharpoonright^l_\bullet A = r \upharpoonright_\bullet A$ ⟨*proof*⟩

lemma *flrestriction-frrestriction*[*simp*]: $(r \upharpoonright^l_\bullet A) \upharpoonright^r_\bullet A = r \upharpoonright_\bullet A$ ⟨*proof*⟩

lemma *frestriction-flrestriction*[*simp*]: $(r \upharpoonright_\bullet A) \upharpoonright^l_\bullet A = r \upharpoonright_\bullet A$ ⟨*proof*⟩

lemma *frestriction-frrestriction*[*simp*]: $(r \upharpoonright_\bullet A) \upharpoonright^r_\bullet A = r \upharpoonright_\bullet A$ ⟨*proof*⟩

Special properties.

lemma *frestriction-vsubset-fpairs*: $r \upharpoonright_\bullet A \subseteq_\circ$ *fpairs* r ⟨*proof*⟩

lemma *frestriction-vsubset-ftimes*: $r \upharpoonright_\bullet A \subseteq_\circ A \mathbin{\widehat{}}_\times 2_\mathbb{N}$ ⟨*proof*⟩

lemma *frestriction-vsubset-rel*: $r \upharpoonright_\bullet A \subseteq_\circ r$ ⟨*proof*⟩

2.10.4 Properties

Domain

definition *fdomain* :: $V \Rightarrow V$ $(\langle \mathcal{D}_\bullet \rangle)$
  where $\mathcal{D}_\bullet\ r = set\ \{a.\ \exists b.\ [a, b]_\circ \in_\circ r\}$
notation *fdomain* $(\langle \mathcal{D}_\bullet \rangle)$

lemma *fdomain-small*[*simp*]: *small* $\{a.\ \exists b.\ [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *fdomainI*[*intro*]:
  assumes $[a, b]_\circ \in_\circ r$
  shows $a \in_\circ \mathcal{D}_\bullet\ r$
  ⟨*proof*⟩

lemma *fdomainD*[*dest*]:
  assumes $a \in_\circ \mathcal{D}_\bullet\ r$
  shows $\exists b.\ [a, b]_\circ \in_\circ r$
  ⟨*proof*⟩

lemma *fdomainE*[*elim*]:
  assumes $a \in_\circ \mathcal{D}_\bullet\ r$
  obtains b **where** $[a, b]_\circ \in_\circ r$
  ⟨*proof*⟩

lemma *fdomain-iff*: $a \in_\circ \mathcal{D}_\bullet\ r \longleftrightarrow (\exists y.\ [a, y]_\circ \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *fdomain-vempty*[*simp*]: $\mathcal{D}_\bullet\ 0 = 0$ ⟨*proof*⟩

lemma *fdomain-vsingleton*[*simp*]: $\mathcal{D}_\bullet\ (set\ \{[a, b]_\circ\}) = set\ \{a\}$ ⟨*proof*⟩

lemma *fdomain-vdoubleton*[*simp*]: $\mathcal{D}_\bullet\ (set\ \{[a, b]_\circ, [c, d]_\circ\}) = set\ \{a, c\}$
  ⟨*proof*⟩

lemma *fdomain-mono*:
  assumes $r \subseteq_\circ s$
  shows $\mathcal{D}_\bullet\ r \subseteq_\circ \mathcal{D}_\bullet\ s$
  ⟨*proof*⟩

lemma *fdomain-vinsert*[*simp*]: $\mathcal{D}_\bullet\ (vinsert\ [a, b]_\circ\ r) = vinsert\ a\ (\mathcal{D}_\bullet\ r)$
  ⟨*proof*⟩

lemma *fdomain-vunion*: $\mathcal{D}_\bullet\ (A \cup_\circ B) = \mathcal{D}_\bullet\ A \cup_\circ \mathcal{D}_\bullet\ B$ ⟨*proof*⟩

lemma *fdomain-vintersection-vsubset*: $\mathcal{D}_\bullet\ (A \cap_\circ B) \subseteq_\circ \mathcal{D}_\bullet\ A \cap_\circ \mathcal{D}_\bullet\ B$ ⟨*proof*⟩

lemma *fdomain-vdiff-vsubset*: $\mathcal{D}_\bullet\ A -_\circ \mathcal{D}_\bullet\ B \subseteq_\circ \mathcal{D}_\bullet\ (A -_\circ B)$ ⟨*proof*⟩

Connections.

lemma *fdomain-fid-on*[*simp*]: $\mathcal{D}_\bullet\ (\mathit{fid\text{-}on}\ A) = A$ ⟨*proof*⟩

lemma *fdomain-fconst-on*[*simp*]: $\mathcal{D}_\bullet\ (\mathit{fconst\text{-}on}\ A\ c) = A$ ⟨*proof*⟩

lemma *fdomain-flrestriction*: $\mathcal{D}_\bullet\ (r \upharpoonright^l_\bullet A) = \mathcal{D}_\bullet\ r \cap_\circ A$ ⟨*proof*⟩

Special properties.

lemma *fdomain-vsubset-ftimes*:
assumes *fpairs* $r \subseteq_\circ A \times_\bullet B$
shows $\mathcal{D}_\bullet\ r \subseteq_\circ A$
⟨*proof*⟩

lemma *fdomain-vsubset-VUnion2*: $\mathcal{D}_\bullet\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ(\bigcup_\circ r))$
⟨*proof*⟩

Range

definition *frange* :: $V \Rightarrow V$ $(\langle\mathcal{R}_\bullet\rangle)$
where $\mathit{frange}\ r = \mathit{set}\ \{b.\ \exists\, a.\ [a, b]_\circ \in_\circ r\}$
notation *frange* $(\langle\mathcal{R}_\bullet\rangle)$

lemma *frange-small*[*simp*]: *small* $\{b.\ \exists\, a.\ [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *frangeI*[*intro*]:
assumes $[a, b]_\circ \in_\circ r$
shows $b \in_\circ \mathcal{R}_\bullet\ r$
⟨*proof*⟩

lemma *frangeD*[*dest*]:
assumes $b \in_\circ \mathcal{R}_\bullet\ r$
shows $\exists\, a.\ [a, b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *frangeE*[*elim*!]:
assumes $b \in_\circ \mathcal{R}_\bullet\ r$
obtains a **where** $[a, b]_\circ \in_\circ r$
⟨*proof*⟩

lemma *frange-iff*: $b \in_\circ \mathcal{R}_\bullet\ r \longleftrightarrow (\exists\, a.\ [a, b]_\circ \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *frange-vempty*[*simp*]: $\mathcal{R}_\bullet\ 0 = 0$ ⟨*proof*⟩

lemma *frange-vsingleton*[*simp*]: $\mathcal{R}_\bullet\ (\mathit{set}\ \{[a, b]_\circ\}) = \mathit{set}\ \{b\}$ ⟨*proof*⟩

lemma *frange-vdoubleton*[*simp*]: $\mathcal{R}_\bullet\ (\mathit{set}\ \{[a, b]_\circ, [c, d]_\circ\}) = \mathit{set}\ \{b, d\}$
⟨*proof*⟩

lemma *frange-mono*:
assumes $r \subseteq_\circ s$

shows $\mathcal{R}_\bullet\ r \subseteq_\circ \mathcal{R}_\bullet\ s$
⟨*proof*⟩

lemma *frange-vinsert*[*simp*]: $\mathcal{R}_\bullet\ (\mathit{vinsert}\ [a,\ b]_\circ\ r) = \mathit{vinsert}\ b\ (\mathcal{R}_\bullet\ r)$ ⟨*proof*⟩

lemma *frange-vunion*: $\mathcal{R}_\bullet\ (r \cup_\circ s) = \mathcal{R}_\bullet\ r \cup_\circ \mathcal{R}_\bullet\ s$ ⟨*proof*⟩

lemma *frange-vintersection-vsubset*: $\mathcal{R}_\bullet\ (r \cap_\circ s) \subseteq_\circ \mathcal{R}_\bullet\ r \cap_\circ \mathcal{R}_\bullet\ s$ ⟨*proof*⟩

lemma *frange-vdiff-vsubset*: $\mathcal{R}_\bullet\ r -_\circ \mathcal{R}_\bullet\ s \subseteq_\circ \mathcal{R}_\bullet\ (r -_\circ s)$ ⟨*proof*⟩

Connections.

lemma *frange-fid-on*[*simp*]: $\mathcal{R}_\bullet\ (\mathit{fid\text{-}on}\ A) = A$ ⟨*proof*⟩

lemma *frange-fconst-on-vempty*[*simp*]: $\mathcal{R}_\bullet\ (\mathit{fconst\text{-}on}\ 0\ c) = 0$ ⟨*proof*⟩

lemma *frange-fconst-on-ne*[*simp*]:
assumes $A \neq 0$
shows $\mathcal{R}_\bullet\ (\mathit{fconst\text{-}on}\ A\ c) = \mathit{set}\ \{c\}$
⟨*proof*⟩

lemma *frange-vrrestriction*: $\mathcal{R}_\bullet\ (r\ \restriction^r_\bullet\ A) = \mathcal{R}_\bullet\ r \cap_\circ A$ ⟨*proof*⟩

Previous connections

lemma *fdomain-fconverse*[*simp*]: $\mathcal{D}_\bullet\ ({r^{-1}}_\bullet) = \mathcal{R}_\bullet\ r$ ⟨*proof*⟩

lemma *frange-fconverse*[*simp*]: $\mathcal{R}_\bullet\ ({r^{-1}}_\bullet) = \mathcal{D}_\bullet\ r$ ⟨*proof*⟩

Special properties.

lemma *frange-iff-vdomain*: $b \in_\circ \mathcal{R}_\bullet\ r \longleftrightarrow (\exists\, a{\in_\circ}\mathcal{D}_\bullet\ r.\ [a,\ b]_\circ \in_\circ r)$ ⟨*proof*⟩

lemma *frange-vsubset-ftimes*:
assumes *fpairs* $r \subseteq_\circ A \times_\bullet B$
shows $\mathcal{R}_\bullet\ r \subseteq_\circ B$
⟨*proof*⟩

lemma *fpairs-vsubset-fdomain-frange*[*simp*]: *fpairs* $r \subseteq_\circ (\mathcal{D}_\bullet\ r) \times_\bullet (\mathcal{R}_\bullet\ r)$
⟨*proof*⟩

lemma *frange-vsubset-VUnion2*: $\mathcal{R}_\bullet\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ(\bigcup_\circ r))$
⟨*proof*⟩

Field

definition *ffield* :: $V \Rightarrow V$
where $\mathit{ffield}\ r = \mathcal{D}_\bullet\ r \cup_\circ \mathcal{R}_\bullet\ r$

abbreviation *app-ffield* :: $V \Rightarrow V$ $(\langle\mathcal{F}_\bullet\rangle)$
where $\mathcal{F}_\bullet\ r \equiv \mathit{ffield}\ r$

Rules.

lemma *ffieldI1*[*intro*]:
assumes $a \in_\circ \mathcal{D}_\bullet\ r \cup_\circ \mathcal{R}_\bullet\ r$
shows $a \in_\circ \mathit{ffield}\ r$
⟨*proof*⟩

lemma *ffieldI2*[*intro*]:

assumes $[a,\ b]_\circ \in_\circ r$
shows $a \in_\circ \mathit{ffield}\ r$
⟨*proof*⟩

lemma *ffieldI3*[*intro*]:
assumes $[a,\ b]_\circ \in_\circ r$
shows $b \in_\circ \mathit{ffield}\ r$
⟨*proof*⟩

lemma *ffieldD*[*intro*]:
assumes $a \in_\circ \mathit{ffield}\ r$
shows $a \in_\circ \mathcal{D}_\bullet\ r \cup_\circ \mathcal{R}_\bullet\ r$
⟨*proof*⟩

lemma *ffieldE*[*elim*]:
assumes $a \in_\circ \mathit{ffield}\ r$ **and** $a \in_\circ \mathcal{D}_\bullet\ r \cup_\circ \mathcal{R}_\bullet\ r \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *ffield-pair*[*elim*]:
assumes $a \in_\circ \mathit{ffield}\ r$
obtains b **where** $[a,\ b]_\circ \in_\circ r \vee [b,\ a]_\circ \in_\circ r$
⟨*proof*⟩

lemma *ffield-iff*: $a \in_\circ \mathit{ffield}\ r \longleftrightarrow (\exists\, b.\ [a,\ b]_\circ \in_\circ r \vee [b,\ a]_\circ \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *ffield-vempty*[*simp*]: $\mathit{ffield}\ 0 = 0$ ⟨*proof*⟩

lemma *ffield-vsingleton*[*simp*]: $\mathit{ffield}\ (\mathit{set}\ \{[a,\ b]_\circ\}) = \mathit{set}\ \{a,\ b\}$ ⟨*proof*⟩

lemma *ffield-vdoubleton*[*simp*]:
$\mathit{ffield}\ (\mathit{set}\ \{[a,\ b]_\circ,\ [c,\ d]_\circ\}) = \mathit{set}\ \{a,\ b,\ c,\ d\}$
⟨*proof*⟩

lemma *ffield-mono*:
assumes $r \subseteq_\circ s$
shows $\mathit{ffield}\ r \subseteq_\circ \mathit{ffield}\ s$
⟨*proof*⟩

lemma *ffield-vinsert*[*simp*]:
$\mathit{ffield}\ (\mathit{vinsert}\ [a,\ b]_\circ\ r) = \mathit{set}\ \{a,\ b\} \cup_\circ (\mathit{ffield}\ r)$
⟨*proof*⟩

lemma *ffield-vunion*[*simp*]: $\mathit{ffield}\ (r \cup_\circ s) = \mathit{ffield}\ r \cup_\circ \mathit{ffield}\ s$
⟨*proof*⟩

Connections.

lemma *fid-on-ffield*[*simp*]: $\mathit{ffield}\ (\mathit{fid\text{-}on}\ A) = A$ ⟨*proof*⟩

lemma *fconst-on-ffield-ne*[*intro*, *simp*]:
assumes $A \neq 0$
shows $\mathit{ffield}\ (\mathit{fconst\text{-}on}\ A\ c) = \mathit{vinsert}\ c\ A$
⟨*proof*⟩

lemma *fconst-on-ffield-vempty*[*simp*]: $\mathit{ffield}\ (\mathit{fconst\text{-}on}\ 0\ c) = 0$ ⟨*proof*⟩

lemma *ffield-fconverse*[*simp*]: $\mathit{ffield}\ (r^{-1}{}_\bullet) = \mathit{ffield}\ r$ ⟨*proof*⟩

Special properties.

lemma *ffield-vsubset-VUnion2*: $\mathcal{F}_\bullet\ r \subseteq_\circ \bigcup_\circ(\bigcup_\circ(\bigcup_\circ r))$
⟨*proof*⟩

Image

definition *fimage* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹‘•› 90)
where r ‘• $A = \mathcal{R}_\bullet\ (r \restriction^l_\bullet A)$
notation *fimage* (**infixr** ‹‘•› 90)

lemma *fimage-small*[*simp*]: *small* $\{b.\ \exists\, a{\in_\circ}A.\ [a,\ b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *fimageI1*:
assumes $x \in_\circ \mathcal{R}_\bullet\ (r \restriction^l_\bullet A)$
shows $x \in_\circ r$ ‘• A
⟨*proof*⟩

lemma *fimageI2*[*intro*]:
assumes $[a,\ b]_\circ \in_\circ r$ **and** $a \in_\circ A$
shows $b \in_\circ r$ ‘• A
⟨*proof*⟩

lemma *fimageD*[*dest*]:
assumes $x \in_\circ r$ ‘• A
shows $x \in_\circ \mathcal{R}_\bullet\ (r \restriction^l_\bullet A)$
⟨*proof*⟩

lemma *fimageE*[*elim*]:
assumes $b \in_\circ r$ ‘• A
obtains a **where** $[a,\ b]_\circ \in_\circ r$ **and** $a \in_\circ A$
⟨*proof*⟩

lemma *fimage-iff*: $b \in_\circ r$ ‘• $A \longleftrightarrow (\exists\, a{\in_\circ}A.\ [a,\ b]_\circ \in_\circ r)$ ⟨*proof*⟩

Set operations.

lemma *fimage-vempty*[*simp*]: 0 ‘• $A = 0$ ⟨*proof*⟩

lemma *fimage-of-vempty*[*simp*]: r ‘• $0 = 0$ ⟨*proof*⟩

lemma *fimage-vsingleton-in*[*intro*, *simp*]:
assumes $a \in_\circ A$
shows *set* $\{[a,\ b]_\circ\}$ ‘• A = *set* $\{b\}$
⟨*proof*⟩

lemma *fimage-vsingleton-nin*[*intro*, *simp*]:
assumes $a \notin_\circ A$
shows *set* $\{[a,\ b]_\circ\}$ ‘• $A = 0$
⟨*proof*⟩

lemma *fimage-vsingleton-vinsert*[*intro*, *simp*]:
set $\{[a,\ b]_\circ\}$ ‘• *vinsert* a A = *set* $\{b\}$
⟨*proof*⟩

lemma *fimage-mono*:
assumes $r' \subseteq_\circ r$ **and** $A' \subseteq_\circ A$

shows $(r' \mathbin{`\bullet} A') \subseteq_\circ (r \mathbin{`\bullet} A)$
⟨*proof*⟩

lemma *fimage-vinsert*: $r \mathbin{`\bullet} (\mathit{vinsert}\ a\ A) = r \mathbin{`\bullet} \mathit{set}\ \{a\} \cup_\circ r \mathbin{`\bullet} A$ ⟨*proof*⟩

lemma *fimage-vunion-left*: $(r \cup_\circ s) \mathbin{`\bullet} A = r \mathbin{`\bullet} A \cup_\circ s \mathbin{`\bullet} A$ ⟨*proof*⟩

lemma *fimage-vunion-right*: $r \mathbin{`\bullet} (A \cup_\circ B) = r \mathbin{`\bullet} A \cup_\circ r \mathbin{`\bullet} B$ ⟨*proof*⟩

lemma *fimage-vintersection*: $r \mathbin{`\bullet} (A \cap_\circ B) \subseteq_\circ r \mathbin{`\bullet} A \cap_\circ r \mathbin{`\bullet} B$ ⟨*proof*⟩

lemma *fimage-vdiff*: $r \mathbin{`\bullet} A -_\circ r \mathbin{`\bullet} B \subseteq_\circ r \mathbin{`\bullet} (A -_\circ B)$ ⟨*proof*⟩

Special properties.

lemma *fimage-vsingleton-iff*[*iff*]: $b \in_\circ r \mathbin{`\bullet} \mathit{set}\ \{a\} \longleftrightarrow [a, b]_\circ \in_\circ r$ ⟨*proof*⟩

lemma *fimage-is-vempty*[*iff*]: $r \mathbin{`\bullet} A = 0 \longleftrightarrow \mathit{vdisjnt}\ (\mathcal{D}_\bullet\ r)\ A$ ⟨*proof*⟩

Connections.

lemma *fid-on-fimage*[*simp*]: $(\mathit{fid\text{-}on}\ A) \mathbin{`\bullet} B = A \cap_\circ B$ ⟨*proof*⟩

lemma *fimage-fconst-on-ne*[*simp*]:
assumes $B \cap_\circ A \neq 0$
shows $(\mathit{fconst\text{-}on}\ A\ c) \mathbin{`\bullet} B = \mathit{set}\ \{c\}$
⟨*proof*⟩

lemma *fimage-fconst-on-vempty*[*simp*]:
assumes *vdisjnt A B*
shows $(\mathit{fconst\text{-}on}\ A\ c) \mathbin{`\bullet} B = 0$
⟨*proof*⟩

lemma *fimage-fconst-on-vsubset-const*[*simp*]: $(\mathit{fconst\text{-}on}\ A\ c) \mathbin{`\bullet} B \subseteq_\circ \mathit{set}\ \{c\}$
⟨*proof*⟩

lemma *fcomp-frange*: $\mathcal{R}_\bullet\ (r \circ_\bullet s) = r \mathbin{`\bullet} (\mathcal{R}_\bullet\ s)$ ⟨*proof*⟩

lemma *fcomp-fimage*: $(r \circ_\bullet s) \mathbin{`\bullet} A = r \mathbin{`\bullet} (s \mathbin{`\bullet} A)$ ⟨*proof*⟩

lemma *fimage-fltrestriction*[*simp*]: $(r \restriction^l_\bullet A) \mathbin{`\bullet} B = r \mathbin{`\bullet} (A \cap_\circ B)$ ⟨*proof*⟩

lemma *fimage-frrestriction*[*simp*]: $(r \restriction^r_\bullet A) \mathbin{`\bullet} B = A \cap_\circ r \mathbin{`\bullet} B$ ⟨*proof*⟩

lemma *fimage-frestriction*[*simp*]: $(r \restriction_\bullet A) \mathbin{`\bullet} B = A \cap_\circ (r \mathbin{`\bullet} (A \cap_\circ B))$ ⟨*proof*⟩

lemma *fimage-fdomain*: $r \mathbin{`\bullet} \mathcal{D}_\bullet\ r = \mathcal{R}_\bullet\ r$ ⟨*proof*⟩

lemma *fimage-eq-imp-fcomp*:
assumes $f \mathbin{`\bullet} A = g \mathbin{`\bullet} B$
shows $(h \circ_\bullet f) \mathbin{`\bullet} A = (h \circ_\bullet g) \mathbin{`\bullet} B$
⟨*proof*⟩

Previous connections.

lemma *fcomp-rel-fconst-on-ftimes*: $r \circ_\bullet (\mathit{fconst\text{-}on}\ A\ c) = A \times_\bullet (r \mathbin{`\bullet} \mathit{set}\ \{c\})$
⟨*proof*⟩

Further special properties.

lemma *fimage-vsubset*:

assumes $r \subseteq_\circ A \times_\bullet B$
shows $r \mathbin{`\!`}_\bullet C \subseteq_\circ B$
⟨*proof*⟩

lemma *fimage-set-def*: $r \mathbin{`\!`}_\bullet A = set\ \{b.\ \exists a \in_\circ A.\ [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

lemma *fimage-vsingleton*: $r \mathbin{`\!`}_\bullet set\ \{a\} = set\ \{b.\ [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

lemma *fimage-strict-vsubset*: $f \mathbin{`\!`}_\bullet A \subseteq_\circ f \mathbin{`\!`}_\bullet \mathcal{D}_\bullet f$ ⟨*proof*⟩

Inverse image

definition *finvimage* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹−`\!`_› 90)
where $r -`\!`_\bullet A = r^{-1}{}_\bullet \mathbin{`\!`}_\bullet A$

lemma *finvimage-small*[*simp*]: $small\ \{a.\ \exists b \in_\circ A.\ [a, b]_\circ \in_\circ r\}$
⟨*proof*⟩

Rules.

lemma *finvimageI*[*intro*]:
assumes $[a, b]_\circ \in_\circ r$ **and** $b \in_\circ A$
shows $a \in_\circ r -`\!`_\bullet A$
⟨*proof*⟩

lemma *finvimageD*[*dest*]:
assumes $a \in_\circ r -`\!`_\bullet A$
shows $a \in_\circ \mathcal{D}_\bullet (r \restriction^r_\bullet A)$
⟨*proof*⟩

lemma *finvimageE*[*elim*]:
assumes $a \in_\circ r -`\!`_\bullet A$
obtains b **where** $[a, b]_\circ \in_\circ r$ **and** $b \in_\circ A$
⟨*proof*⟩

lemma *finvimageI1*:
assumes $a \in_\circ \mathcal{D}_\bullet (r \restriction^r_\bullet A)$
shows $a \in_\circ r -`\!`_\bullet A$
⟨*proof*⟩

lemma *finvimageD1*:
assumes $a \in_\circ r -`\!`_\bullet A$
shows $a \in_\circ \mathcal{D}_\bullet (r \restriction^r_\bullet A)$
⟨*proof*⟩

lemma *finvimageE1*:
assumes $a \in_\circ r -`\!`_\bullet A$ **and** $a \in_\circ \mathcal{D}_\bullet (r \restriction^r_\bullet A) \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *finvimageI2*:
assumes $a \in_\circ r^{-1}{}_\bullet \mathbin{`\!`}_\bullet A$
shows $a \in_\circ r -`\!`_\bullet A$
⟨*proof*⟩

lemma *finvimageD2*:
assumes $a \in_\circ r -`\!`_\bullet A$

shows $a \in_\circ r^{-1}_\bullet \mathbin{`}_\bullet A$
⟨*proof*⟩

lemma *finvimageE2*:
assumes $a \in_\circ r \mathbin{-`}_\circ A$ **and** $a \in_\circ r^{-1}_\circ \mathbin{`}_\circ A \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *finvimage-iff*: $a \in_\circ r \mathbin{-`}_\bullet A \longleftrightarrow (\exists\, b{\in_\circ}A.\ [a,\ b]_\circ \in_\circ r)$ ⟨*proof*⟩

lemma *finvimage-iff1*: $a \in_\circ r \mathbin{-`}_\bullet A \longleftrightarrow a \in_\circ \mathcal{D}_\bullet\ (r \restriction^r_\bullet A)$ ⟨*proof*⟩

lemma *finvimage-iff2*: $a \in_\circ r \mathbin{-`}_\bullet A \longleftrightarrow a \in_\circ r^{-1}_\bullet \mathbin{`}_\bullet A$ ⟨*proof*⟩

Set operations.

lemma *finvimage-vempty*[*simp*]: $0 \mathbin{-`}_\bullet A = 0$ ⟨*proof*⟩

lemma *finvimage-of-vempty*[*simp*]: $r \mathbin{-`}_\bullet 0 = 0$ ⟨*proof*⟩

lemma *finvimage-vsingleton-in*[*intro*, *simp*]:
assumes $b \in_\circ A$
shows *set* $\{[a,\ b]_\circ\} \mathbin{-`}_\bullet A =$ *set* $\{a\}$
⟨*proof*⟩

lemma *finvimage-vsingleton-nin*[*intro*, *simp*]:
assumes $b \notin_\circ A$
shows *set* $\{[a,\ b]_\circ\} \mathbin{-`}_\bullet A = 0$
⟨*proof*⟩

lemma *finvimage-vsingleton-vinsert*[*intro*, *simp*]:
set $\{[a,\ b]_\circ\} \mathbin{-`}_\bullet$ *vinsert* $b\ A =$ *set* $\{a\}$
⟨*proof*⟩

lemma *finvimage-mono*:
assumes $r' \subseteq_\circ r$ **and** $A' \subseteq_\circ A$
shows $(r' \mathbin{-`}_\bullet A') \subseteq_\circ (r \mathbin{-`}_\bullet A)$
⟨*proof*⟩

lemma *finvimage-vinsert*: $r \mathbin{-`}_\bullet (\mathit{vinsert}\ a\ A) = r \mathbin{-`}_\bullet \mathit{set}\ \{a\} \cup_\circ r \mathbin{-`}_\bullet A$ ⟨*proof*⟩

lemma *finvimage-vunion-left*: $(r \cup_\circ s) \mathbin{-`}_\bullet A = r \mathbin{-`}_\bullet A \cup_\circ s \mathbin{-`}_\bullet A$ ⟨*proof*⟩

lemma *finvimage-vunion-right*: $r \mathbin{-`}_\bullet (A \cup_\circ B) = r \mathbin{-`}_\bullet A \cup_\circ r \mathbin{-`}_\bullet B$ ⟨*proof*⟩

lemma *finvimage-vintersection*: $r \mathbin{-`}_\bullet (A \cap_\circ B) \subseteq_\circ r \mathbin{-`}_\bullet A \cap_\circ r \mathbin{-`}_\bullet B$ ⟨*proof*⟩

lemma *finvimage-vdiff*: $r \mathbin{-`}_\bullet A -_\circ r \mathbin{-`}_\bullet B \subseteq_\circ r \mathbin{-`}_\bullet (A -_\circ B)$ ⟨*proof*⟩

Special properties.

lemma *finvimage-set-def*: $r \mathbin{-`}_\bullet A =$ *set* $\{a.\ \exists\, b{\in_\circ}A.\ [a,\ b]_\circ \in_\circ r\}$ ⟨*proof*⟩

lemma *finvimage-eq-fdomain-frestriction*: $r \mathbin{-`}_\bullet A = \mathcal{D}_\bullet\ (r \restriction^r_\bullet A)$ ⟨*proof*⟩

lemma *finvimage-frange*[*simp*]: $r \mathbin{-`}_\bullet \mathcal{R}_\bullet\ r = \mathcal{D}_\bullet\ r$
⟨*proof*⟩

lemma *finvimage-frange-vsubset*[*simp*]:
assumes $\mathcal{R}_\bullet\ r \subseteq_\circ B$

shows $r -`_\bullet B = \mathcal{D}_\bullet\ r$
⟨*proof*⟩

Connections.

lemma *finvimage-fid-on*[*simp*]: (*fid-on A*) $-`_\bullet B = A \cap_\circ B$ ⟨*proof*⟩

lemma *finvimage-fconst-on-vsubset-fdomain*[*simp*]: (*fconst-on A c*) $-`_\bullet B \subseteq_\circ A$
⟨*proof*⟩

lemma *finvimage-fconst-on-ne*[*simp*]:
assumes $c \in_\circ B$
shows (*fconst-on A c*) $-`_\bullet B = A$
⟨*proof*⟩

lemma *finvimage-fconst-on-vempty*[*simp*]:
assumes $c \notin_\circ B$
shows (*fconst-on A c*) $-`_\bullet B = 0$
⟨*proof*⟩

lemma *finvimage-fcomp*: $(g \circ_\bullet f) -`_\bullet x = f -`_\bullet (g -`_\bullet x)$
⟨*proof*⟩

lemma *finvimage-fconverse*[*simp*]: $r^{-1}{}_\bullet -`_\bullet A = r\ `_\bullet A$ ⟨*proof*⟩

lemma *finvimage-flrestriction*[*simp*]: $(r \upharpoonright^l_\bullet A) -`_\bullet B = A \cap_\circ r -`_\bullet B$ ⟨*proof*⟩

lemma *finvimage-frrestriction*[*simp*]: $(r \upharpoonright^r_\bullet A) -`_\bullet B = (r -`_\bullet (A \cap_\circ B))$ ⟨*proof*⟩

lemma *finvimage-frestriction*[*simp*]: $(r \upharpoonright_\bullet A) -`_\bullet B = A \cap_\circ (r -`_\bullet (A \cap_\circ B))$
⟨*proof*⟩

Previous connections.

lemma *fdomain-fcomp*[*simp*]: $\mathcal{D}_\bullet\ (r \circ_\bullet s) = s -`_\bullet \mathcal{D}_\bullet\ r$ ⟨*proof*⟩

2.10.5 Classification of relations

Binary relation

locale *fbrelation* =
fixes $r :: V$
assumes *fbrelation*[*simp*]: *fpairs r* = *r*

locale *fbrelation-pair* = r_1: *fbrelation* r_1 + r_2: *fbrelation* r_2 **for** r_1 r_2

Rules.

lemma *fpairs-eqI*[*intro*!]:
assumes $\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = [a,\ b]_\circ$
shows *fpairs r* = *r*
⟨*proof*⟩

lemma *fpairs-eqD*[*dest*]:
assumes *fpairs r* = *r*
shows $\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = [a,\ b]_\circ$
⟨*proof*⟩

lemma *fpairs-eqE*[*elim*!]:
assumes *fpairs r* = *r* **and** $(\bigwedge x.\ x \in_\circ r \Longrightarrow \exists\, a\ b.\ x = [a,\ b]_\circ) \Longrightarrow P$
shows *P*

⟨*proof*⟩

lemmas *fbrelationI*[*intro*!] = *fbrelation.intro*
lemmas *fbrelationD*[*dest*!] = *fbrelation.fbrelation*

lemma *fbrelationE*[*elim*!]:
assumes *fbrelation r* **and** (*fpairs r* = *r*) ⟹ *P*
shows *P*
⟨*proof*⟩

lemma *fbrelationE*1:
assumes *fbrelation r* **and** *x* ∈$_\circ$ *r*
obtains *a b* **where** *x* = [*a*, *b*]$_\circ$
⟨*proof*⟩

lemma *fbrelationD*1[*dest*]:
assumes *fbrelation r* **and** *x* ∈$_\circ$ *r*
shows ∃ *a b*. *x* = [*a*, *b*]$_\circ$
⟨*proof*⟩

Set operations.

lemma *fbrelation-vsubset*:
assumes *fbrelation s* **and** *r* ⊆$_\circ$ *s*
shows *fbrelation r*
⟨*proof*⟩

lemma *fbrelation-vinsert*: *fbrelation* (*vinsert* [*a*, *b*]$_\circ$ *r*) ⟷ *fbrelation r*
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vinsertI*: *fbrelation* (*vinsert* [*a*, *b*]$_\circ$ *r*)
⟨*proof*⟩

lemma *fbrelation-vinsertD*[*dest*]:
assumes *fbrelation* (*vinsert* ⟨*a*, *b*⟩ *r*)
shows *fbrelation r*
⟨*proof*⟩

lemma *fbrelation-vunion*: *fbrelation* (*r* ∪$_\circ$ *s*) ⟷ *fbrelation r* ∧ *fbrelation s*
⟨*proof*⟩

lemma (**in** *fbrelation-pair*) *fbrelation-vunionI*: *fbrelation* (*r*$_1$ ∪$_\circ$ *r*$_2$)
⟨*proof*⟩

lemma *fbrelation-vunionD*[*dest*]:
assumes *fbrelation* (*r* ∪$_\circ$ *s*)
shows *fbrelation r* **and** *fbrelation s*
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vintersectionI*: *fbrelation* (*r* ∩$_\circ$ *s*)
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vdiffI*: *fbrelation* (*r* −$_\circ$ *s*)
⟨*proof*⟩

Connections.

lemma *fbrelation-vempty*: *fbrelation* 0 ⟨*proof*⟩

lemma *fbrelation-vsingleton*: *fbrelation* (*set* {[*a*, *b*]$_\circ$}) ⟨*proof*⟩

global-interpretation *frel-vsingleton*: *fbrelation* ‹*set* $\{[a, b]_\circ\}$›
⟨*proof*⟩

lemma *fbrelation-vdoubleton*: *fbrelation* (*set* $\{[a, b]_\circ, [c, d]_\circ\}$) ⟨*proof*⟩

lemma *fbrelation-sid-on*[*simp*]: *fbrelation* (*fid-on A*) ⟨*proof*⟩

lemma *fbrelation-fconst-on*[*simp*]: *fbrelation* (*fconst-on A c*) ⟨*proof*⟩

lemma (**in** *fbrelation-pair*) *fbrelation-fcomp*: *fbrelation* ($r_1 \circ_\bullet r_2$)
⟨*proof*⟩

sublocale *fbrelation-pair* ⊆ *fcomp*$_{21}$: *fbrelation* ‹$r_2 \circ_\bullet r_1$›
⟨*proof*⟩

sublocale *fbrelation-pair* ⊆ *fcomp*$_{12}$: *fbrelation* ‹$r_1 \circ_\bullet r_2$›
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-fconverse*: *fbrelation* ($r^{-1}{}_\bullet$)
⟨*proof*⟩

lemma *fbrelation-flrestriction*[*intro*, *simp*]: *fbrelation* ($r \restriction^l_\bullet A$) ⟨*proof*⟩

lemma *fbrelation-frrestriction*[*intro*, *simp*]: *fbrelation* ($r \restriction^r_\bullet A$) ⟨*proof*⟩

lemma *fbrelation-frestriction*[*intro*, *simp*]: *fbrelation* ($r \restriction_\bullet A$) ⟨*proof*⟩

Previous connections.

lemma (**in** *fbrelation*) *fconverse-fconverse*[*simp*]: $(r^{-1}{}_\bullet)^{-1}{}_\bullet = r$
⟨*proof*⟩

lemma (**in** *fbrelation-pair*) *fconverse-mono*[*simp*]: ${r_1}^{-1}{}_\bullet \subseteq_\circ {r_2}^{-1}{}_\bullet \longleftrightarrow r_1 \subseteq_\circ r_2$
⟨*proof*⟩

lemma (**in** *fbrelation-pair*) *fconverse-inject*[*simp*]: ${r_1}^{-1}{}_\bullet = {r_2}^{-1}{}_\bullet \longleftrightarrow r_1 = r_2$
⟨*proof*⟩

lemma (**in** *fbrelation*) *fconverse-vsubset-swap*-2:
assumes $r^{-1}{}_\bullet \subseteq_\circ s$
shows $r \subseteq_\circ s^{-1}{}_\bullet$
⟨*proof*⟩

lemma (**in** *fbrelation*) *flrestriction-fdomain*[*simp*]: $r \restriction^l_\bullet \mathcal{D}_\bullet\ r = r$
⟨*proof*⟩

lemma (**in** *fbrelation*) *frrestriction-frange*[*simp*]: $r \restriction^r_\bullet \mathcal{R}_\bullet\ r = r$
⟨*proof*⟩

Special properties.

lemma *vsubset-vtimes-fbrelation*:
assumes $r \subseteq_\circ A \times_\bullet B$
shows *fbrelation r*
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vintersection-vdomain*:
assumes *vdisjnt* ($\mathcal{D}_\bullet\ r$) ($\mathcal{D}_\bullet\ s$)
shows *vdisjnt r s*

⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vintersection-vrange*:
assumes *vdisjnt* ($\mathcal{R}_\bullet$ r) ($\mathcal{R}_\bullet$ s)
shows *vdisjnt* r s
⟨*proof*⟩

lemma (**in** *fbrelation*) *fbrelation-vintersection-vfield*:
assumes *vdisjnt* (*ffield* r) (*ffield* s)
shows *vdisjnt* r s
⟨*proof*⟩

lemma (**in** *fbrelation*) *vdomain-vrange-vtimes*: $r \subseteq_\circ \mathcal{D}_\bullet\ r \times_\bullet \mathcal{R}_\bullet\ r$
⟨*proof*⟩

lemma (**in** *fbrelation*) *fconverse-eq-frel*[*intro*, *simp*]:
assumes $\bigwedge a\ b.\ [a, b]_\circ \in_\circ r \Longrightarrow [b, a]_\circ \in_\circ r$
shows ${r^{-1}}_\bullet = r$
⟨*proof*⟩

lemma *fcomp-fconverse-frel-eq-frel-fbrelationI*:
assumes ${r^{-1}}_\bullet \circ_\bullet r = r$
shows *fbrelation* r
⟨*proof*⟩

Alternative forms of existing results.

lemmas [*intro*, *simp*] = *fbrelation.fconverse-fconverse*
and *fconverse-eq-frel*[*intro*, *simp*] = *fbrelation.fconverse-eq-frel*

context
fixes r_1 r_2
assumes r_1: *fbrelation* r_1
and r_2: *fbrelation* r_2
begin

lemmas-with[*OF fbrelation-pair.intro*[*OF* r_1 r_2]] :
fbrelation-fconverse-mono[*intro*, *simp*] = *fbrelation-pair.fconverse-mono*
and *fbrelation-frrestriction-srange*[*intro*, *simp*] =
fbrelation-pair.fconverse-inject

end

2.11 Further results related to the von Neumann hierarchy of sets

2.11.1 Background

The subsection presents several further auxiliary results about the von Neumann hierarchy of sets. The primary general reference for this section is [59].

2.11.2 Further properties of *Vfrom*

Reusable patterns.

lemma *Vfrom-Ord-bundle*:
assumes $A = A$ **and** $i = i$
shows *Vfrom A i* = *Vfrom A* (*rank i*) **and** *Ord* (*rank i*)
⟨*proof*⟩

lemma *Vfrom-in-bundle*:
assumes $i \in_\circ j$ **and** $A = A$ **and** $B = B$
shows *Vfrom A i* = *Vfrom A* (*rank i*)
and *Ord* (*rank i*)
and *Vfrom B j* = *Vfrom B* (*rank j*)
and *Ord* (*rank j*)
and *rank i* $\in_\circ$ *rank j*
⟨*proof*⟩

Elementary corollaries.

lemma *Ord-Vset-in-Vset-succI*[*intro*]:
assumes *Ord* α
shows *Vset* $\alpha \in_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemma *Ord-in-in-VsetI*[*intro*]:
assumes *Ord* α **and** $a \in_\circ \alpha$
shows $a \in_\circ$ *Vset* α
⟨*proof*⟩

Transitivity of the constant *Vfrom*.

lemma *Vfrom-trans*[*intro*]:
assumes *Transset A* **and** $x \in_\circ X$ **and** $X \in_\circ$ *Vfrom A i*
shows $x \in_\circ$ *Vfrom A i*
⟨*proof*⟩

lemma *Vset-trans*[*intro*]:
assumes $x \in_\circ X$ **and** $X \in_\circ$ *Vset i*
shows $x \in_\circ$ *Vset i*
⟨*proof*⟩

Monotonicity of the constant *Vfrom*.

lemma *Vfrom-in-mono*:
assumes $A \subseteq_\circ B$ **and** $i \in_\circ j$
shows *Vfrom A i* $\in_\circ$ *Vfrom B j*
⟨*proof*⟩

lemmas *Vset-in-mono* = *Vfrom-in-mono*[*OF order-refl*, *of* - - 0]

lemma *Vfrom-vsubset-mono*:

assumes $A \subseteq_\circ B$ **and** $i \subseteq_\circ j$
shows *Vfrom* $A\ i \subseteq_\circ$ *Vfrom* $B\ j$
⟨*proof*⟩

lemmas *Vset-vsubset-mono* = *Vfrom-vsubset-mono*[*OF order-refl*, *of* - - 0]

lemma *arg*1-*vsubset-Vfrom*: $a \subseteq_\circ$ *Vfrom* $a\ i$ ⟨*proof*⟩

lemma *VPow-vsubset-Vset*:
— Based on Theorem 9.10 from [59]
assumes $X \in_\circ$ *Vset* i
shows *VPow* $X \subseteq_\circ$ *Vset* i
⟨*proof*⟩

lemma *Vfrom-vsubset-VPow-Vfrom*:
assumes *Transset* A
shows *Vfrom* $A\ i \subseteq_\circ$ *VPow* (*Vfrom* $A\ i$)
⟨*proof*⟩

lemma *arg*1-*vsubset-VPow-Vfrom*:
assumes *Transset* A
shows $A \subseteq_\circ$ *VPow* (*Vfrom* $A\ i$)
⟨*proof*⟩

2.11.3 Operations closed with respect to *Vset*

Empty set.

lemma *Limit-vempty-in-VsetI*:
assumes *Limit* α
shows $0 \in_\circ$ *Vset* α
⟨*proof*⟩

Subset.

lemma *vsubset-in-VsetI*[*intro*]:
assumes $a \subseteq_\circ A$ **and** $A \in_\circ$ *Vset* i
shows $a \in_\circ$ *Vset* i
⟨*proof*⟩

lemma *Ord-vsubset-in-Vset-succI*:
assumes *Ord* α **and** $A \subseteq_\circ$ *Vset* α
shows $A \in_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

Power set.

lemma *Limit-VPow-in-VsetI*[*intro*]:
assumes *Limit* α **and** $A \in_\circ$ *Vset* α
shows *VPow* $A \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *VPow-in-Vset-revD*:
assumes *VPow* $A \in_\circ$ *Vset* i
shows $A \in_\circ$ *Vset* i
⟨*proof*⟩

lemma *Ord-VPow-in-Vset-succI*:
assumes *Ord* α **and** $a \in_\circ$ *Vset* α
shows *VPow* $a \in_\circ$ *Vset* (*succ* α)

⟨*proof*⟩

lemma *Ord-VPow-in-Vset-succD*:
assumes *Ord* α **and** *VPow* $a \in_\circ$ *Vset* (*succ* α)
shows $a \in_\circ$ *Vset* α
⟨*proof*⟩

Union of elements.

lemma *VUnion-in-VsetI*[*intro*]:
assumes $A \in_\circ$ *Vset* i
shows $\bigcup_\circ A \in_\circ$ *Vset* i
⟨*proof*⟩

lemma *Limit-VUnion-in-VsetD*:
assumes *Limit* α **and** $\bigcup_\circ A \in_\circ$ *Vset* α
shows $A \in_\circ$ *Vset* α
⟨*proof*⟩

Intersection of elements.

lemma *VInter-in-VsetI*[*intro*]:
assumes $A \in_\circ$ *Vset* α
shows $\bigcap_\circ A \in_\circ$ *Vset* α
⟨*proof*⟩

Singleton.

lemma *Limit-vsingleton-in-VsetI*[*intro*]:
assumes *Limit* α **and** $a \in_\circ$ *Vset* α
shows *set* $\{a\} \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Limit-vsingleton-in-VsetD*:
assumes *set* $\{a\} \in_\circ$ *Vset* α
shows $a \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-vsingleton-in-Vset-succI*:
assumes *Ord* α **and** $a \in_\circ$ *Vset* α
shows *set* $\{a\} \in_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

Doubleton.

lemma *Limit-vdoubleton-in-VsetI*[*intro*]:
assumes *Limit* α **and** $a \in_\circ$ *Vset* α **and** $b \in_\circ$ *Vset* α
shows *set* $\{a, b\} \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *vdoubleton-in-VsetD*:
assumes *set* $\{a, b\} \in_\circ$ *Vset* α
shows $a \in_\circ$ *Vset* α **and** $b \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-vdoubleton-in-Vset-succI*:
assumes *Ord* α **and** $a \in_\circ$ *Vset* α **and** $b \in_\circ$ *Vset* α
shows *set* $\{a, b\} \in_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

Pairwise union.

lemma *vunion-in-VsetI*[*intro*]:
assumes $a \in_\circ Vset\ i$ **and** $b \in_\circ Vset\ i$
shows $a \cup_\circ b \in_\circ Vset\ i$
⟨*proof*⟩

lemma *vunion-in-VsetD*:
assumes $a \cup_\circ b \in_\circ Vset\ \alpha$
shows $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
⟨*proof*⟩

Pairwise intersection.

lemma *vintersection-in-VsetI*[*intro*]:
assumes $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
shows $a \cap_\circ b \in_\circ Vset\ \alpha$
⟨*proof*⟩

Set difference.

lemma *vdiff-in-VsetI*[*intro*]:
assumes $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
shows $a -_\circ b \in_\circ Vset\ \alpha$
⟨*proof*⟩

vinsert.

lemma *vinsert-in-VsetI*[*intro*]:
assumes *Limit* α **and** $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
shows *vinsert* $a\ b \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *vinsert-in-Vset-succI*[*intro*]:
assumes *Ord* α **and** $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
shows *vinsert* $a\ b \in_\circ Vset\ (succ\ \alpha)$
⟨*proof*⟩

lemma *vinsert-in-Vset-succI'*[*intro*]:
assumes *Ord* α **and** $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ (succ\ \alpha)$
shows *vinsert* $a\ b \in_\circ Vset\ (succ\ \alpha)$
⟨*proof*⟩

lemma *vinsert-in-VsetD*:
assumes *vinsert* $a\ b \in_\circ Vset\ \alpha$
shows $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-insert-in-VsetI*:
assumes [*intro*]: *Limit* α
and [*simp*]: *small* x
and *set* $x \in_\circ Vset\ \alpha$
and [*intro*]: $a \in_\circ Vset\ \alpha$
shows *set* (*insert* $a\ x$) $\in_\circ Vset\ \alpha$
⟨*proof*⟩

Pair.

lemma *Limit-vpair-in-VsetI*[*intro*]:
assumes *Limit* α **and** $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
shows $\langle a, b \rangle \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *vpair-in-VsetD*[*intro*]:
assumes $\langle a,\ b\rangle \in_\circ Vset\ \alpha$
shows $a \in_\circ Vset\ \alpha$ **and** $b \in_\circ Vset\ \alpha$
⟨*proof*⟩

Cartesian product.

lemma *Limit-vtimes-in-VsetI*[*intro*]:
assumes *Limit* α **and** $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$
shows $A \times_\circ B \in_\circ Vset\ \alpha$
⟨*proof*⟩

Binary relations.

lemma (in *vbrelation*) *vbrelation-Limit-in-VsetI*[*intro*]:
assumes *Limit* α **and** $\mathcal{D}_\circ\ r \in_\circ Vset\ \alpha$ **and** $\mathcal{R}_\circ\ r \in_\circ Vset\ \alpha$
shows $r \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma
assumes $r \in_\circ Vset\ \alpha$
shows *vdomain-in-VsetI*: $\mathcal{D}_\circ\ r \in_\circ Vset\ \alpha$
and *vrange-in-VsetI*: $\mathcal{R}_\circ\ r \in_\circ Vset\ \alpha$
and *vfield-in-VsetI*: $\mathcal{F}_\circ\ r \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in *vbrelation*) *vbrelation-Limit-vsubset-VsetI*:
assumes *Limit* α **and** $\mathcal{D}_\circ\ r \subseteq_\circ Vset\ \alpha$ **and** $\mathcal{R}_\circ\ r \subseteq_\circ Vset\ \alpha$
shows $r \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma
assumes $r \in_\circ Vset\ \alpha$
shows *fdomain-in-VsetI*: $\mathcal{D}_\bullet\ r \in_\circ Vset\ \alpha$
and *frange-in-VsetI*: $\mathcal{R}_\bullet\ r \in_\circ Vset\ \alpha$
and *ffield-in-VsetI*: $\mathcal{F}_\bullet\ r \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in *vsv*) *vsv-Limit-vrange-in-VsetI*[*intro*]:
assumes *Limit* α **and** $\mathcal{R}_\circ\ r \subseteq_\circ Vset\ \alpha$ **and** *vfinite* $(\mathcal{D}_\circ\ r)$
shows $\mathcal{R}_\circ\ r \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in *vsv*) *vsv-Limit-vsv-in-VsetI*[*intro*]:
assumes *Limit* α
and $\mathcal{D}_\circ\ r \in_\circ Vset\ \alpha$
and $\mathcal{R}_\circ\ r \subseteq_\circ Vset\ \alpha$
and *vfinite* $(\mathcal{D}_\circ\ r)$
shows $r \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vcomp-in-VsetI*:
assumes *Limit* α **and** $r \in_\circ Vset\ \alpha$ **and** $s \in_\circ Vset\ \alpha$
shows $r \circ_\circ s \in_\circ Vset\ \alpha$
⟨*proof*⟩

Operations on indexed families of sets.

lemma *Limit-vifintersection-in-VsetI*:
assumes *Limit* α **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \in_\circ Vset\ \alpha$ **and** *vfinite I*

shows $(\bigcap_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vifunion-in-VsetI*:
assumes *Limit* α **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \in_\circ Vset\ \alpha$ **and** *vfinite I*
shows $(\bigcup_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vifunion-in-Vset-if-VLambda-in-VsetI*:
assumes *Limit* α **and** *VLambda I A* $\in_\circ$ *Vset* α
shows $(\bigcup_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vproduct-in-VsetI*:
assumes *Limit* α
and $I \in_\circ Vset\ \alpha$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \in_\circ Vset\ \alpha$
and *vfinite I*
shows $(\prod_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vproduct-in-Vset-if-VLambda-in-VsetI*:
assumes *Limit* α **and** *VLambda I A* $\in_\circ$ *Vset* α
shows $(\prod_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vdunion-in-Vset-if-VLambda-in-VsetI*:
assumes *Limit* α **and** *VLambda I A* $\in_\circ$ *Vset* α
shows $(\coprod_\circ i \in_\circ I.\ A\ i) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *vrange-vprojection-in-VsetI*:
assumes *Limit* α
and $A \in_\circ Vset\ \alpha$
and $\bigwedge f.\ f \in_\circ A \Longrightarrow vsv\ f$
and $\bigwedge f.\ f \in_\circ A \Longrightarrow x \in_\circ \mathcal{D}_\circ\ f$
shows $\mathcal{R}_\circ\ (\lambda f \in_\circ A.\ f(\!|x|\!)) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-vcpower-in-VsetI*:
assumes *Limit* α **and** $n \in_\circ Vset\ \alpha$ **and** $A \in_\circ Vset\ \alpha$ **and** *vfinite n*
shows $A\ \widehat{}_\times\ n \in_\circ Vset\ \alpha$
⟨*proof*⟩

Finite sets.

lemma *Limit-vfinite-in-VsetI*:
assumes *Limit* α **and** $A \subseteq_\circ Vset\ \alpha$ **and** *vfinite A*
shows $A \in_\circ Vset\ \alpha$
⟨*proof*⟩

Ordinal numbers.

lemma *Limit-omega-in-VsetI*:
assumes *Limit* α
shows $a_{\mathbb{N}} \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma *Limit-succ-in-VsetI*:
assumes *Limit* α **and** $a \in_\circ Vset\ \alpha$

shows *succ a* $\in_\circ$ *Vset* α
⟨*proof*⟩

Sequences.

lemma (**in** *vfsequence*) *vfsequence-Limit-vcons-in-VsetI*:
assumes *Limit* α **and** $x \in_\circ$ *Vset* α **and** $xs \in_\circ$ *Vset* α
shows *vcons xs x* $\in_\circ$ *Vset* α
⟨*proof*⟩

ftimes.

lemma *Limit-ftimes-in-VsetI*:
assumes *Limit* α **and** $A \in_\circ$ *Vset* α **and** $B \in_\circ$ *Vset* α
shows $A \times_\bullet B \in_\circ$ *Vset* α
⟨*proof*⟩

Auxiliary results.

lemma *vempty-in-Vset-succ*[*simp*, *intro*]: $0 \in_\circ$ *Vfrom a* (*succ b*)
⟨*proof*⟩

lemma *Limit-vid-on-in-Vset*:
assumes *Limit* α **and** $A \in_\circ$ *Vset* α
shows *vid-on A* $\in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-vpair-in-Vset-succI*[*intro*]:
assumes *Ord* α **and** $a \in_\circ$ *Vset* α **and** $b \in_\circ$ *Vset* α
shows $\langle a, b\rangle \in_\circ$ *Vset* (*succ* (*succ* α))
⟨*proof*⟩

lemma *Limit-vifunion-vsubset-VsetI*:
assumes *Limit* α **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \in_\circ$ *Vset* α
shows $(\bigcup_\circ i \in_\circ I.\ A\ i) \subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma *Limit-vproduct-vsubset-Vset-succI*:
assumes *Limit* α **and** $I \in_\circ$ *Vset* α **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \subseteq_\circ$ *Vset* α
shows $(\prod_\circ i \in_\circ I.\ A\ i) \subseteq_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemma *Limit-vproduct-vsubset-Vset-succI'*:
assumes *Limit* α **and** $I \in_\circ$ *Vset* α **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow A\ i \in_\circ$ *Vset* α
shows $(\prod_\circ i \in_\circ I.\ A\ i) \subseteq_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemma (**in** *vfsequence*) *vfsequence-Ord-vcons-in-Vset-succI*:
assumes *Ord* α
and $\omega \in_\circ \alpha$
and $x \in_\circ$ *Vset* α
and *xs* $\in_\circ$ *Vset* (*succ* (*succ* (*succ* α)))
shows *vcons xs x* $\in_\circ$ *Vset* (*succ* (*succ* (*succ* α)))
⟨*proof*⟩

lemma *Limit-VUnion-vdomain-in-VsetI*:
assumes *Limit* α **and** $Q \in_\circ$ *Vset* α
shows $(\bigcup_\circ r \in_\circ Q.\ \mathcal{D}_\circ\ r) \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Limit-VUnion-vrange-in-VsetI*:
assumes *Limit* α **and** $Q \in_\circ$ *Vset* α
shows $(\bigcup_\circ r \in_\circ Q.\ \mathcal{R}_\circ\ r) \in_\circ$ *Vset* α
⟨*proof*⟩

2.11.4 Axioms for *Vset* α

The subsection demonstrates that the axioms of ZFC except for the Axiom Schema of Replacement hold in *Vset* α for any limit ordinal α such that $\omega \in_\circ \alpha$[1].

locale $\mathcal{Z}$ =
fixes α
assumes *Limit-α*[*intro*, *simp*]: *Limit* α
and *omega-in-α*[*intro*, *simp*]: $\omega \in_\circ \alpha$
begin

lemmas [*intro*] = $\mathcal{Z}$*-axioms*

lemma *vempty-Z-def*: 0 = *set* $\{x.\ x \neq x\}$ ⟨*proof*⟩

lemma *vempty-is-zet*[*intro*, *simp*]: $0 \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Extensionality*:
assumes $a \in_\circ$ *Vset* α **and** $x = y$ **and** $x \in_\circ a$
shows $y \in_\circ a$ **and** $x \in_\circ$ *Vset* α **and** $y \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Pairing*:
assumes $a \in_\circ$ *Vset* α **and** $b \in_\circ$ *Vset* α
shows *set* $\{a, b\} \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Unions*:
assumes $a \in_\circ$ *Vset* α
shows $\bigcup_\circ a \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Powers*:
assumes $a \in_\circ$ *Vset* α
shows *VPow* $a \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Regularity*:
assumes $a \neq 0$ **and** $a \in_\circ$ *Vset* α
obtains x **where** $x \in_\circ a$ **and** $x \cap_\circ a = 0$
⟨*proof*⟩

lemma *Axiom-of-Infinity*: $\omega \in_\circ$ *Vset* α
⟨*proof*⟩

lemma *Axiom-of-Choice*:
assumes $A \in_\circ$ *Vset* α
obtains f **where** $f \in_\circ$ *Vset* α **and** $\bigwedge x.\ x \in_\circ A \Longrightarrow x \neq 0 \Longrightarrow f(\!|x|\!) \in_\circ x$
⟨*proof*⟩

[1] The presentation of the axioms is loosely based on the statement of the axioms of ZFC in Chapters 1-11 in [59].

end

Trivial corollaries.

lemma (in $\mathcal{Z}$**)** *Ord-α*: *Ord* α ⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $\mathcal{Z}$*-Vset-ω2-vsubset-Vset*: *Vset* $(\omega + \omega) \subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $\mathcal{Z}$*-Limit-αω*: *Limit* $(\alpha + \omega)$ ⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $\mathcal{Z}$*-α-αω*: $\alpha \in_\circ \alpha + \omega$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** $\mathcal{Z}$*-ω-αω*: $\omega \in_\circ \alpha + \omega$
⟨*proof*⟩

lemma $\mathcal{Z}$*-ωω*: $\mathcal{Z}$ $(\omega + \omega)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *in-omega-in-omega-plus*[*intro*]:
assumes $a \in_\circ \omega$
shows $a \in_\circ$ *Vset* $(\alpha + \omega)$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *ord-of-nat-in-Vset*[*simp*]: $a_{\mathbb{N}} \in_\circ$ *Vset* α ⟨*proof*⟩

vfsequences-on.

lemma (in $\mathcal{Z}$**)** *vfsequences-on-in-VsetI*:
assumes $X \in_\circ$ *Vset* α
shows *vfsequences-on* $X \in_\circ$ *Vset* α
⟨*proof*⟩

2.11.5 Existence of a disjoint subset in *Vset* α

definition *mk-doubleton* :: $V \Rightarrow V \Rightarrow V$
where *mk-doubleton* $X\ a$ = *set* $\{a, X\}$

definition *mk-doubleton-image* :: $V \Rightarrow V \Rightarrow V$
where *mk-doubleton-image* $X\ Y$ = *set* (*mk-doubleton* Y ‘ *elts* X)

lemma *inj-on-mk-doubleton*: *inj-on* (*mk-doubleton* X) (*elts* X)
⟨*proof*⟩

lemma *mk-doubleton-image-vsubset-veqpoll*:
assumes $X \subseteq_\circ Y$
shows *mk-doubleton-image* $X\ X \approx_\circ$ *mk-doubleton-image* $X\ Y$
⟨*proof*⟩

lemma *mk-doubleton-image-veqpoll*:
assumes $X \subseteq_\circ Y$
shows $X \approx_\circ$ *mk-doubleton-image* $X\ Y$
⟨*proof*⟩

lemma *vdisjnt-mk-doubleton-image*: *vdisjnt* (*mk-doubleton-image* $X\ Y$) Y
⟨*proof*⟩

lemma *Limit-mk-doubleton-image-vsubset-Vset*:
assumes *Limit* α **and** $X \subseteq_\circ Y$ **and** $Y \in_\circ$ *Vset* α

shows *mk-doubleton-image X Y* $\subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-mk-doubleton-image-vsubset-Vset-succ*:
assumes *Ord* α **and** $X \subseteq_\circ Y$ **and** $Y \in_\circ$ *Vset* α
shows *mk-doubleton-image X Y* $\subseteq_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemma *Limit-ex-eqpoll-vdisjnt*:
assumes *Limit* α **and** $X \subseteq_\circ Y$ **and** $Y \in_\circ$ *Vset* α
obtains Z **where** $X \approx_\circ Z$ **and** *vdisjnt Z Y* **and** $Z \subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-ex-eqpoll-vdisjnt*:
assumes *Ord* α **and** $X \subseteq_\circ Y$ **and** $Y \in_\circ$ *Vset* α
obtains Z **where** $X \approx_\circ Z$ **and** *vdisjnt Z Y* **and** $Z \subseteq_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

2.12 n-ary operation

2.12.1 Partial n-ary operation

locale *pnop* = *vsv f* **for** *A n f* :: *V* +
  assumes *pnop-n*: $n \in_\circ \omega$
    and *pnop-vdomain*: $\mathcal{D}_\circ\ f \subseteq_\circ A\ \widehat{\times}\ n$
    and *pnop-vrange*: $\mathcal{R}_\circ\ f \subseteq_\circ A$

Rules.

lemma *pnopI*[*intro*]:
  assumes *vsv f*
    and $n \in_\circ \omega$
    and $\mathcal{D}_\circ\ f \subseteq_\circ A\ \widehat{\times}\ n$
    and $\mathcal{R}_\circ\ f \subseteq_\circ A$
  shows *pnop A n f*
  ⟨*proof*⟩

lemma *pnopD*[*dest*]:
  assumes *pnop A n f*
  shows *vsv f*
    and $n \in_\circ \omega$
    and $\mathcal{D}_\circ\ f \subseteq_\circ A\ \widehat{\times}\ n$
    and $\mathcal{R}_\circ\ f \subseteq_\circ A$
  ⟨*proof*⟩

lemma *pnopE*[*elim*]:
  assumes *pnop A n f*
  obtains *vsv f*
    and $n \in_\circ \omega$
    and $\mathcal{D}_\circ\ f \subseteq_\circ A\ \widehat{\times}\ n$
    and $\mathcal{R}_\circ\ f \subseteq_\circ A$
  ⟨*proof*⟩

2.12.2 Total n-ary operation

locale *nop* = *vsv f* **for** *A n f* :: *V* +
  assumes *nop-n*: $n \in_\circ \omega$
    and *nop-vdomain*: $\mathcal{D}_\circ\ f = A\ \widehat{\times}\ n$
    and *nop-vrange*: $\mathcal{R}_\circ\ f \subseteq_\circ A$

sublocale *nop* ⊆ *pnop A n f*
⟨*proof*⟩

Rules.

lemma *nopI*[*intro*]:
  assumes *vsv f*
    and $n \in_\circ \omega$
    and $\mathcal{D}_\circ\ f = A\ \widehat{\times}\ n$
    and $\mathcal{R}_\circ\ f \subseteq_\circ A$
  shows *nop A n f*
  ⟨*proof*⟩

lemma *nopD*[*dest*]:
  assumes *nop A n f*
  shows *vsv f*
    and $n \in_\circ \omega$
    and $\mathcal{D}_\circ\ f = A\ \widehat{\times}\ n$

and $\mathcal{R}_\circ\ f \subseteq_\circ A$
⟨*proof*⟩

lemma *nopE*[*elim*]:
assumes *nop A n f*
obtains *vsv f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and $\mathcal{R}_\circ\ f \subseteq_\circ A$
⟨*proof*⟩

2.12.3 Injective n-ary operation

locale *nop-v11* = *v11 f* **for** *A n f* :: *V* +
assumes *nop-v11-n*: $n \in_\circ \omega$
and *nop-v11-vdomain*: $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and *nop-v11-vrange*: $\mathcal{R}_\circ\ f \subseteq_\circ A$

sublocale *nop-v11* ⊆ *nop*
⟨*proof*⟩

Rules.

lemma *nop-v11I*[*intro*]:
assumes *v11 f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and $\mathcal{R}_\circ\ f \subseteq_\circ A$
shows *nop-v11 A n f*
⟨*proof*⟩

lemma *nop-v11D*[*dest*]:
assumes *nop-v11 A n f*
shows *v11 f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and $\mathcal{R}_\circ\ f \subseteq_\circ A$
⟨*proof*⟩

lemma *nop-v11E*[*elim*]:
assumes *nop-v11 A n f*
obtains *v11 f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and $\mathcal{R}_\circ\ f \subseteq_\circ A$
⟨*proof*⟩

2.12.4 Surjective n-ary operation

locale *nop-onto* = *vsv f* **for** *A n f* :: *V* +
assumes *nop-onto-n*: $n \in_\circ \omega$
and *nop-onto-vdomain*: $\mathcal{D}_\circ\ f = A\ ^\frown_\times\ n$
and *nop-onto-vrange*: $\mathcal{R}_\circ\ f = A$

sublocale *nop-onto* ⊆ *nop*
⟨*proof*⟩

Rules.

lemma *nop-ontoI*[*intro*]:

assumes *vsv f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and $\mathcal{R}_\circ\ f = A$
shows *nop-onto A n f*
⟨*proof*⟩

lemma *nop-ontoD*[*dest*]:
assumes *nop-onto A n f*
shows *vsv f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and $\mathcal{R}_\circ\ f = A$
⟨*proof*⟩

lemma *nop-ontoE*[*elim*]:
assumes *nop-onto A n f*
obtains *vsv f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and $\mathcal{R}_\circ\ f = A$
⟨*proof*⟩

2.12.5 Bijective n-ary operation

locale *nop-bij* = *v*11 *f* **for** *A n f* :: *V* +
assumes *nop-bij-n*: $n \in_\circ \omega$
and *nop-bij-vdomain*: $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and *nop-bij-vrange*: $\mathcal{R}_\circ\ f = A$

sublocale *nop-bij* ⊆ *nop-v*11
⟨*proof*⟩

sublocale *nop-bij* ⊆ *nop-onto*
⟨*proof*⟩

Rules.

lemma *nop-bijI*[*intro*]:
assumes *v*11 *f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and $\mathcal{R}_\circ\ f = A$
shows *nop-bij A n f*
⟨*proof*⟩

lemma *nop-bijD*[*dest*]:
assumes *nop-bij A n f*
shows *v*11 *f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$
and $\mathcal{R}_\circ\ f = A$
⟨*proof*⟩

lemma *nop-bijE*[*elim*]:
assumes *nop-bij A n f*
obtains *v*11 *f*
and $n \in_\circ \omega$
and $\mathcal{D}_\circ\ f = A \mathbin{\widehat{\times}} n$

and $\mathcal{R}_\circ\ f = A$
⟨*proof*⟩

2.12.6 Scalar

locale *scalar* =
fixes $A\ f$
assumes *scalar-nop*: $nop\ A\ 0\ f$

sublocale $scalar \subseteq nop\ A\ 0\ f$
rewrites *scalar-vdomain*[*simp*]: $A\ \widehat{}_\times\ 0 = set\ \{0\}$
⟨*proof*⟩

Rules.

lemmas *scalarI*[*intro*] = *scalar.intro*

lemma *scalarD*[*dest*]:
assumes $scalar\ A\ f$
shows $nop\ A\ 0\ f$
⟨*proof*⟩

lemma *scalarE*[*elim*]:
assumes $scalar\ A\ f$
obtains $nop\ A\ 0\ f$
⟨*proof*⟩

2.12.7 Unary operation

locale $unop = nop\ A\ ‹1_{\mathbf{N}}›\ f$ **for** $A\ f$

Rules.

lemmas *unopI*[*intro*] = *unop.intro*

lemma *unopD*[*dest*]:
assumes $unop\ A\ f$
shows $nop\ A\ (1_{\mathbf{N}})\ f$
⟨*proof*⟩

lemma *unopE*[*elim*]:
assumes $unop\ A\ f$
obtains $nop\ A\ (1_{\mathbf{N}})\ f$
⟨*proof*⟩

2.12.8 Injective unary operation

locale $unop\text{-}v11 = nop\text{-}v11\ A\ ‹1_{\mathbf{N}}›\ f$ **for** $A\ f$

sublocale $unop\text{-}v11 \subseteq unop\ A\ f$ ⟨*proof*⟩

Rules.

lemma *unop-v11I*[*intro*]:
assumes $nop\text{-}v11\ A\ (1_{\mathbf{N}})\ f$
shows $unop\text{-}v11\ A\ f$
⟨*proof*⟩

lemma *unop-v11D*[*dest*]:
assumes $unop\text{-}v11\ A\ f$
shows $nop\text{-}v11\ A\ (1_{\mathbf{N}})\ f$

⟨*proof*⟩

lemma *unop-v*11*E*[*elim*]:
assumes *unop-v*11 *A f*
obtains *nop-v*11 *A* (1$_{\mathbb{N}}$) *f*
⟨*proof*⟩

2.12.9 Surjective unary operation

locale *unop-onto* = *nop-onto A* ‹1$_{\mathbb{N}}$› *f* **for** *A f*

sublocale *unop-onto* ⊆ *unop A f* ⟨*proof*⟩

Rules.

lemma *unop-ontoI*[*intro*]:
assumes *nop-onto A* (1$_{\mathbb{N}}$) *f*
shows *unop-onto A f*
⟨*proof*⟩

lemma *unop-ontoD*[*dest*]:
assumes *unop-onto A f*
shows *nop-onto A* (1$_{\mathbb{N}}$) *f*
⟨*proof*⟩

lemma *unop-ontoE*[*elim*]:
assumes *unop-onto A f*
obtains *nop-onto A* (1$_{\mathbb{N}}$) *f*
⟨*proof*⟩

lemma *unop-ontoI′*[*intro*]:
assumes *unop A f* **and** *A* ⊆$_{\circ}$ $\mathcal{R}_{\circ}$ *f*
shows *unop-onto A f*
⟨*proof*⟩

2.12.10 Bijective unary operation

locale *unop-bij* = *nop-bij A* ‹1$_{\mathbb{N}}$› *f* **for** *A f*

sublocale *unop-bij* ⊆ *unop-v*11 *A f*
⟨*proof*⟩

sublocale *unop-bij* ⊆ *unop-onto A f*
⟨*proof*⟩

Rules.

lemma *unop-bijI*[*intro*]:
assumes *nop-bij A* (1$_{\mathbb{N}}$) *f*
shows *unop-bij A f*
⟨*proof*⟩

lemma *unop-bijD*[*dest*]:
assumes *unop-bij A f*
shows *nop-bij A* (1$_{\mathbb{N}}$) *f*
⟨*proof*⟩

lemma *unop-bijE*[*elim*]:
assumes *unop-bij A f*
obtains *nop-bij A* (1$_{\mathbb{N}}$) *f*

⟨*proof*⟩

lemma *unop-bijI'*[*intro*]:
assumes *unop-v11* A f **and** $A \subseteq_\circ \mathcal{R}_\circ\ f$
shows *unop-bij* A f
⟨*proof*⟩

2.12.11 Partial binary operation

locale *pbinop* = *pnop* A ‹$2_\mathbb{N}$› f **for** A f

sublocale *pbinop* ⊆ *dom*: *fbrelation* ‹$\mathcal{D}_\circ\ f$›
⟨*proof*⟩

Rules.

lemmas *pbinopI*[*intro*] = *pbinop.intro*

lemma *pbinopD*[*dest*]:
assumes *pbinop* A f
shows *pnop* A ($2_\mathbb{N}$) f
⟨*proof*⟩

lemma *pbinopE*[*elim*]:
assumes *pbinop* A f
obtains *pnop* A ($2_\mathbb{N}$) f
⟨*proof*⟩

Elementary properties.

lemma (**in** *pbinop*) *fbinop-vcard*:
assumes $x \in_\circ \mathcal{D}_\circ\ f$
shows *vcard* $x = 2_\mathbb{N}$
⟨*proof*⟩

2.12.12 Total binary operation

locale *binop* = *nop* A ‹$2_\mathbb{N}$› f **for** A f

sublocale *binop* ⊆ *pbinop* ⟨*proof*⟩

Rules.

lemmas *binopI*[*intro*] = *binop.intro*

lemma *binopD*[*dest*]:
assumes *binop* A f
shows *nop* A ($2_\mathbb{N}$) f
⟨*proof*⟩

lemma *binopE*[*elim*]:
assumes *binop* A f
obtains *nop* A ($2_\mathbb{N}$) f
⟨*proof*⟩

Elementary properties.

lemma *binop-eqI*:
assumes *binop* A g
and *binop* A f
and $\bigwedge a\ b.\ [\![\ a \in_\circ A;\ b \in_\circ A\ ]\!] \Longrightarrow g(\!|a,\ b|\!)_\bullet = f(\!|a,\ b|\!)_\bullet$

shows $g = f$
⟨*proof*⟩

lemma (**in** *binop*) *binop-app-in-vrange*[*intro*]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$
shows $f(\!|a, b|\!)_\bullet \in_\circ \mathcal{R}_\circ f$
⟨*proof*⟩

2.12.13 Injective binary operation

locale *binop-v*11 = *nop-v*11 A ‹$2_\mathbb{N}$› f **for** A f

sublocale *binop-v*11 ⊆ *binop* A f ⟨*proof*⟩

Rules.

lemma *binop-v*11*I*[*intro*]:
assumes *nop-v*11 A ($2_\mathbb{N}$) f
shows *binop-v*11 A f
⟨*proof*⟩

lemma *binop-v*11*D*[*dest*]:
assumes *binop-v*11 A f
shows *nop-v*11 A ($2_\mathbb{N}$) f
⟨*proof*⟩

lemma *binop-v*11*E*[*elim*]:
assumes *binop-v*11 A f
obtains *nop-v*11 A ($2_\mathbb{N}$) f
⟨*proof*⟩

2.12.14 Surjective binary operation

locale *binop-onto* = *nop-onto* A ‹$2_\mathbb{N}$› f **for** A f

sublocale *binop-onto* ⊆ *binop* A f ⟨*proof*⟩

Rules.

lemma *binop-ontoI*[*intro*]:
assumes *nop-onto* A ($2_\mathbb{N}$) f
shows *binop-onto* A f
⟨*proof*⟩

lemma *binop-ontoD*[*dest*]:
assumes *binop-onto* A f
shows *nop-onto* A ($2_\mathbb{N}$) f
⟨*proof*⟩

lemma *binop-ontoE*[*elim*]:
assumes *binop-onto* A f
obtains *nop-onto* A ($2_\mathbb{N}$) f
⟨*proof*⟩

lemma *binop-ontoI′*[*intro*]:
assumes *binop* A f **and** $A \subseteq_\circ \mathcal{R}_\circ f$
shows *binop-onto* A f
⟨*proof*⟩

2.12.15 Bijective binary operation

locale *binop-bij* = *nop-bij* A ‹$2_{\mathbb{N}}$› f **for** A f

sublocale *binop-bij* ⊆ *binop-v11* A f
⟨*proof*⟩

sublocale *binop-bij* ⊆ *binop-onto* A f
⟨*proof*⟩

Rules.

lemma *binop-bijI*[*intro*]:
assumes *nop-bij* A ($2_{\mathbb{N}}$) f
shows *binop-bij* A f
⟨*proof*⟩

lemma *binop-bijD*[*dest*]:
assumes *binop-bij* A f
shows *nop-bij* A ($2_{\mathbb{N}}$) f
⟨*proof*⟩

lemma *binop-bijE*[*elim*]:
assumes *binop-bij* A f
obtains *nop-bij* A ($2_{\mathbb{N}}$) f
⟨*proof*⟩

lemma *binop-bijI'*[*intro*]:
assumes *binop-v11* A f **and** $A \subseteq_\circ \mathcal{R}_\circ\ f$
shows *binop-bij* A f
⟨*proof*⟩

2.12.16 Flip

definition *fflip* :: $V \Rightarrow V$
where *fflip* $f = (\lambda ab{\in_\circ}(\mathcal{D}_\circ\ f)^{-1}_\bullet.\ f(\!|ab(\!|1_{\mathbb{N}}|\!),\ ab(\!|0|\!)|\!)_\bullet)$

Elementary properties.

lemma *fflip-vempty*[*simp*]: *fflip* 0 = 0 ⟨*proof*⟩

lemma *fflip-vsv*: *vsv* (*fflip* f)
⟨*proof*⟩

lemma *vdomain-fflip*[*simp*]: $\mathcal{D}_\circ$ (*fflip* f) = $(\mathcal{D}_\circ\ f)^{-1}_\bullet$
⟨*proof*⟩

lemma (**in** *pbinop*) *vrange-fflip*: $\mathcal{R}_\circ$ (*fflip* f) = $\mathcal{R}_\circ\ f$
⟨*proof*⟩

lemma *fflip-app*[*simp*]:
assumes $[a,\ b]_\circ \in_\circ \mathcal{D}_\circ\ f$
shows *fflip* $f(\!|b,\ a|\!)_\bullet = f(\!|a,\ b|\!)_\bullet$
⟨*proof*⟩

lemma (**in** *pbinop*) *pbinop-fflip-fflip*: *fflip* (*fflip* f) = f
⟨*proof*⟩

lemma (**in** *binop*) *pbinop-fflip-app*[*simp*]:
assumes $a \in_\circ A$ **and** $b \in_\circ A$

shows *fflip* $f(\!|b,\ a|\!)_\bullet = f(\!|a,\ b|\!)_\bullet$
⟨*proof*⟩

lemma *fflip-vsingleton*: *fflip* (*set* $\{\langle [a,\ b]_\circ,\ c\rangle\}$) = *set* $\{\langle [b,\ a]_\circ,\ c\rangle\}$
⟨*proof*⟩

2.13 Construction of integer numbers, rational numbers and real numbers

2.13.1 Background

The set of real numbers $\mathbb{R}_\circ$ is defined in a way such that it agrees with the set of natural numbers ω. However, otherwise, real numbers are allowed to be arbitrary sets in *Vset* $(\omega + \omega)$.[2] Integer and rational numbers are exposed via canonical injections into the set of real numbers from the types *int* and *rat*, respectively. Lastly, common operations on the real, integer and rational numbers are defined and some of their main properties are exposed.

The primary reference for this section is the textbook *The Real Numbers and Real Analysis* by E. Bloch [14]. Nonetheless, it is not claimed that the exposition of the subject presented in this section is entirely congruent with the exposition in the aforementioned reference.

declare *One-nat-def*[*simp del*]

named-theorems *vnumber-simps*

lemmas [*vnumber-simps*] =
Collect-mem-eq Ball-def[*symmetric*] *Bex-def*[*symmetric*] *vsubset-eq*[*symmetric*]

Supplementary material for the evaluation of the upper bound of the cardinality of the continuum.

lemma *inj-image-ord-of-nat*: *inj* (*image ord-of-nat*)
⟨*proof*⟩

lemma *vlepoll-VPow-omega-if-vreal-lepoll-real*:
assumes $x \lesssim$ (*UNIV*::*real set*)
shows *set* $x \lesssim_\circ$ *VPow* ω
⟨*proof*⟩

2.13.2 Real numbers

Definition

abbreviation *real* :: *nat* ⇒ *real*
where *real* ≡ *of-nat*

definition *nat-of-real* :: *real* ⇒ *nat*
where *nat-of-real* = *inv-into UNIV real*

definition *vreal-of-real-impl* :: *real* ⇒ *V*
where *vreal-of-real-impl* = (*SOME V-of*::*real*⇒*V*. *inj V-of*)

lemma *inj-vreal-of-real-impl*: *inj vreal-of-real-impl*
⟨*proof*⟩

lemma *inj-on-inv-vreal-of-real-impl*:
inj-on (*inv vreal-of-real-impl*) (*range vreal-of-real-impl*)
⟨*proof*⟩

lemma *range-vreal-of-real-impl-vlepoll-VPow-omega*:
set (*range vreal-of-real-impl*) $\lesssim_\circ$ *VPow* ω
⟨*proof*⟩

definition *vreal-impl* :: *V*

[2] The idea itself is not new, e.g., see [26].

where *vreal-impl* =
(
SOME y.
range vreal-of-real-impl ≈ *elts y* ∧
vdisjnt y ω ∧
y ∈∘ *Vset* (*ω* + *ω*)
)

lemma *vreal-impl-eqpoll*: *range vreal-of-real-impl* ≈ *elts vreal-impl*
and *vreal-impl-vdisjnt*: *vdisjnt vreal-impl ω*
and *vreal-impl-in-Vset-ss-omega*: *vreal-impl* ∈∘ *Vset* (*ω* + *ω*)
⟨*proof*⟩

definition *vreal-of-real-impl′* :: *V* ⇒ *V*
where *vreal-of-real-impl′* =
(*SOME f. bij-betw f* (*range vreal-of-real-impl*) (*elts vreal-impl*))

lemma *vreal-of-real-impl′-bij-betw*:
bij-betw vreal-of-real-impl′ (*range vreal-of-real-impl*) (*elts vreal-impl*)
⟨*proof*⟩

definition *vreal-of-real-impl″* :: *real* ⇒ *V*
where *vreal-of-real-impl″* = *vreal-of-real-impl′* ∘ *vreal-of-real-impl*

lemma *vreal-of-real-impl″*: *disjnt* (*range vreal-of-real-impl″*) (*elts ω*)
⟨*proof*⟩

lemma *inj-vreal-of-real-impl″*: *inj vreal-of-real-impl″*
⟨*proof*⟩

Main definitions.

definition *vreal-of-real* :: *real* ⇒ *V*
where *vreal-of-real x* =
(*if x* ∈ ℕ *then* (*nat-of-real x*)ℕ *else vreal-of-real-impl″ x*)

notation *vreal-of-real* (‹-ℝ› [1000] 999)

declare [[*coercion vreal-of-real* :: *real* ⇒ *V*]]

definition *vreal* :: *V* (‹ℝ∘›)
where *vreal* = *set* (*range vreal-of-real*)

definition *real-of-vreal* :: *V* ⇒ *real*
where *real-of-vreal* = *inv-into UNIV vreal-of-real*

Rules.

lemma *vreal-of-real-in-vrealI*[*intro*, *simp*]: *a*ℝ ∈∘ ℝ∘
⟨*proof*⟩

lemma *vreal-of-real-in-vrealE*[*elim*]:
assumes *a* ∈∘ ℝ∘
obtains *b* **where** *b*ℝ = *a*
⟨*proof*⟩

Elementary properties.

lemma *vnat-eq-vreal*: *x*ℕ = *x*ℝ ⟨*proof*⟩

lemma *omega-vsubset-vreal*: $\omega \subseteq_\circ \mathbb{R}_\circ$
⟨*proof*⟩

lemma *inj-vreal-of-real*: *inj vreal-of-real*
⟨*proof*⟩

lemma *vreal-in-Vset-ω2*: $\mathbb{R}_\circ \in_\circ$ *Vset* $(\omega + \omega)$
⟨*proof*⟩

lemma *real-of-vreal-vreal-of-real*[*simp*]: *real-of-vreal* $(a_{\mathbb{R}})$ = *a*
⟨*proof*⟩

Transfer rules

definition *cr-vreal* :: *V* ⇒ *real* ⇒ *bool*
where *cr-vreal a b* ⟷ (*a* = *vreal-of-real b*)

lemma *cr-vreal-right-total*[*transfer-rule*]: *right-total cr-vreal*
⟨*proof*⟩

lemma *cr-vreal-bi-uniqie*[*transfer-rule*]: *bi-unique cr-vreal*
⟨*proof*⟩

lemma *cr-vreal-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-vreal = (λ*x*. *x* $\in_\circ \mathbb{R}_\circ$)
⟨*proof*⟩

lemma *vreal-transfer*[*transfer-rule*]:
(*rel-set cr-vreal*) (*elts* $\mathbb{R}_\circ$) (*UNIV*::*real set*)
⟨*proof*⟩

lemma *vreal-of-real-transfer*[*transfer-rule*]: *cr-vreal* (*vreal-of-real a*) *a*
⟨*proof*⟩

Constants and operations

Auxiliary.

lemma *vreal-fsingleton-in-fproduct-vreal*: $[a_{\mathbb{R}}]_\circ \in_\circ \mathbb{R}_\circ \mathbin{\widehat{}}_\times 1_{\mathbb{N}}$ ⟨*proof*⟩

lemma *vreal-fpair-in-fproduct-vreal*: $[a_{\mathbb{R}}, b_{\mathbb{R}}]_\circ \in_\circ \mathbb{R}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}$ ⟨*proof*⟩

Zero.

lemma *vreal-zero*: $0_{\mathbb{R}}$ = (0::*V*)
⟨*proof*⟩

One.

lemma *vreal-one*: $1_{\mathbb{R}}$ = (1::*V*)
⟨*proof*⟩

Addition.

definition *vreal-plus* :: *V*
where *vreal-plus* =
(λ*x*$\in_\circ \mathbb{R}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}$. (*real-of-vreal* (*x*($|0_{\mathbb{N}}|$)) + *real-of-vreal* (*x*($|1_{\mathbb{N}}|$)))$_{\mathbb{R}}$)

abbreviation *vreal-plus-app* :: *V* ⇒ *V* ⇒ *V* (**infixl** ‹+$_{\mathbb{R}}$› 65)
where *vreal-plus-app a b* ≡ *vreal-plus*($|a, b|$)$_\bullet$
notation *vreal-plus-app* (**infixl** ‹+$_{\mathbb{R}}$› 65)

lemma *vreal-plus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal* ===> *cr-vreal*)
(+$_{\mathbb{R}}$) (+)
⟨*proof*⟩

Multiplication.

definition *vreal-mult* :: *V*
where *vreal-mult* =
(λ*x*∈$_\circ$$\mathbb{R}_\circ$ ^$_\times$ $2_{\mathbb{N}}$. (*real-of-vreal* (*x*(|$0_{\mathbb{N}}$|)) ∗ *real-of-vreal* (*x*(|$1_{\mathbb{N}}$|)))$_{\mathbb{R}}$)

abbreviation *vreal-mult-app* (**infixl** ‹∗$_{\mathbb{R}}$› 70)
where *vreal-mult-app a b* ≡ *vreal-mult*(|*a*, *b*|)•
notation *vreal-mult-app* (**infixl** ‹∗$_{\mathbb{R}}$› 70)

lemma *vreal-mult-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal* ===> *cr-vreal*) (∗$_{\mathbb{R}}$) (∗)
⟨*proof*⟩

Unary minus.

definition *vreal-uminus* :: *V*
where *vreal-uminus* = (λ*x*∈$_\circ$$\mathbb{R}_\circ$. (*uminus* (*real-of-vreal x*))$_{\mathbb{R}}$)

abbreviation *vreal-uminus-app* (‹−$_{\mathbb{R}}$ -› [81] 80)
where −$_{\mathbb{R}}$ *a* ≡ *vreal-uminus*(|*a*|)

lemma *vreal-uminus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal*) (*vreal-uminus-app*) (*uminus*)
⟨*proof*⟩

Multiplicative inverse.

definition *vreal-inverse* :: *V*
where *vreal-inverse* = (λ*x*∈$_\circ$$\mathbb{R}_\circ$. (*inverse* (*real-of-vreal x*))$_{\mathbb{R}}$)

abbreviation *vreal-inverse-app* (‹(-$^{-1}{}_{\mathbb{R}}$)› [1000] 999)
where $a^{-1}{}_{\mathbb{R}}$ ≡ *vreal-inverse*(|*a*|)

lemma *vreal-inverse-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal*) (*vreal-inverse-app*) (*inverse*)
⟨*proof*⟩

Order.

definition *vreal-le* :: *V*
where *vreal-le* =
set {[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ ∈$_\circ$ $\mathbb{R}_\circ$ ^$_\times$ $2_{\mathbb{N}}$ ∧ *real-of-vreal a* ≤ *real-of-vreal b*}

abbreviation *vreal-le′* (‹(-/ ≤$_{\mathbb{R}}$ -)› [51, 51] 50)
where *a* ≤$_{\mathbb{R}}$ *b* ≡ [*a*, *b*]$_\circ$ ∈$_\circ$ *vreal-le*

lemma *small-vreal-le*[*simp*]:
small
{[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ ∈$_\circ$ $\mathbb{R}_\circ$ ^$_\times$ $2_{\mathbb{N}}$ ∧ *real-of-vreal a* ≤ *real-of-vreal b*}
⟨*proof*⟩

lemma *vreal-le-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal* ===> (=)) *vreal-le′* (≤)
⟨*proof*⟩

Strict order.

definition *vreal-ls* :: *V*
where *vreal-ls* =
set {[*a*, *b*]∘ | *a b*. [*a*, *b*]∘ ∈∘ $\mathbb{R}_\circ$ ^× $2_\mathbb{N}$ ∧ *real-of-vreal a* < *real-of-vreal b*}

abbreviation *vreal-ls′* (‹(-/ <$_\mathbb{R}$ -)› [51, 51] 50)
where *a* <$_\mathbb{R}$ *b* ≡ [*a*, *b*]∘ ∈∘ *vreal-ls*

lemma *small-vreal-ls*[*simp*]:
small
{[*a*, *b*]∘ | *a b*. [*a*, *b*]∘ ∈∘ $\mathbb{R}_\circ$ ^× $2_\mathbb{N}$ ∧ *real-of-vreal a* < *real-of-vreal b*}
⟨*proof*⟩

lemma *vreal-ls-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal* ===> (=)) *vreal-ls′* (<)
⟨*proof*⟩

Subtraction.

definition *vreal-minus* :: *V*
where *vreal-minus* =
(λ*x*∈∘$\mathbb{R}_\circ$ ^× $2_\mathbb{N}$. (*real-of-vreal* (*x*(|$0_\mathbb{N}$|)) − *real-of-vreal* (*x*(|$1_\mathbb{N}$|)))$_\mathbb{R}$)

abbreviation *vreal-minus-app* (**infixl** ‹−$_\mathbb{R}$› 65)
where *vreal-minus-app a b* ≡ *vreal-minus*(|*a*, *b*|)•

lemma *vreal-minus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vreal* ===> *cr-vreal* ===> *cr-vreal*) (−$_\mathbb{R}$) (−)
⟨*proof*⟩

Axioms of an ordered field with the least upper bound property.

The exposition follows the Definitions 2.2.1 and 2.2.3 from the textbook *The Real Numbers and Real Analysis* by E. Bloch [14].

lemma *vreal-zero-closed*: $0_\mathbb{R}$ ∈∘ $\mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-one-closed*: $1_\mathbb{R}$ ∈∘ $\mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-plus-closed*:
assumes *x* ∈∘ $\mathbb{R}_\circ$ **and** *y* ∈∘ $\mathbb{R}_\circ$
shows *x* +$_\mathbb{R}$ *y* ∈∘ $\mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-uminus-closed*:
assumes *x* ∈∘ $\mathbb{R}_\circ$
shows −$_\mathbb{R}$ *x* ∈∘ $\mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-mult-closed*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$
shows $x *_{\mathbb{R}} y \in_\circ \mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-inverse-closed*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows ${x^{-1}}_{\mathbb{R}} \in_\circ \mathbb{R}_\circ$
⟨*proof*⟩

Associative Law for Addition: Definition 2.2.1.a.

lemma *vreal-assoc-law-addition*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$ **and** $z \in_\circ \mathbb{R}_\circ$
shows $(x +_{\mathbb{R}} y) +_{\mathbb{R}} z = x +_{\mathbb{R}} (y +_{\mathbb{R}} z)$
⟨*proof*⟩

Commutative Law for Addition: Definition 2.2.1.b.

lemma *vreal-commutative-law-addition*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$
shows $x +_{\mathbb{R}} y = y +_{\mathbb{R}} x$
⟨*proof*⟩

Identity Law for Addition: Definition 2.2.1.c.

lemma *vreal-identity-law-addition*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows $x +_{\mathbb{R}} 0_{\mathbb{R}} = x$
⟨*proof*⟩

Inverses Law for Addition: Definition 2.2.1.d.

lemma *vreal-inverses-law-addition*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows $x +_{\mathbb{R}} (-_{\mathbb{R}} x) = 0_{\mathbb{R}}$
⟨*proof*⟩

Associative Law for Multiplication: Definition 2.2.1.e.

lemma *vreal-assoc-law-multiplication*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$ **and** $z \in_\circ \mathbb{R}_\circ$
shows $(x *_{\mathbb{R}} y) *_{\mathbb{R}} z = x *_{\mathbb{R}} (y *_{\mathbb{R}} z)$
⟨*proof*⟩

Commutative Law for Multiplication: Definition 2.2.1.f.

lemma *vreal-commutative-law-multiplication*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$
shows $x *_{\mathbb{R}} y = y *_{\mathbb{R}} x$
⟨*proof*⟩

Identity Law for Multiplication: Definition 2.2.1.g.

lemma *vreal-identity-law-multiplication*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows $x *_{\mathbb{R}} 1_{\mathbb{R}} = x$
⟨*proof*⟩

Inverses Law for Multiplication: Definition 2.2.1.h.

lemma *vreal-inverses-law-multiplication*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $x \neq 0_{\mathbb{R}}$
shows $x *_{\mathbb{R}} {x^{-1}}_{\mathbb{R}} = 1_{\mathbb{R}}$

⟨*proof*⟩

Distributive Law: Definition 2.2.1.i.

lemma *vreal-distributive-law*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$ **and** $z \in_\circ \mathbb{R}_\circ$
shows $x *_{\mathbb{R}} (y +_{\mathbb{R}} z) = x *_{\mathbb{R}} y +_{\mathbb{R}} x *_{\mathbb{R}} z$
⟨*proof*⟩

Trichotomy Law: Definition 2.2.1.j.

lemma *vreal-trichotomy-law*:
assumes $x \in_\circ \mathbb{R}_\circ$ $y \in_\circ \mathbb{R}_\circ$
shows
$(x <_{\mathbb{R}} y \wedge \sim(x = y) \wedge \sim(y <_{\mathbb{R}} x)) \vee$
$(\sim(x <_{\mathbb{R}} y) \wedge x = y \wedge \sim(y <_{\mathbb{R}} x)) \vee$
$(\sim(x <_{\mathbb{R}} y) \wedge \sim(x = y) \wedge y <_{\mathbb{R}} x)$
⟨*proof*⟩

Transitive Law: Definition 2.2.1.k.

lemma *vreal-transitive-law*:
assumes $x \in_\circ \mathbb{R}_\circ$
and $y \in_\circ \mathbb{R}_\circ$
and $z \in_\circ \mathbb{R}_\circ$
and $x <_{\mathbb{R}} y$ **and** $y <_{\mathbb{R}} z$
shows $x <_{\mathbb{R}} z$
⟨*proof*⟩

Addition Law of Order: Definition 2.2.1.l.

lemma *vreal-addition-law-of-order*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$ **and** $z \in_\circ \mathbb{R}_\circ$ **and** $x <_{\mathbb{R}} y$
shows $x +_{\mathbb{R}} z <_{\mathbb{R}} y +_{\mathbb{R}} z$
⟨*proof*⟩

Multiplication Law of Order: Definition 2.2.1.m.

lemma *vreal-multiplication-law-of-order*:
assumes $x \in_\circ \mathbb{R}_\circ$
and $y \in_\circ \mathbb{R}_\circ$
and $z \in_\circ \mathbb{R}_\circ$
and $x <_{\mathbb{R}} y$
and $0_{\mathbb{R}} <_{\mathbb{R}} z$
shows $x *_{\mathbb{R}} z <_{\mathbb{R}} y *_{\mathbb{R}} z$
⟨*proof*⟩

Non-Triviality: Definition 2.2.1.n.

lemma *vreal-non-triviality*: $0_{\mathbb{R}} \neq 1_{\mathbb{R}}$
⟨*proof*⟩

Least upper bound property: Definition 2.2.3.

lemma *least-upper-bound-property*:
defines *vreal-ub* $S\ M \equiv (S \subseteq_\circ \mathbb{R}_\circ \wedge M \in_\circ \mathbb{R}_\circ \wedge (\forall\, x \in_\circ S.\ x \leq_{\mathbb{R}} M))$
assumes $A \subseteq_\circ \mathbb{R}_\circ$ **and** $A \neq 0$ **and** $\exists\, M.$ *vreal-ub* $A\ M$
obtains M **where** *vreal-ub* $A\ M$ **and** $\bigwedge T.$ *vreal-ub* $A\ T \Longrightarrow M \leq_{\mathbb{R}} T$
⟨*proof*⟩

Fundamental properties of other operations

Minus.

lemma *vreal-minus-closed*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$
shows $x -_{\mathbb{R}} y \in_\circ \mathbb{R}_\circ$
⟨*proof*⟩

lemma *vreal-minus-eq-plus-uminus*:
assumes $x \in_\circ \mathbb{R}_\circ$ **and** $y \in_\circ \mathbb{R}_\circ$
shows $x -_{\mathbb{R}} y = x +_{\mathbb{R}} (-_{\mathbb{R}} y)$
⟨*proof*⟩

Unary minus.

lemma *vreal-uminus-uminus*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows $x = -_{\mathbb{R}} (-_{\mathbb{R}} x)$
⟨*proof*⟩

Multiplicative inverse.

lemma *vreal-inverse-inverse*:
assumes $x \in_\circ \mathbb{R}_\circ$
shows $x = (x^{-1}{}_{\mathbb{R}})^{-1}{}_{\mathbb{R}}$
⟨*proof*⟩

Further properties

Addition.

global-interpretation *vreal-plus*: *binop-onto* ‹$\mathbb{R}_\circ$› *vreal-plus*
⟨*proof*⟩

Unary minus.

global-interpretation *vreal-uminus*: *v11* *vreal-uminus*
rewrites *vreal-uminus-vdomain*[*simp*]: $\mathcal{D}_\circ$ *vreal-uminus* = $\mathbb{R}_\circ$
and *vreal-uminus-vrange*[*simp*]: $\mathcal{R}_\circ$ *vreal-uminus* = $\mathbb{R}_\circ$
⟨*proof*⟩

Multiplication.

global-interpretation *vreal-mult*: *binop-onto* ‹$\mathbb{R}_\circ$› *vreal-mult*
⟨*proof*⟩

Multiplicative inverse.

global-interpretation *vreal-inverse*: *v11* *vreal-inverse*
rewrites *vreal-inverse-vdomain*[*simp*]: $\mathcal{D}_\circ$ *vreal-inverse* = $\mathbb{R}_\circ$
and *vreal-inverse-vrange*[*simp*]: $\mathcal{R}_\circ$ *vreal-inverse* = $\mathbb{R}_\circ$
⟨*proof*⟩

2.13.3 Integer numbers

Definition

definition *vint-of-int* :: *int* ⇒ *V*
where *vint-of-int* = *vreal-of-real*

notation *vint-of-int* (‹-$_{\mathbb{Z}}$› [999] 999)

declare [[*coercion vint-of-int* :: *int* ⇒ *V*]]

definition *vint* :: *V* (‹$\mathbb{Z}_\circ$›)
where *vint* = *set* (*range vint-of-int*)

definition *int-of-vint* :: $V \Rightarrow$ *int*
where *int-of-vint* = *inv-into UNIV vint-of-int*

Rules.

lemma *vint-of-int-in-vintI*[*intro*, *simp*]: $a_{\mathbb{Z}} \in_{\circ} \mathbb{Z}_{\circ}$ ⟨*proof*⟩

lemma *vint-of-int-in-vintE*[*elim*]:
assumes $a \in_{\circ} \mathbb{Z}_{\circ}$
obtains b **where** $b_{\mathbb{Z}} = a$
⟨*proof*⟩

Elementary properties

lemma *vint-vsubset-vreal*: $\mathbb{Z}_{\circ} \subseteq_{\circ} \mathbb{R}_{\circ}$
⟨*proof*⟩

lemma *inj-vint-of-int*: *inj vint-of-int*
⟨*proof*⟩

lemma *vint-in-Vset-ω2*: $\mathbb{Z}_{\circ} \in_{\circ} Vset\ (\omega + \omega)$
⟨*proof*⟩

lemma *int-of-vint-vint-of-int*[*simp*]: *int-of-vint* $(a_{\mathbb{Z}}) = a$
⟨*proof*⟩

Transfer rules.

definition *cr-vint* :: $V \Rightarrow$ *int* $\Rightarrow$ *bool*
where *cr-vint a b* $\longleftrightarrow$ $(a =$ *vint-of-int b*$)$

lemma *cr-vint-right-total*[*transfer-rule*]: *right-total cr-vint*
⟨*proof*⟩

lemma *cr-vint-bi-unqie*[*transfer-rule*]: *bi-unique cr-vint*
⟨*proof*⟩

lemma *cr-vint-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-vint = $(\lambda x.\ x \in_{\circ} \mathbb{Z}_{\circ})$
⟨*proof*⟩

lemma *vint-transfer*[*transfer-rule*]:
(*rel-set cr-vint*) (*elts* $\mathbb{Z}_{\circ}$) (*UNIV*::*int set*)
⟨*proof*⟩

lemma *vint-of-int-transfer*[*transfer-rule*]: *cr-vint* (*vint-of-int a*) *a*
⟨*proof*⟩

Constants and operations

Auxiliary.

lemma *vint-fsingleton-in-fproduct-vint*: $[a_{\mathbb{Z}}]_{\circ} \in_{\circ} \mathbb{Z}_{\circ} \mathbin{\widehat{}}_{\times} 1_{\mathbb{N}}$ ⟨*proof*⟩

lemma *vint-fpair-in-fproduct-vint*: $[a_{\mathbb{Z}}, b_{\mathbb{Z}}]_{\circ} \in_{\circ} \mathbb{Z}_{\circ} \mathbin{\widehat{}}_{\times} 2_{\mathbb{N}}$ ⟨*proof*⟩

Zero.

lemma *vint-zero*: $0_{\mathbb{Z}} = (0 :: V)$ ⟨*proof*⟩

One.

lemma *vint-one*: $1_{\mathbb{Z}} = (1::V)$ ⟨*proof*⟩

Addition.

definition *vint-plus* :: *V*
  where *vint-plus* =
    $(\lambda x \in_\circ \mathbb{Z}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}.\ (\mathit{int\text{-}of\text{-}vint}\ (x(\!|0_{\mathbb{N}}|\!)) + \mathit{int\text{-}of\text{-}vint}\ (x(\!|1_{\mathbb{N}}|\!)))_{\mathbb{Z}})$

abbreviation *vint-plus-app* (**infixl** ‹+$_{\mathbb{Z}}$› 65)
  where *vint-plus-app a b* ≡ *vint-plus*(|*a*, *b*|)$_\bullet$

lemma *vint-plus-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint* ===> *cr-vint*) (+$_{\mathbb{Z}}$) (+)
  ⟨*proof*⟩

Multiplication.

definition *vint-mult* :: *V*
  where *vint-mult* =
    $(\lambda x \in_\circ \mathbb{Z}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}.\ (\mathit{int\text{-}of\text{-}vint}\ (x(\!|0_{\mathbb{N}}|\!)) * \mathit{int\text{-}of\text{-}vint}\ (x(\!|1_{\mathbb{N}}|\!)))_{\mathbb{Z}})$

abbreviation *vint-mult-app* (**infixl** ‹*$_{\mathbb{Z}}$› 65)
  where *vint-mult-app a b* ≡ *vint-mult*(|*a*, *b*|)$_\bullet$

lemma *vint-mult-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint* ===> *cr-vint*) (*$_{\mathbb{Z}}$) (*)
  ⟨*proof*⟩

Unary minus.

definition *vint-uminus* :: *V*
  where *vint-uminus* = $(\lambda x \in_\circ \mathbb{Z}_\circ.\ (\mathit{uminus}\ (\mathit{int\text{-}of\text{-}vint}\ x))_{\mathbb{Z}})$

abbreviation *vint-uminus-app* (‹−$_{\mathbb{Z}}$ -› [81] 80)
  where −$_{\mathbb{Z}}$ *a* ≡ *vint-uminus*(|*a*|)

lemma *vint-uminus-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint*) (*vint-uminus-app*) (*uminus*)
  ⟨*proof*⟩

Order.

definition *vint-le* :: *V*
  where *vint-le* =
    set $\{[a, b]_\circ \mid a\ b.\ [a, b]_\circ \in_\circ \mathbb{Z}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}} \wedge \mathit{int\text{-}of\text{-}vint}\ a \leq \mathit{int\text{-}of\text{-}vint}\ b\}$

abbreviation *vint-le′* (‹(-/ ≤$_{\mathbb{Z}}$ -)› [51, 51] 50)
  where *a* ≤$_{\mathbb{Z}}$ *b* ≡ $[a, b]_\circ \in_\circ$ *vint-le*

lemma *small-vint-le*[*simp*]:
  small $\{[a, b]_\circ \mid a\ b.\ [a, b]_\circ \in_\circ \mathbb{Z}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}} \wedge \mathit{int\text{-}of\text{-}vint}\ a \leq \mathit{int\text{-}of\text{-}vint}\ b\}$
⟨*proof*⟩

lemma *vint-le-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint* ===> (=)) *vint-le′* (≤)
  ⟨*proof*⟩

Strict order.

definition *vint-ls* :: *V*
  where *vint-ls* =
    set {$[a,\ b]_\circ$ | *a b*. $[a,\ b]_\circ \in_\circ \mathbb{Z}_\circ \mathbin{\hat{}}_\times 2_\mathbb{N} \wedge$ *int-of-vint a* < *int-of-vint b*}

abbreviation *vint-ls′* (‹(-/ $<_\mathbb{Z}$ -)› [51, 51] 50)
  where $a <_\mathbb{Z} b \equiv [a,\ b]_\circ \in_\circ$ *vint-ls*

lemma *small-vint-ls*[*simp*]:
  small {$[a,\ b]_\circ$ | *a b*. $[a,\ b]_\circ \in_\circ \mathbb{Z}_\circ \mathbin{\hat{}}_\times 2_\mathbb{N} \wedge$ *int-of-vint a* < *int-of-vint b*}
⟨*proof*⟩

lemma *vint-ls-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint* ===> (=)) *vint-ls′* (<)
  ⟨*proof*⟩

Subtraction.

definition *vint-minus* :: *V*
  where *vint-minus* =
    ($\lambda x \in_\circ \mathbb{Z}_\circ \mathbin{\hat{}}_\times 2_\mathbb{N}$. (*int-of-vint* ($x(\!|0_\mathbb{N}|\!)$) − *int-of-vint* ($x(\!|1_\mathbb{N}|\!)$))$_\mathbb{Z}$)

abbreviation *vint-minus-app* (**infixl** ‹$-_\mathbb{Z}$› 65)
  where *vint-minus-app a b* ≡ *vint-minus*$(\!|a,\ b|\!)_\bullet$

lemma *vint-minus-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vint* ===> *cr-vint* ===> *cr-vint*) ($-_\mathbb{Z}$) (−)
  ⟨*proof*⟩

Axioms of a well ordered integral domain

The exposition follows Definition 1.4.1 from the textbook *The Real Numbers and Real Analysis* by E. Bloch [14].

lemma *vint-zero-closed*: $0_\mathbb{Z} \in_\circ \mathbb{Z}_\circ$ ⟨*proof*⟩

lemma *vint-one-closed*: $1_\mathbb{Z} \in_\circ \mathbb{Z}_\circ$ ⟨*proof*⟩

lemma *vint-plus-closed*:
  assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
  shows $x +_\mathbb{Z} y \in_\circ \mathbb{Z}_\circ$
⟨*proof*⟩

lemma *vint-mult-closed*:
  assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
  shows $x *_\mathbb{Z} y \in_\circ \mathbb{Z}_\circ$
⟨*proof*⟩

lemma *vint-uminus-closed*:
  assumes $x \in_\circ \mathbb{Z}_\circ$
  shows $-_\mathbb{Z} x \in_\circ \mathbb{Z}_\circ$
⟨*proof*⟩

Associative Law for Addition: Definition 1.4.1.a.

lemma *vint-assoc-law-addition*:
  assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$ **and** $z \in_\circ \mathbb{Z}_\circ$

shows $(x +_{\mathbb{Z}} y) +_{\mathbb{Z}} z = x +_{\mathbb{Z}} (y +_{\mathbb{Z}} z)$
⟨*proof*⟩

Commutative Law for Addition: Definition 1.4.1.b.

lemma *vint-commutative-law-addition*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
shows $x +_{\mathbb{Z}} y = y +_{\mathbb{Z}} x$
⟨*proof*⟩

Identity Law for Addition: Definition 1.4.1.c.

lemma *vint-identity-law-addition*:
assumes [*simp*]: $x \in_\circ \mathbb{Z}_\circ$
shows $x +_{\mathbb{Z}} 0_{\mathbb{Z}} = x$
⟨*proof*⟩

Inverses Law for Addition: Definition 1.4.1.d.

lemma *vint-inverses-law-addition*:
assumes [*simp*]: $x \in_\circ \mathbb{Z}_\circ$
shows $x +_{\mathbb{Z}} (-_{\mathbb{Z}} x) = 0_{\mathbb{Z}}$
⟨*proof*⟩

Associative Law for Multiplication: Definition 1.4.1.e.

lemma *vint-assoc-law-multiplication*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$ **and** $z \in_\circ \mathbb{Z}_\circ$
shows $(x *_{\mathbb{Z}} y) *_{\mathbb{Z}} z = x *_{\mathbb{Z}} (y *_{\mathbb{Z}} z)$
⟨*proof*⟩

Commutative Law for Multiplication: Definition 1.4.1.f.

lemma *vint-commutative-law-multiplication*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
shows $x *_{\mathbb{Z}} y = y *_{\mathbb{Z}} x$
⟨*proof*⟩

Identity Law for multiplication: Definition 1.4.1.g.

lemma *vint-identity-law-multiplication*:
assumes $x \in_\circ \mathbb{Z}_\circ$
shows $x *_{\mathbb{Z}} 1_{\mathbb{Z}} = x$
⟨*proof*⟩

Distributive Law for Multiplication: Definition 1.4.1.h.

lemma *vint-distributive-law*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$ **and** $z \in_\circ \mathbb{Z}_\circ$
shows $x *_{\mathbb{Z}} (y +_{\mathbb{Z}} z) = (x *_{\mathbb{Z}} y) +_{\mathbb{Z}} (x *_{\mathbb{Z}} z)$
⟨*proof*⟩

No Zero Divisors Law: Definition 1.4.1.i.

lemma *vint-no-zero-divisors-law*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$ **and** $x *_{\mathbb{Z}} y = 0_{\mathbb{Z}}$
shows $x = 0_{\mathbb{Z}} \vee y = 0_{\mathbb{Z}}$
⟨*proof*⟩

Trichotomy Law: Definition 1.4.1.j

lemma *vint-trichotomy-law*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
shows
$(x <_{\mathbb{Z}} y \wedge \sim(x = y) \wedge \sim(y <_{\mathbb{Z}} x)) \vee$

$(\tilde{}(x <_{\mathbb{Z}} y) \wedge x = y \wedge \tilde{}(y <_{\mathbb{Z}} x)) \vee$
$(\tilde{}(x <_{\mathbb{Z}} y) \wedge \tilde{}(x = y) \wedge y <_{\mathbb{Z}} x)$
⟨*proof*⟩

Transitive Law: Definition 1.4.1.k

lemma *vint-transitive-law*:
assumes $x \in_\circ \mathbb{Z}_\circ$
and $y \in_\circ \mathbb{Z}_\circ$
and $z \in_\circ \mathbb{Z}_\circ$
and $x <_{\mathbb{Z}} y$
and $y <_{\mathbb{Z}} z$
shows $x <_{\mathbb{Z}} z$
⟨*proof*⟩

Addition Law of Order: Definition 1.4.1.l

lemma *vint-addition-law-of-order*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$ **and** $z \in_\circ \mathbb{Z}_\circ$ **and** $x <_{\mathbb{Z}} y$
shows $x +_{\mathbb{Z}} z <_{\mathbb{Z}} y +_{\mathbb{Z}} z$
⟨*proof*⟩

Multiplication Law of Order: Definition 1.4.1.m

lemma *vint-multiplication-law-of-order*:
assumes $x \in_\circ \mathbb{Z}_\circ$
and $y \in_\circ \mathbb{Z}_\circ$
and $z \in_\circ \mathbb{Z}_\circ$
and $x <_{\mathbb{Z}} y$
and $0_{\mathbb{Z}} <_{\mathbb{Z}} z$
shows $x *_{\mathbb{Z}} z <_{\mathbb{Z}} y *_{\mathbb{Z}} z$
⟨*proof*⟩

Non-Triviality: Definition 1.4.1.n

lemma *vint-non-triviality*: $0_{\mathbb{Z}} \neq 1_{\mathbb{Z}}$
⟨*proof*⟩

Well-Ordering Principle.

lemma *well-ordering-principle*:
assumes $A \subseteq_\circ \mathbb{Z}_\circ$
and $a \in_\circ \mathbb{Z}_\circ$
and $A \neq 0$
and $\bigwedge x.\ x \in_\circ A \Longrightarrow a <_{\mathbb{Z}} x$
obtains b **where** $b \in_\circ A$ **and** $\bigwedge x.\ x \in_\circ A \Longrightarrow b \leq_{\mathbb{Z}} x$
⟨*proof*⟩

Fundamental properties of other operations

Minus.

lemma *vint-minus-closed*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
shows $x -_{\mathbb{Z}} y \in_\circ \mathbb{Z}_\circ$
⟨*proof*⟩

lemma *vint-minus-eq-plus-uminus*:
assumes $x \in_\circ \mathbb{Z}_\circ$ **and** $y \in_\circ \mathbb{Z}_\circ$
shows $x -_{\mathbb{Z}} y = x +_{\mathbb{Z}} (-_{\mathbb{Z}} y)$
⟨*proof*⟩

Unary minus.

lemma *vint-uminus-uminus*:
assumes $x \in_\circ \mathbb{Z}_\circ$
shows $x = -_\mathbb{Z}\ (-_\mathbb{Z}\ x)$
⟨*proof*⟩

Further properties

Addition.

global-interpretation *vint-plus*: *binop-onto* ‹$\mathbb{Z}_\circ$› *vint-plus*
⟨*proof*⟩

Unary minus.

global-interpretation *vint-uminus*: *v*11 *vint-uminus*
rewrites *vint-uminus-vdomain*[*simp*]: $\mathcal{D}_\circ$ *vint-uminus* = $\mathbb{Z}_\circ$
and *vint-uminus-vrange*[*simp*]: $\mathcal{R}_\circ$ *vint-uminus* = $\mathbb{Z}_\circ$
⟨*proof*⟩

Multiplication.

global-interpretation *vint-mult*: *binop-onto* ‹$\mathbb{Z}_\circ$› *vint-mult*
⟨*proof*⟩

Misc.

lemma (in $\mathcal{Z}$**)** *vint-in-Vset*[*intro*]: $\mathbb{Z}_\circ \in_\circ$ *Vset* α
⟨*proof*⟩

2.13.4 Rational numbers

Definition

definition *vrat-of-rat* :: *rat* ⇒ *V*
where *vrat-of-rat x* = *vreal-of-real* (*real-of-rat x*)

notation *vrat-of-rat* (‹-$_\mathbb{Q}$›) [999] 999)

declare [[*coercion vrat-of-rat* :: *rat* ⇒ *V*]]

definition *vrat* :: *V* (‹$\mathbb{Q}_\circ$›)
where *vrat* = *set* (*range vrat-of-rat*)

definition *rat-of-vrat* :: *V* ⇒ *rat*
where *rat-of-vrat* = *inv-into UNIV vrat-of-rat*

Rules.

lemma *vrat-of-rat-in-vratI*[*intro*, *simp*]: $a_\mathbb{Q} \in_\circ \mathbb{Q}_\circ$ ⟨*proof*⟩

lemma *vrat-of-rat-in-vratE*[*elim*]:
assumes $a \in_\circ \mathbb{Q}_\circ$
obtains *b* **where** $b_\mathbb{Q}$ = *a*
⟨*proof*⟩

Elementary properties

lemma *vrat-vsubset-vreal*: $\mathbb{Q}_\circ \subseteq_\circ \mathbb{R}_\circ$
⟨*proof*⟩

lemma *vrat-in-Vset-ω2*: $\mathbb{Q}_\circ \in_\circ$ *Vset* (ω + ω)
⟨*proof*⟩

lemma *inj-vrat-of-rat*: *inj vrat-of-rat*
⟨*proof*⟩

lemma *rat-of-vrat-vrat-of-rat*[*simp*]: *rat-of-vrat* ($a_{\mathbb{Q}}$) = *a*
⟨*proof*⟩

Transfer rules.

definition *cr-vrat* :: *V* ⇒ *rat* ⇒ *bool*
where *cr-vrat a b* ⟷ (*a* = *vrat-of-rat b*)

lemma *cr-vrat-right-total*[*transfer-rule*]: *right-total cr-vrat*
⟨*proof*⟩

lemma *cr-vrat-bi-unqie*[*transfer-rule*]: *bi-unique cr-vrat*
⟨*proof*⟩

lemma *cr-vrat-transfer-domain-rule*[*transfer-domain-rule*]:
Domainp cr-vrat = (λ*x*. *x* $\in_\circ \mathbb{Q}_\circ$)
⟨*proof*⟩

lemma *vrat-transfer*[*transfer-rule*]:
(*rel-set cr-vrat*) (*elts* $\mathbb{Q}_\circ$) (*UNIV*::*rat set*)
⟨*proof*⟩

lemma *vrat-of-rat-transfer*[*transfer-rule*]: *cr-vrat* (*vrat-of-rat a*) *a*
⟨*proof*⟩

Operations

lemma *vrat-fsingleton-in-fproduct-vrat*: $[a_{\mathbb{Q}}]_\circ \in_\circ \mathbb{Q}_\circ \mathbin{\widehat{}}_\times 1_{\mathbb{N}}$ ⟨*proof*⟩

lemma *vrat-fpair-in-fproduct-vrat*: $[a_{\mathbb{Q}}, b_{\mathbb{Q}}]_\circ \in_\circ \mathbb{Q}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}$ ⟨*proof*⟩

Zero.

lemma *vrat-zero*: $0_{\mathbb{Q}}$ = (0::*V*) ⟨*proof*⟩

One.

lemma *vrat-one*: $1_{\mathbb{Q}}$ = (1::*V*) ⟨*proof*⟩

Addition.

definition *vrat-plus* :: *V*
where *vrat-plus* =
(λ*x*$\in_\circ\mathbb{Q}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}$. (*rat-of-vrat* ($x(\!|0_{\mathbb{N}}|\!)$) + *rat-of-vrat* ($x(\!|1_{\mathbb{N}}|\!)$))$_{\mathbb{Q}}$)

abbreviation *vrat-plus-app* (**infixl** ‹+$_{\mathbb{Q}}$› 65)
where *vrat-plus-app a b* ≡ *vrat-plus*$(\!|a, b|\!)_\bullet$

lemma *vrat-plus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vrat* ===> *cr-vrat* ===> *cr-vrat*) (+$_{\mathbb{Q}}$) (+)
⟨*proof*⟩

Multiplication.

definition *vrat-mult* :: *V*
where *vrat-mult* =
(λ*x*$\in_\circ\mathbb{Q}_\circ \mathbin{\widehat{}}_\times 2_{\mathbb{N}}$. (*rat-of-vrat* ($x(\!|0_{\mathbb{N}}|\!)$) * *rat-of-vrat* ($x(\!|1_{\mathbb{N}}|\!)$))$_{\mathbb{Q}}$)

abbreviation *vrat-mult-app* (**infixl** ‹$*_{\mathbb{Q}}$› 65)
  where *vrat-mult-app a b* ≡ *vrat-mult*(|*a*, *b*|)$_\bullet$

lemma *vrat-mult-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vrat* ===> *cr-vrat* ===> *cr-vrat*) ($*_{\mathbb{Q}}$) (*)
  ⟨*proof*⟩

Unary minus.

definition *vrat-uminus* :: *V*
  where *vrat-uminus* = ($\lambda x \in_\circ \mathbb{Q}_\circ$. (*uminus* (*rat-of-vrat x*))$_{\mathbb{Q}}$)

abbreviation *vrat-uminus-app* (‹$-_{\mathbb{Q}}$ -› [81] 80)
  where $-_{\mathbb{Q}}$ *a* ≡ *vrat-uminus*(|*a*|)

lemma *vrat-uminus-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vrat* ===> *cr-vrat*) (*vrat-uminus-app*) (*uminus*)
  ⟨*proof*⟩

Multiplicative inverse.

definition *vrat-inverse* :: *V*
  where *vrat-inverse* = ($\lambda x \in_\circ \mathbb{Q}_\circ$. (*inverse* (*rat-of-vrat x*))$_{\mathbb{Q}}$)

abbreviation *vrat-inverse-app* (‹($-^{-1}{}_{\mathbb{Q}}$)› [1000] 999)
  where $a^{-1}{}_{\mathbb{Q}}$ ≡ *vrat-inverse*(|*a*|)

lemma *vrat-inverse-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vrat* ===> *cr-vrat*) (*vrat-inverse-app*) (*inverse*)
  ⟨*proof*⟩

Order.

definition *vrat-le* :: *V*
  where *vrat-le* =
    set {[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ $\mathbb{Q}_\circ$ $\widehat{}_\times$ $2_{\mathbb{N}}$ ∧ *rat-of-vrat a* ≤ *rat-of-vrat b*}

abbreviation *vrat-le′* (‹(-/ $\leq_{\mathbb{Q}}$ -)› [51, 51] 50)
  where *a* $\leq_{\mathbb{Q}}$ *b* ≡ [*a*, *b*]$_\circ$ $\in_\circ$ *vrat-le*

lemma *small-vrat-le*[*simp*]:
  small {[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ $\mathbb{Q}_\circ$ $\widehat{}_\times$ $2_{\mathbb{N}}$ ∧ *rat-of-vrat a* ≤ *rat-of-vrat b*}
⟨*proof*⟩

lemma *vrat-le-transfer*[*transfer-rule*]:
  includes *lifting-syntax*
  shows (*cr-vrat* ===> *cr-vrat* ===> (=)) *vrat-le′* (≤)
  ⟨*proof*⟩

Strict order.

definition *vrat-ls* :: *V*
  where *vrat-ls* =
    set {[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ $\mathbb{Q}_\circ$ $\widehat{}_\times$ $2_{\mathbb{N}}$ ∧ *rat-of-vrat a* < *rat-of-vrat b*}

abbreviation *vrat-ls′* (‹(-/ $<_{\mathbb{Q}}$ -)› [51, 51] 50)
  where *a* $<_{\mathbb{Q}}$ *b* ≡ [*a*, *b*]$_\circ$ $\in_\circ$ *vrat-ls*

lemma *small-vrat-ls*[*simp*]:
small {[*a*, *b*]$_\circ$ | *a b*. [*a*, *b*]$_\circ$ $\in_\circ$ $\mathbb{Q}_\circ$ $\widehat{\ }_\times$ $2_\mathbb{N}$ $\wedge$ *rat-of-vrat a* < *rat-of-vrat b*}
⟨*proof*⟩

lemma *vrat-ls-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vrat* ===> *cr-vrat* ===> (=)) *vrat-ls*′ (<)
⟨*proof*⟩

Subtraction.

definition *vrat-minus* :: *V*
where *vrat-minus* =
(λ*x*∈$_\circ$$\mathbb{Q}_\circ$ $\widehat{\ }_\times$ $2_\mathbb{N}$. (*rat-of-vrat* (*x*(|$0_\mathbb{N}$|)) − *rat-of-vrat* (*x*(|$1_\mathbb{N}$|)))$_\mathbb{Q}$)

abbreviation *vrat-minus-app* (**infixl** ‹−$_\mathbb{Q}$› 65)
where *vrat-minus-app a b* ≡ *vrat-minus*(|*a*, *b*|)$_\bullet$

lemma *vrat-minus-transfer*[*transfer-rule*]:
includes *lifting-syntax*
shows (*cr-vrat* ===> *cr-vrat* ===> *cr-vrat*)
(−$_\mathbb{Q}$) (−)
⟨*proof*⟩

Axioms of an ordered field

The exposition follows Theorem 1.5.5 from the textbook *The Real Numbers and Real Analysis* by E. Bloch [14].

lemma *vrat-zero-closed*: $0_\mathbb{Q}$ $\in_\circ$ $\mathbb{Q}_\circ$ ⟨*proof*⟩

lemma *vrat-one-closed*: $1_\mathbb{Q}$ $\in_\circ$ $\mathbb{Q}_\circ$ ⟨*proof*⟩

lemma *vrat-plus-closed*:
assumes *x* $\in_\circ$ $\mathbb{Q}_\circ$ *y* $\in_\circ$ $\mathbb{Q}_\circ$
shows *x* +$_\mathbb{Q}$ *y* $\in_\circ$ $\mathbb{Q}_\circ$
⟨*proof*⟩

lemma *vrat-mult-closed*:
assumes *x* $\in_\circ$ $\mathbb{Q}_\circ$ **and** *y* $\in_\circ$ $\mathbb{Q}_\circ$
shows *x* *$_\mathbb{Q}$ *y* $\in_\circ$ $\mathbb{Q}_\circ$
⟨*proof*⟩

lemma *vrat-uminus-closed*:
assumes *x* $\in_\circ$ $\mathbb{Q}_\circ$
shows −$_\mathbb{Q}$ *x* $\in_\circ$ $\mathbb{Q}_\circ$
⟨*proof*⟩

lemma *vrat-inverse-closed*:
assumes *x* $\in_\circ$ $\mathbb{Q}_\circ$
shows $x^{-1}{}_\mathbb{Q}$ $\in_\circ$ $\mathbb{Q}_\circ$
⟨*proof*⟩

Associative Law for Addition: Theorem 1.5.5.1.

lemma *vrat-assoc-law-addition*:
assumes *x* $\in_\circ$ $\mathbb{Q}_\circ$ **and** *y* $\in_\circ$ $\mathbb{Q}_\circ$ **and** *z* $\in_\circ$ $\mathbb{Q}_\circ$
shows (*x* +$_\mathbb{Q}$ *y*) +$_\mathbb{Q}$ *z* = *x* +$_\mathbb{Q}$ (*y* +$_\mathbb{Q}$ *z*)
⟨*proof*⟩

Commutative Law for Addition: Theorem 1.5.5.2.

lemma *vrat-commutative-law-addition*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$
shows $x +_\mathbb{Q} y = y +_\mathbb{Q} x$
⟨*proof*⟩

Identity Law for Addition: Theorem 1.5.5.3.

lemma *vrat-identity-law-addition*:
assumes [*simp*]: $x \in_\circ \mathbb{Q}_\circ$
shows $x +_\mathbb{Q} 0_\mathbb{Q} = x$
⟨*proof*⟩

Inverses Law for Addition: Theorem 1.5.5.4.

lemma *vrat-inverses-law-addition*:
assumes [*simp*]: $x \in_\circ \mathbb{Q}_\circ$
shows $x +_\mathbb{Q} (-_\mathbb{Q}\ x) = 0_\mathbb{Q}$
⟨*proof*⟩

Associative Law for Multiplication: Theorem 1.5.5.5.

lemma *vrat-assoc-law-multiplication*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$ **and** $z \in_\circ \mathbb{Q}_\circ$
shows $(x *_\mathbb{Q} y) *_\mathbb{Q} z = x *_\mathbb{Q} (y *_\mathbb{Q} z)$
⟨*proof*⟩

Commutative Law for Multiplication: Theorem 1.5.5.6.

lemma *vrat-commutative-law-multiplication*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$
shows $x *_\mathbb{Q} y = y *_\mathbb{Q} x$
⟨*proof*⟩

Identity Law for multiplication: Theorem 1.5.5.7.

lemma *vrat-identity-law-multiplication*:
assumes $x \in_\circ \mathbb{Q}_\circ$
shows $x *_\mathbb{Q} 1_\mathbb{Q} = x$
⟨*proof*⟩

Inverses Law for Multiplication: Definition 2.2.1.8.

lemma *vrat-inverses-law-multiplication*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $x \neq 0_\mathbb{Q}$
shows $x *_\mathbb{Q} x^{-1}{}_\mathbb{Q} = 1_\mathbb{Q}$
⟨*proof*⟩

Distributive Law for Multiplication: Theorem 1.5.5.9.

lemma *vrat-distributive-law*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$ **and** $z \in_\circ \mathbb{Q}_\circ$
shows $x *_\mathbb{Q} (y +_\mathbb{Q} z) = (x *_\mathbb{Q} y) +_\mathbb{Q} (x *_\mathbb{Q} z)$
⟨*proof*⟩

Trichotomy Law: Theorem 1.5.5.10.

lemma *vrat-trichotomy-law*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$
shows
$(x <_\mathbb{Q} y \wedge \sim(x = y) \wedge \sim(y <_\mathbb{Q} x)) \vee$
$(\sim(x <_\mathbb{Q} y) \wedge x = y \wedge \sim(y <_\mathbb{Q} x)) \vee$
$(\sim(x <_\mathbb{Q} y) \wedge \sim(x = y) \wedge y <_\mathbb{Q} x)$

⟨*proof*⟩

Transitive Law: Theorem 1.5.5.11.

lemma *vrat-transitive-law*:
assumes $x \in_\circ \mathbb{Q}_\circ$
and $y \in_\circ \mathbb{Q}_\circ$
and $z \in_\circ \mathbb{Q}_\circ$
and $x <_{\mathbb{Q}} y$
and $y <_{\mathbb{Q}} z$
shows $x <_{\mathbb{Q}} z$
⟨*proof*⟩

Addition Law of Order: Theorem 1.5.5.12.

lemma *vrat-addition-law-of-order*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$ **and** $z \in_\circ \mathbb{Q}_\circ$ **and** $x <_{\mathbb{Q}} y$
shows $x +_{\mathbb{Q}} z <_{\mathbb{Q}} y +_{\mathbb{Q}} z$
⟨*proof*⟩

Multiplication Law of Order: Theorem 1.5.5.13.

lemma *vrat-multiplication-law-of-order*:
assumes $x \in_\circ \mathbb{Q}_\circ$
and $y \in_\circ \mathbb{Q}_\circ$
and $z \in_\circ \mathbb{Q}_\circ$
and $x <_{\mathbb{Q}} y$
and $0_{\mathbb{Q}} <_{\mathbb{Q}} z$
shows $x *_{\mathbb{Q}} z <_{\mathbb{Q}} y *_{\mathbb{Q}} z$
⟨*proof*⟩

Non-Triviality: Theorem 1.5.5.14.

lemma *vrat-non-triviality*: $0_{\mathbb{Q}} \neq 1_{\mathbb{Q}}$
⟨*proof*⟩

Fundamental properties of other operations

Minus.

lemma *vrat-minus-closed*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$
shows $x -_{\mathbb{Q}} y \in_\circ \mathbb{Q}_\circ$
⟨*proof*⟩

lemma *vrat-minus-eq-plus-uminus*:
assumes $x \in_\circ \mathbb{Q}_\circ$ **and** $y \in_\circ \mathbb{Q}_\circ$
shows $x -_{\mathbb{Q}} y = x +_{\mathbb{Q}} (-_{\mathbb{Q}} y)$
⟨*proof*⟩

Unary minus.

lemma *vrat-uminus-uminus*:
assumes $x \in_\circ \mathbb{Q}_\circ$
shows $x = -_{\mathbb{Q}} (-_{\mathbb{Q}} x)$
⟨*proof*⟩

Multiplicative inverse.

lemma *vrat-inverse-inverse*:
assumes $x \in_\circ \mathbb{Q}_\circ$
shows $x = (x^{-1}{}_{\mathbb{Q}})^{-1}{}_{\mathbb{Q}}$
⟨*proof*⟩

Further properties

Addition.

global-interpretation *vrat-plus*: *binop-onto* ‹$\mathbb{Q}_\circ$› *vrat-plus*
⟨*proof*⟩

Unary minus.

global-interpretation *vrat-uminus*: *v*11 *vrat-uminus*
  rewrites *vrat-uminus-vdomain*[*simp*]: $\mathcal{D}_\circ$ *vrat-uminus* = $\mathbb{Q}_\circ$
    and *vrat-uminus-vrange*[*simp*]: $\mathcal{R}_\circ$ *vrat-uminus* = $\mathbb{Q}_\circ$
⟨*proof*⟩

Multiplication.

global-interpretation *vrat-mult*: *binop-onto* ‹$\mathbb{Q}_\circ$› *vrat-mult*
⟨*proof*⟩

Multiplicative inverse.

global-interpretation *vrat-inverse*: *v*11 *vrat-inverse*
  rewrites *vrat-inverse-vdomain*[*simp*]: $\mathcal{D}_\circ$ *vrat-inverse* = $\mathbb{Q}_\circ$
    and *vrat-inverse-vrange*[*simp*]: $\mathcal{R}_\circ$ *vrat-inverse* = $\mathbb{Q}_\circ$
⟨*proof*⟩

Misc.

lemma (**in** $\mathcal{Z}$) *vrat-in-Vset*[*intro*]: $\mathbb{Q}_\circ \in_\circ$ *Vset* α
  ⟨*proof*⟩

2.13.5 Upper bound on the cardinality of the continuum for V

lemma *inj-on-inv-vreal-of-real*: *inj-on* (*inv vreal-of-real*) (*elts* $\mathbb{R}_\circ$)
  ⟨*proof*⟩

lemma *vreal-vlepoll-VPow-omega*: $\mathbb{R}_\circ \lesssim_\circ$ *VPow* ω
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *vreal-in-Vset*[*intro*]: $\mathbb{R}_\circ \in_\circ$ *Vset* α
  ⟨*proof*⟩

2.14 Example I: absence of replacement in $V_{\omega+\omega}$

The statement of the main result presented in this subsection can be found in [5][3]

definition *repl-ex-fun* :: *V*
where *repl-ex-fun* = $(\lambda i \in_\circ \omega.\ Vfrom\ \omega\ i)$

mk-VLambda *repl-ex-fun-def*
|*vsv repl-ex-fun-vsv*|
|*vdomain repl-ex-fun-vdomain*|
|*app repl-ex-fun-app*|

lemma *repl-ex-fun-vrange*: $\mathcal{R}_\circ$ *repl-ex-fun* $\subseteq_\circ$ *Vset* $(\omega + \omega)$
⟨*proof*⟩

lemma *Limit-vsv-not-in-Vset-if-vrange-not-in-Vset*:
assumes *Limit* α **and** $\mathcal{R}_\circ$ $f \notin_\circ$ *Vset* α
shows $f \notin_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-not-in-Vset*:
assumes *Ord* α
shows $\alpha \notin_\circ$ *Vset* α
⟨*proof*⟩

lemma *Ord-succ-vsusbset-Vfrom-succ*:
assumes *Transset A* **and** *Ord a* **and** $a \in_\circ$ *Vfrom A i*
shows *succ a* $\subseteq_\circ$ *Vfrom A* (*succ i*)
⟨*proof*⟩

lemma *Ord-succ-in-Vfrom-succ*:
assumes *Transset A* **and** *Ord a* **and** $a \in_\circ$ *Vfrom A i*
shows *succ a* $\in_\circ$ *Vfrom A* (*succ* (*succ i*))
⟨*proof*⟩

lemma ω*-vplus-in-Vfrom-*ω:
assumes $j \in_\circ \omega$
shows $\omega + j \in_\circ$ *Vfrom* ω (*succ* $(2_{\mathbb{N}} * j)$)
⟨*proof*⟩

lemma *repl-ex-fun-vrange-not-in-Vset*: $\mathcal{R}_\circ$ *repl-ex-fun* $\notin_\circ$ *Vset* $(\omega + \omega)$
⟨*proof*⟩

lemma *repl-ex-fun-not-in-Vset*: *repl-ex-fun* $\notin_\circ$ *Vset* $(\omega + \omega)$
⟨*proof*⟩

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

2.15 Example II: topological spaces

2.15.1 Background

The section presents elements of the foundations of the theory of topological spaces formalized in *ZFC in HOL*. The definitions were adopted (with amendments) from the main library of Isabelle/HOL and [35].

named-theorems *ts-struct-field-simps*

2.15.2 $\mathcal{Z}$-sequence

locale $\mathcal{Z}$*-vfsequence* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{S}$ **for** α $\mathfrak{S}$ +
  assumes *vrange-vsubset-Vset*: $\mathcal{R}_\circ$ $\mathfrak{S}$ $\subseteq_\circ$ *Vset* α

Rules.

lemma $\mathcal{Z}$*-vfsequenceI*[*intro*]:
  assumes $\mathcal{Z}$ α **and** *vfsequence* $\mathfrak{S}$ **and** $\mathcal{R}_\circ$ $\mathfrak{S}$ $\subseteq_\circ$ *Vset* α
  shows $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
  ⟨*proof*⟩

lemmas $\mathcal{Z}$*-vfsequenceD*[*dest*] = $\mathcal{Z}$*-vfsequence.axioms*

lemma $\mathcal{Z}$*-vfsequenceE*[*elim*]:
  assumes $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
  obtains $\mathcal{Z}$ α **and** *vfsequence* $\mathfrak{S}$ **and** $\mathcal{R}_\circ$ $\mathfrak{S}$ $\subseteq_\circ$ *Vset* α
  ⟨*proof*⟩

Elementary properties.

context $\mathcal{Z}$*-vfsequence*
begin

lemma (**in** $\mathcal{Z}$*-vfsequence*) $\mathcal{Z}$*-vfsequence-vdomain-in-Vset*[*intro*, *simp*]:
  $\mathcal{D}_\circ$ $\mathfrak{S}$ $\in_\circ$ *Vset* α
  ⟨*proof*⟩

lemma (**in** $\mathcal{Z}$*-vfsequence*) $\mathcal{Z}$*-vfsequence-vrange-in-Vset*[*intro*, *simp*]:
  $\mathcal{R}_\circ$ $\mathfrak{S}$ $\in_\circ$ *Vset* α
  ⟨*proof*⟩

lemma (**in** $\mathcal{Z}$*-vfsequence*) $\mathcal{Z}$*-vfsequence-struct-in-Vset*: $\mathfrak{S}$ $\in_\circ$ *Vset* α
  ⟨*proof*⟩

end

2.15.3 Topological space

definition $\mathcal{A}$ **where** [*ts-struct-field-simps*]: $\mathcal{A}$ = 0
definition $\mathcal{T}$ **where** [*ts-struct-field-simps*]: $\mathcal{T}$ = $1_{\mathbb{N}}$

locale $\mathcal{Z}$*-ts* = $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$ **for** α $\mathfrak{S}$ +
  assumes $\mathcal{Z}$*-ts-length*: $2_{\mathbb{N}}$ ≤ *vcard* $\mathfrak{S}$
    and $\mathcal{Z}$*-ts-closed*[*intro*]: A $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$ ⟹ A $\subseteq_\circ$ $\mathfrak{S}(\!|\mathcal{A}|\!)$
    and $\mathcal{Z}$*-ts-domain*[*intro*, *simp*]: $\mathfrak{S}(\!|\mathcal{A}|\!)$ $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$
    and $\mathcal{Z}$*-ts-vintersection*[*intro*]:
      A $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$ ⟹ B $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$ ⟹ A $\cap_\circ$ B $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$
    and $\mathcal{Z}$*-ts-VUnion*[*intro*]: X $\subseteq_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$ ⟹ $\bigcup_\circ X$ $\in_\circ$ $\mathfrak{S}(\!|\mathcal{T}|\!)$

Rules.

lemma $\mathcal{Z}$*-tsI*[*intro*]:
assumes $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
and $2_{\mathbb{N}} \leq$ *vcard* $\mathfrak{S}$
and $\bigwedge A.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{A}|\!)$
and $\mathfrak{S}(\!|\mathcal{A}|\!) \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge A\ B.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge X.\ X \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
shows $\mathcal{Z}$*-ts* α $\mathfrak{S}$
⟨*proof*⟩

lemma $\mathcal{Z}$*-tsD*[*dest*]:
assumes $\mathcal{Z}$*-ts* α $\mathfrak{S}$
shows $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
and $2_{\mathbb{N}} \leq$ *vcard* $\mathfrak{S}$
and $\bigwedge A.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{A}|\!)$
and $\mathfrak{S}(\!|\mathcal{A}|\!) \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge A\ B.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge X.\ X \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
⟨*proof*⟩

lemma $\mathcal{Z}$*-tsE*[*elim*]:
assumes $\mathcal{Z}$*-ts* α $\mathfrak{S}$
obtains $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
and $2_{\mathbb{N}} \leq$ *vcard* $\mathfrak{S}$
and $\bigwedge A.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{A}|\!)$
and $\mathfrak{S}(\!|\mathcal{A}|\!) \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge A\ B.\ A \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
and $\bigwedge X.\ X \subseteq_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!) \Longrightarrow \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
⟨*proof*⟩

Elementary properties.

lemma (**in** $\mathcal{Z}$*-ts*) $\mathcal{Z}$*-ts-vempty-in-ts*: $0 \in_{\circ} \mathfrak{S}(\!|\mathcal{T}|\!)$
⟨*proof*⟩

2.15.4 Indiscrete topology

definition *ts-indiscrete* :: $V \Rightarrow V$
where *ts-indiscrete* $A = [A,\ set\ \{0,\ A\}]_{\circ}$

named-theorems *ts-indiscrete-simps*

lemma *ts-indiscrete-*$\mathcal{A}$[*ts-indiscrete-simps*]: *ts-indiscrete* $A(\!|\mathcal{A}|\!) = A$
⟨*proof*⟩

lemma *ts-indiscrete-*$\mathcal{T}$[*ts-indiscrete-simps*]: *ts-indiscrete* $A(\!|\mathcal{T}|\!) = set\ \{0,\ A\}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) $\mathcal{Z}$*-ts-ts-indiscrete*:
assumes $A \in_{\circ} Vset\ \alpha$
shows $\mathcal{Z}$*-ts* α (*ts-indiscrete* A)
⟨*proof*⟩

2.16 Example III: abstract algebra

2.16.1 Background

The section presents several examples of algebraic structures formalized in *ZFC in HOL*. The definitions were adopted (with amendments) from the main library of Isabelle/HOL.

named-theorems *sgrp-struct-field-simps*

lemmas [*sgrp-struct-field-simps*] = $\mathcal{A}$-*def*

2.16.2 Semigroup

Foundations

definition *mbinop* **where** [*sgrp-struct-field-simps*]: *mbinop* = $1_{\mathbf{N}}$

locale $\mathcal{Z}$-*sgrp-basis* = $\mathcal{Z}$-*vfsequence* α $\mathfrak{S}$ + *op*: *binop* ‹$\mathfrak{S}(\!|\mathcal{A}|\!)$› ‹$\mathfrak{S}(\!|$*mbinop*$|\!)$›
  for α $\mathfrak{S}$ +
  assumes $\mathcal{Z}$-*sgrp-length*: *vcard* $\mathfrak{S}$ = $2_{\mathbf{N}}$

abbreviation *sgrp-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\odot_{\circ 1}$› 70)
  where *sgrp-app* $\mathfrak{S}$ a b ≡ $\mathfrak{S}(\!|$*mbinop*$|\!)(\!|a, b|\!)_\bullet$
notation *sgrp-app* (**infixl** ‹$\odot_\circ$› 70)

Rules.

lemma $\mathcal{Z}$-*sgrp-basisI*[*intro*]:
  assumes $\mathcal{Z}$-*vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $2_{\mathbf{N}}$
    and *binop* ($\mathfrak{S}(\!|\mathcal{A}|\!)$) ($\mathfrak{S}(\!|$*mbinop*$|\!)$)
  shows $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
  ⟨*proof*⟩

lemma $\mathcal{Z}$-*sgrp-basisD*[*dest*]:
  assumes $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
  shows $\mathcal{Z}$-*vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $2_{\mathbf{N}}$
    and *binop* ($\mathfrak{S}(\!|\mathcal{A}|\!)$) ($\mathfrak{S}(\!|$*mbinop*$|\!)$)
  ⟨*proof*⟩

lemma $\mathcal{Z}$-*sgrp-basisE*[*elim*]:
  assumes $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
  shows $\mathcal{Z}$-*vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $2_{\mathbf{N}}$
    and *binop* ($\mathfrak{S}(\!|\mathcal{A}|\!)$) ($\mathfrak{S}(\!|$*mbinop*$|\!)$)
  ⟨*proof*⟩

Simple semigroup

locale $\mathcal{Z}$-*sgrp* = $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$ **for** α $\mathfrak{S}$ +
  assumes $\mathcal{Z}$-*sgrp-assoc*:
    ⟦ $a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$; $b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$; $c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$ ⟧ ⟹
      $(a \odot_{\circ\mathfrak{S}} b) \odot_{\circ\mathfrak{S}} c = a \odot_{\circ\mathfrak{S}} (b \odot_{\circ\mathfrak{S}} c)$

Rules.

lemma $\mathcal{Z}$-*sgrpI*[*intro*]:
  assumes $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
    and $\bigwedge a\ b\ c$. ⟦ $a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$; $b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$; $c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)$ ⟧ ⟹
      $(a \odot_{\circ\mathfrak{S}} b) \odot_{\circ\mathfrak{S}} c = a \odot_{\circ\mathfrak{S}} (b \odot_{\circ\mathfrak{S}} c)$

shows $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
⟨*proof*⟩

lemma $\mathcal{Z}$-*sgrpD*[*dest*]:
assumes $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
shows $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
$(a \odot_{\circ\mathfrak{S}} b) \odot_{\circ\mathfrak{S}} c = a \odot_{\circ\mathfrak{S}} (b \odot_{\circ\mathfrak{S}} c)$
⟨*proof*⟩

lemma $\mathcal{Z}$-*sgrpE*[*elim*]:
assumes $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
obtains $\mathcal{Z}$-*sgrp-basis* α $\mathfrak{S}$
and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
$(a \odot_{\circ\mathfrak{S}} b) \odot_{\circ\mathfrak{S}} c = a \odot_{\circ\mathfrak{S}} (b \odot_{\circ\mathfrak{S}} c)$
⟨*proof*⟩

2.16.3 Commutative semigroup

locale $\mathcal{Z}$-*csgrp* = $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$ **for** α $\mathfrak{S}$ +
assumes $\mathcal{Z}$-*csgrp-commutative*:
$[\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow a \odot_{\circ\mathfrak{S}} b = b \odot_{\circ\mathfrak{S}} a$

Rules.

lemma $\mathcal{Z}$-*csgrpI*[*intro*]:
assumes $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow a \odot_{\circ\mathfrak{S}} b = b \odot_{\circ\mathfrak{S}} a$
shows $\mathcal{Z}$-*csgrp* α $\mathfrak{S}$
⟨*proof*⟩

lemma $\mathcal{Z}$-*csgrpD*[*dest*]:
assumes $\mathcal{Z}$-*csgrp* α $\mathfrak{S}$
shows $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow a \odot_{\circ\mathfrak{S}} b = b \odot_{\circ\mathfrak{S}} a$
⟨*proof*⟩

lemma $\mathcal{Z}$-*csgrpE*[*elim*]:
assumes $\mathcal{Z}$-*csgrp* α $\mathfrak{S}$
obtains $\mathcal{Z}$-*sgrp* α $\mathfrak{S}$
and $\bigwedge a\ b.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow a \odot_{\circ\mathfrak{S}} b = b \odot_{\circ\mathfrak{S}} a$
⟨*proof*⟩

2.16.4 Semiring

Foundations

definition *vplus* :: V **where** [*sgrp-struct-field-simps*]: *vplus* = $1_{\mathbf{N}}$
definition *vmult* :: V **where** [*sgrp-struct-field-simps*]: *vmult* = $2_{\mathbf{N}}$

abbreviation *vplus-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** ‹$+_{\circ 1}$› 65)
where $a +_{\circ\mathfrak{S}} b \equiv \mathfrak{S}(\!|vplus|\!)(\!|a,b|\!)_\bullet$
notation *vplus-app* (**infixl** ‹$+_{\circ 1}$› 65)

abbreviation *vmult-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** ‹$*_{\circ 1}$› 70)
where $a *_{\circ\mathfrak{S}} b \equiv \mathfrak{S}(\!|vmult|\!)(\!|a,b|\!)_\bullet$
notation *vmult-app* (**infixl** ‹$*_{\circ 1}$› 70)

Simple semiring

locale $\mathcal{Z}$*-srng* = $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$ **for** α $\mathfrak{S}$ +
  assumes $\mathcal{Z}$*-srng-length*: *vcard* $\mathfrak{S}$ = $3_{\mathbb{N}}$
    and $\mathcal{Z}$*-srng-*$\mathcal{Z}$*-csgrp-vplus*: $\mathcal{Z}$*-csgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$
    and $\mathcal{Z}$*-srng-*$\mathcal{Z}$*-sgrp-vmult*: $\mathcal{Z}$*-sgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$
    and $\mathcal{Z}$*-srng-distrib-right*:
      $[\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
        $(a +_{\circ\mathfrak{S}} b) *_{\circ\mathfrak{S}} c = (a *_{\circ\mathfrak{S}} c) +_{\circ\mathfrak{S}} (b *_{\circ\mathfrak{S}} c)$
    and $\mathcal{Z}$*-srng-distrib-left*:
      $[\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
        $a *_{\circ\mathfrak{S}} (b +_{\circ\mathfrak{S}} c) = (a *_{\circ\mathfrak{S}} b) +_{\circ\mathfrak{S}} (a *_{\circ\mathfrak{S}} c)$
begin

sublocale *vplus*: $\mathcal{Z}$*-csgrp* α ‹$[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$›
  rewrites $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ(\!|\mathcal{A}|\!) = \mathfrak{S}(\!|\mathcal{A}|\!)$
    and $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ(\!|mbinop|\!) = \mathfrak{S}(\!|vplus|\!)$
    and *sgrp-app* $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$ = *vplus-app* $\mathfrak{S}$
⟨*proof*⟩

sublocale *vmult*: $\mathcal{Z}$*-sgrp* α ‹$[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$›
  rewrites $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ(\!|\mathcal{A}|\!) = \mathfrak{S}(\!|\mathcal{A}|\!)$
    and $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ(\!|mbinop|\!) = \mathfrak{S}(\!|vmult|\!)$
    and *sgrp-app* $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$ = *vmult-app* $\mathfrak{S}$
⟨*proof*⟩

end

Rules.

lemma $\mathcal{Z}$*-srngI*[*intro*]:
  assumes $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $3_{\mathbb{N}}$
    and $\mathcal{Z}$*-csgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$
    and $\mathcal{Z}$*-sgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$
    and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
      $(a +_{\circ\mathfrak{S}} b) *_{\circ\mathfrak{S}} c = (a *_{\circ\mathfrak{S}} c) +_{\circ\mathfrak{S}} (b *_{\circ\mathfrak{S}} c)$
    and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
      $a *_{\circ\mathfrak{S}} (b +_{\circ\mathfrak{S}} c) = (a *_{\circ\mathfrak{S}} b) +_{\circ\mathfrak{S}} (a *_{\circ\mathfrak{S}} c)$
  shows $\mathcal{Z}$*-srng* α $\mathfrak{S}$
  ⟨*proof*⟩

lemma $\mathcal{Z}$*-srngD*[*dest*]:
  assumes $\mathcal{Z}$*-srng* α $\mathfrak{S}$
  shows $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $3_{\mathbb{N}}$
    and $\mathcal{Z}$*-csgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$
    and $\mathcal{Z}$*-sgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$
    and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
      $(a +_{\circ\mathfrak{S}} b) *_{\circ\mathfrak{S}} c = (a *_{\circ\mathfrak{S}} c) +_{\circ\mathfrak{S}} (b *_{\circ\mathfrak{S}} c)$
    and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
      $a *_{\circ\mathfrak{S}} (b +_{\circ\mathfrak{S}} c) = (a *_{\circ\mathfrak{S}} b) +_{\circ\mathfrak{S}} (a *_{\circ\mathfrak{S}} c)$
  ⟨*proof*⟩

lemma $\mathcal{Z}$*-srngE*[*elim*]:
  assumes $\mathcal{Z}$*-srng* α $\mathfrak{S}$
  obtains $\mathcal{Z}$*-vfsequence* α $\mathfrak{S}$
    and *vcard* $\mathfrak{S}$ = $3_{\mathbb{N}}$
    and $\mathcal{Z}$*-csgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vplus|\!)]_\circ$

and $\mathcal{Z}$*-sgrp* α $[\mathfrak{S}(\!|\mathcal{A}|\!), \mathfrak{S}(\!|vmult|\!)]_\circ$
and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
$(a +_{\circ\mathfrak{S}} b) *_{\circ\mathfrak{S}} c = (a *_{\circ\mathfrak{S}} c) +_{\circ\mathfrak{S}} (b *_{\circ\mathfrak{S}} c)$
and $\bigwedge a\ b\ c.\ [\![\ a \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ b \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!);\ c \in_\circ \mathfrak{S}(\!|\mathcal{A}|\!)\ ]\!] \Longrightarrow$
$a *_{\circ\mathfrak{S}} (b +_{\circ\mathfrak{S}} c) = (a *_{\circ\mathfrak{S}} b) +_{\circ\mathfrak{S}} (a *_{\circ\mathfrak{S}} c)$
⟨*proof*⟩

2.16.5 Integer numbers form a semiring

definition *vint-struct* :: V ($\langle\mathfrak{S}_{\mathbb{Z}}\rangle$)
where *vint-struct* = $[\mathbb{Z}_\circ, \textit{vint-plus}, \textit{vint-mult}]_\circ$

named-theorems *vint-struct-simps*

lemma *vint-struct-*$\mathcal{A}$[*vint-struct-simps*]: $\mathfrak{S}_{\mathbb{Z}}(\!|\mathcal{A}|\!) = \mathbb{Z}_\circ$
⟨*proof*⟩

lemma *vint-struct-vplus*[*vint-struct-simps*]: $\mathfrak{S}_{\mathbb{Z}}(\!|vplus|\!) = \textit{vint-plus}$
⟨*proof*⟩

lemma *vint-struct-vmult*[*vint-struct-simps*]: $\mathfrak{S}_{\mathbb{Z}}(\!|vmult|\!) = \textit{vint-mult}$
⟨*proof*⟩

context $\mathcal{Z}$
begin

lemma $\mathcal{Z}$*-srng-vint*: $\mathcal{Z}$*-srng* α $\mathfrak{S}_{\mathbb{Z}}$
⟨*proof*⟩

Interpretation.

interpretation *vint*: $\mathcal{Z}$*-srng* α $\langle\mathfrak{S}_{\mathbb{Z}}\rangle$
rewrites $\mathfrak{S}_{\mathbb{Z}}(\!|\mathcal{A}|\!) = \mathbb{Z}_\circ$
and $\mathfrak{S}_{\mathbb{Z}}(\!|vplus|\!) = \textit{vint-plus}$
and $\mathfrak{S}_{\mathbb{Z}}(\!|vmult|\!) = \textit{vint-mult}$
and *vplus-app* ($\mathfrak{S}_{\mathbb{Z}}$) = *vint-plus-app*
and *vmult-app* ($\mathfrak{S}_{\mathbb{Z}}$) = *vint-mult-app*
⟨*proof*⟩

thm *vint.vmult.*$\mathcal{Z}$*-sgrp-assoc*
thm *vint.vplus.*$\mathcal{Z}$*-sgrp-assoc*
thm *vint.*$\mathcal{Z}$*-srng-distrib-left*

end

Digraphs

3.1 Introduction

3.1.1 Background

Many concepts that are normally associated with category theory can be generalized to directed graphs. It is the goal of this chapter to expose these generalized concepts and provide the relevant foundations for the development of the notion of a semicategory in the next chapter. It is important to note, however, that it is not the goal of this chapter to present a comprehensive canonical theory of directed graphs. Nonetheless, there is little that could prevent one from extending this body of work by providing canonical results from the theory of directed graphs.

3.1.2 Preliminaries

declare *One-nat-def*[*simp del*]

named-theorems *slicing-simps*
named-theorems *slicing-commute*
named-theorems *slicing-intros*

named-theorems *dg-op-simps*
named-theorems *dg-op-intros*

named-theorems *dg-cs-simps*
named-theorems *dg-cs-intros*

named-theorems *dg-shared-cs-simps*
named-theorems *dg-shared-cs-intros*

3.1.3 CS setup for foundations

named-theorems *V-cs-simps*
named-theorems *V-cs-intros*

named-theorems *Ord-cs-simps*
named-theorems *Ord-cs-intros*

Basic *HOL*

lemma (**in** *semilattice-sup*) *sup-commute'*:
  shows $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcup b = b' \sqcup a'$
    and $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcup b' = b \sqcup a'$
    and $b' = b \Longrightarrow a' = a \Longrightarrow a' \sqcup b = b' \sqcup a$
    and $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcup b' = b \sqcup a'$
    and $b' = b \Longrightarrow a' = a \Longrightarrow a' \sqcup b' = b \sqcup a$
  ⟨*proof*⟩

lemma (**in** *semilattice-inf*) *inf-commute'*:

shows $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcap b = b' \sqcap a'$
and $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcap b' = b \sqcap a'$
and $b' = b \Longrightarrow a' = a \Longrightarrow a' \sqcap b = b' \sqcap a$
and $b' = b \Longrightarrow a' = a \Longrightarrow a \sqcap b' = b \sqcap a'$
and $b' = b \Longrightarrow a' = a \Longrightarrow a' \sqcap b' = b \sqcap a$
⟨*proof*⟩

lemmas [*V-cs-simps*] =
if-P
if-not-P
*inf.absorb*1
*inf.absorb*2
*sup.absorb*1
*sup.absorb*2
add-0-*right*
add-0

lemmas [*V-cs-intros*] =
conjI
sup-commute'
inf-commute'
sup.commute
inf.commute

Lists for *HOL*

lemma *list-all-singleton*: *list-all P* [*x*] = *P x* ⟨*proof*⟩

lemma *replicate-one*: *replicate* 1 *x* = [*x*]
⟨*proof*⟩

lemma *list-all-mono*:
assumes *list-all P xs* **and** $P \leq Q$
shows *list-all Q xs*
⟨*proof*⟩

lemma *pred-in-set-mono*:
assumes $S \subseteq T$
shows $(\lambda x.\ x \in S) \leq (\lambda x.\ x \in T)$
⟨*proof*⟩

lemma *elts-subset-mono*:
assumes $S \subseteq_\circ T$
shows *elts S* ⊆ *elts T*
⟨*proof*⟩

lemma *list-all-replicate*:
assumes *P x*
shows *list-all P* (*replicate n x*)
⟨*proof*⟩

lemma *list-all-set*:
assumes *list-all P xs* **and** *x* ∈ *list.set xs*
shows *P x*
⟨*proof*⟩

lemma *list-map-id*:
assumes *list-all* (λ*x*. *f x* = *x*) *xs*

shows *map f xs = xs*
⟨*proof*⟩

lemmas [*V-cs-simps*] =
List.append.append-Nil
List.append-Nil2
List.append.append-Cons
List.rev.simps(1)
list.map(1,2)
rev.simps(2)
List.map-append
list-all-append
replicate.replicate-0
rev-replicate
semiring-1-*class.of-nat*-0
group-add-class.minus-zero
group-add-class.minus-minus
replicate.replicate-Suc
replicate-one
list-all-singleton

lemmas [*V-cs-intros*] =
exI
pred-in-set-mono
elts-subset-mono
list-all-replicate

Foundations

abbreviation (*input*) *if3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *if3 a b c* ≡
(
$\lambda i.$ *if* $i = 0 \Rightarrow a$
| $i = 1_{\mathbb{N}} \Rightarrow b$
| *otherwise* ⇒ *c*
)
lemma *if3*-0[*V-cs-simps*]: *if3 a b c* 0 = *a* ⟨*proof*⟩
lemma *if3*-1[*V-cs-simps*]: *if3 a b c* $(1_{\mathbb{N}})$ = *b* ⟨*proof*⟩
lemma *if3*-2[*V-cs-simps*]: *if3 a b c* $(2_{\mathbb{N}})$ = *c* ⟨*proof*⟩

lemma *vinsertI1′*:
assumes *x′* = *x*
shows *x* ∈$_\circ$ *vinsert x′ A*
⟨*proof*⟩

lemma *in-vsingleton*[*V-cs-intros*]:
assumes *f* = *a*
shows *f* ∈$_\circ$ *set* {*a*}
⟨*proof*⟩

lemma *a-in-succ-a*: *a* ∈$_\circ$ *succ a* ⟨*proof*⟩

lemma *a-in-succ-xI*:
assumes *a* ∈$_\circ$ *x*
shows *a* ∈$_\circ$ *succ x*
⟨*proof*⟩

lemma *vone-ne*[*V-cs-intros*]: $1_{\mathbb{N}} \neq 0$ ⟨*proof*⟩

lemmas [*V-cs-simps*] =
vinsert-set-insert-eq
beta
set-empty
vcard-0

lemmas [*V-cs-intros*] =
mem-not-refl
succ-notin-self
vset-neq-1
vset-neq-2
nin-vinsertI
*vinsertI*1′
*vinsertI*2
vfinite-vinsert
vfinite-vsingleton
vdisjnt-nin-right
vdisjnt-nin-left
*vunionI*1
*vunionI*2
vunion-in-VsetI
vintersection-in-VsetI
vsubset-reflexive
vsingletonI
small-insert small-empty
Limit-vtimes-in-VsetI
Limit-VPow-in-VsetI
a-in-succ-a
vsubset-vempty

Binary relations

lemma *vtimesI*′[*V-cs-intros*]:
assumes $ab = \langle a, b \rangle$ **and** $a \in_\circ A$ **and** $b \in_\circ B$
shows $ab \in_\circ A \times_\circ B$
⟨*proof*⟩

lemma *vrange-vcomp-vsubset*[*V-cs-intros*]:
assumes $\mathcal{R}_\circ\ r \subseteq_\circ B$
shows $\mathcal{R}_\circ\ (r \circ_\circ s) \subseteq_\circ B$
⟨*proof*⟩

lemma *vrange-vconst-on-vsubset*[*V-cs-intros*]:
assumes $a \in_\circ R$
shows $\mathcal{R}_\circ\ (\mathit{vconst\text{-}on}\ A\ a) \subseteq_\circ R$
⟨*proof*⟩

lemma *vrange-vcomp-eq-vrange*[*V-cs-simps*]:
assumes $\mathcal{D}_\circ\ r = \mathcal{R}_\circ\ s$
shows $\mathcal{R}_\circ\ (r \circ_\circ s) = \mathcal{R}_\circ\ r$
⟨*proof*⟩

lemmas [*V-cs-simps*] =
vdomain-vsingleton
vdomain-vlrestriction
vdomain-vlrestriction-vsubset
vdomain-vcomp-vsubset

vdomain-vconverse
vrange-vconverse
vdomain-vconst-on
vconverse-vtimes
vdomain-VLambda

lemmas [*V-cs-intros*] = *vcpower-vsubset-mono*

Single-valued functions

lemmas (in *vsv*) [*V-cs-intros*] = *vsv-axioms*

lemma *vpair-app*:
assumes $j = a$
shows *set* $\{\langle a,\ b\rangle\}(\!|j|\!) = b$
⟨*proof*⟩

lemmas [*V-cs-simps*] =
vpair-app
vsv.vlrestriction-app
vsv-vcomp-at
vid-on-atI

lemmas (in *vsv*) [*V-cs-intros*] = *vsv-vimageI2′*

lemmas [*V-cs-intros*] =
vsv-vsingleton
vsv.vsv-vimageI2′
vsv-vcomp

Injective single-valued functions

lemmas (in *v11*) [*V-cs-intros*] = *v11-axioms*

lemma (in *v11*) *v11-vconverse-app-in-vdomain′*:
assumes $y \in_\circ \mathcal{R}_\circ\ r$ **and** $A = \mathcal{D}_\circ\ r$
shows ${r^{-1}}_\circ(\!|y|\!) \in_\circ A$
⟨*proof*⟩

lemmas (in *v11*) [*V-cs-intros*] = *v11-vconverse-app-in-vdomain′*
lemmas [*V-cs-intros*] = *v11.v11-vconverse-app-in-vdomain′*

lemmas (in *v11*) [*V-cs-simps*] =
v11-app-if-vconverse-app[*rotated* −2]
v11-app-vconverse-app
v11-vconverse-app-app

lemmas [*V-cs-simps*] =
v11.v11-vconverse-app[*rotated* −1]
v11.v11-app-vconverse-app
v11.v11-vconverse-app-app

lemmas [*V-cs-intros*] =
v11D(1)
v11.v11-vconverse
v11-vcomp

Operations on indexed families of sets

lemmas [*V-cs-simps*] =
vprojection-app
vprojection-vdomain

lemmas [*V-cs-intros*] = *vprojection-vsv*

Finite sequences

lemmas (**in** *vfsequence*) [*V-cs-intros*] = *vfsequence-axioms*

lemmas (**in** *vfsequence*) [*V-cs-simps*] = *vfsequence-vdomain*
lemmas [*V-cs-simps*] = *vfsequence.vfsequence-vdomain*

lemmas [*V-cs-intros*] =
vfsequence.vfsequence-vcons
vfsequence-vempty

lemmas [*V-cs-simps*] =
vfinite-0-*left*
vfinite-0-*right*

Binary relation as a finite sequence

lemmas [*V-cs-simps*] =
fconverse-vunion
fconverse-ftimes
vdomain-fflip

lemmas [*V-cs-intros*] =
*ftimesI*2
vcpower-two-ftimesI

Ordinals

lemmas [*Ord-cs-intros*] =
Limit-right-Limit-mult
Limit-left-Limit-mult
Ord-succ-mono
Limit-plus-omega-vsubset-Limit
Limit-plus-nat-in-Limit

von Neumann hierarchy

lemma (**in** $\mathcal{Z}$) *omega-in-any*[*V-cs-intros*]:
assumes $\alpha \subseteq_\circ \beta$
shows $\omega \in_\circ \beta$
⟨*proof*⟩

lemma *Ord-vsubset-succ*[*V-cs-intros*]:
assumes *Ord* α **and** *Ord* β **and** $\alpha \subseteq_\circ \beta$
shows $\alpha \subseteq_\circ$ *succ* β
⟨*proof*⟩

lemma *Ord-in-Vset-succ*[*V-cs-intros*]:
assumes *Ord* α **and** $a \in_\circ$ *Vset* α
shows $a \in_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemma *Ord-vsubset-Vset-succ*[*V-cs-intros*]:
assumes *Ord* α **and** $B \subseteq_\circ$ *Vset* α
shows $B \subseteq_\circ$ *Vset* (*succ* α)
⟨*proof*⟩

lemmas (**in** $\mathcal{Z}$) [*V-cs-intros*] =
omega-in-α
Ord-α
Limit-α

lemmas [*V-cs-intros*] =
vempty-in-Vset-succ
$\mathcal{Z}$*.ord-of-nat-in-Vset*
Vset-in-mono
Limit-vpair-in-VsetI
Vset-vsubset-mono
Ord-succ
Limit-vempty-in-VsetI
Limit-insert-in-VsetI
vfsequence.vfsequence-Limit-vcons-in-VsetI
vfsequence.vfsequence-Ord-vcons-in-Vset-succI
Limit-vdoubleton-in-VsetI
Limit-omega-in-VsetI
Limit-ftimes-in-VsetI

n-ary operations

lemmas [*V-cs-simps*] =
fflip-app
vdomain-fflip

Countable ordinals as a set

named-theorems *omega-of-set*
named-theorems *nat-omega-simps-extra*

lemmas [*nat-omega-simps-extra*] =
add-num-simps
Suc-numeral
Suc-1
le-num-simps
less-numeral-simps(1,2)
less-num-simps
less-one
nat-omega-simps

lemmas [*omega-of-set*] = *nat-omega-simps-extra*

lemma *set-insert-succ*[*omega-of-set*]:
assumes [*simp*]: *small b* **and** *set b* = $a_{\mathbb{N}}$
shows *set* (*insert* ($a_{\mathbb{N}}$) *b*) = *succ* ($a_{\mathbb{N}}$)
⟨*proof*⟩

lemma *set-0*[*omega-of-set*]: *set* {0} = *succ* 0 ⟨*proof*⟩

Sequences

named-theorems *vfsequence-simps*

named-theorems *vfsequence-intros*

lemmas [*vfsequence-simps*] =
vfsequence.vfsequence-at-last[*rotated*]
vfsequence.vfsequence-vcard-vcons[*rotated*]
vfsequence.vfsequence-at-not-last[*rotated*]

lemmas [*vfsequence-intros*] =
vfsequence.vfsequence-vcons
vfsequence-vempty

Further numerals

named-theorems *nat-omega-intros*

lemma [*nat-omega-intros*]:
assumes $a < b$
shows $a_{\mathbb{N}} \in_{\circ} b_{\mathbb{N}}$
⟨*proof*⟩

lemma [*nat-omega-intros*]:
assumes $0 < b$
shows $0 \in_{\circ} b_{\mathbb{N}}$
⟨*proof*⟩

lemma [*nat-omega-intros*]:
assumes $a = \mathit{numeral}\ b$
shows $(0{::}\mathit{nat}) < a$
⟨*proof*⟩

lemma *nat-le-if-in*[*nat-omega-intros*]:
assumes $x_{\mathbb{N}} \in_{\circ} y_{\mathbb{N}}$
shows $x_{\mathbb{N}} \leq y_{\mathbb{N}}$
⟨*proof*⟩

lemma *vempty-le-nat*[*nat-omega-intros*]: $0 \leq y_{\mathbb{N}}$ ⟨*proof*⟩

lemmas [*nat-omega-intros*] =
preorder-class.order-refl
preorder-class.eq-refl

Generally available foundational results

lemma (in $\mathcal{Z}$**)** $\mathcal{Z}$*-β*:
assumes $\beta = \alpha$
shows $\mathcal{Z}\ \beta$
⟨*proof*⟩

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

3.2 Digraph

3.2.1 Background

named-theorems *dg-field-simps*

definition *Obj* :: *V* **where** [*dg-field-simps*]: *Obj* = 0
definition *Arr* :: *V* **where** [*dg-field-simps*]: *Arr* = $1_{\mathbb{N}}$
definition *Dom* :: *V* **where** [*dg-field-simps*]: *Dom* = $2_{\mathbb{N}}$
definition *Cod* :: *V* **where** [*dg-field-simps*]: *Cod* = $3_{\mathbb{N}}$

3.2.2 Arrow with a domain and a codomain

The definition of and notation for an arrow with a domain and codomain is adapted from Chapter I-1 in [39]. The definition is applicable to digraphs and all other relevant derived entities, such as semicategories and categories, that are presented in the subsequent chapters.

In this work, by convention, the definition of an arrow with a domain and a codomain is nearly always preferred to the explicit use of the domain and codomain functions for the specification of the fundamental properties of arrows. Thus, to say that f is an arrow with the domain a, it is preferable to write $f : a \mapsto_{\mathfrak{C}} b$ (b can be assumed to be arbitrary) instead of $f \in_{\circ} \mathfrak{C}(\!|Arr|\!)$ and $\mathfrak{C}(\!|Dom|\!)(\!|f|\!) = a$.

definition *is-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  where *is-arr* $\mathfrak{C}$ *a b f* $\longleftrightarrow$ $f \in_{\circ} \mathfrak{C}(\!|Arr|\!) \wedge \mathfrak{C}(\!|Dom|\!)(\!|f|\!) = a \wedge \mathfrak{C}(\!|Cod|\!)(\!|f|\!) = b$

syntax *-is-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool* ($\langle$- : - $\mapsto_1$ -$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-arr* $\rightleftharpoons$ *is-arr*
translations $f : a \mapsto_{\mathfrak{C}} b \rightleftharpoons$ *CONST is-arr* $\mathfrak{C}$ *a b f*

Rules.

mk-ide *is-arr-def*
  |*intro is-arrI*|
  |*dest is-arrD*[*dest*]|
  |*elim is-arrE*[*elim*]|

lemmas [*dg-shared-cs-intros*, *dg-cs-intros*] = *is-arrD*(1)
lemmas [*dg-shared-cs-simps*, *dg-cs-simps*] = *is-arrD*(2,3)

3.2.3 *Hom*-set

See Chapter I-8 in [39].

abbreviation *Hom* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *Hom* $\mathfrak{C}$ *a b* $\equiv$ *set* $\{f.\ f : a \mapsto_{\mathfrak{C}} b\}$

lemma *small-Hom*[*simp*]: *small* $\{f.\ f : a \mapsto_{\mathfrak{C}} b\}$ $\langle$*proof*$\rangle$

Rules.

lemma *HomI*[*dg-shared-cs-intros*, *dg-cs-intros*]:
  assumes $f : a \mapsto_{\mathfrak{C}} b$
  shows $f \in_{\circ}$ *Hom* $\mathfrak{C}$ *a b*
  $\langle$*proof*$\rangle$

lemma *in-Hom-iff*[*dg-shared-cs-simps*, *dg-cs-simps*]:
  $f \in_{\circ}$ *Hom* $\mathfrak{C}$ *a b* $\longleftrightarrow$ $f : a \mapsto_{\mathfrak{C}} b$
  $\langle$*proof*$\rangle$

The *Hom*-sets in a given digraph are pairwise disjoint. This property was exposed as Axiom (v) in an alternative definition of a category presented in Chapter I-8 in [39]. Within the scope of the definitional framework employed in this study, this property holds unconditionally.

lemma *Hom-vdisjnt*:
assumes $a \neq a' \vee b \neq b'$
and $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $a' \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $b \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
and $b' \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
shows *vdisjnt* (*Hom* $\mathfrak{C}$ a b) (*Hom* $\mathfrak{C}$ a' b')
⟨*proof*⟩

3.2.4 Digraph: background information

The definition of a digraph that is employed in this work is similar to the definition of a *directed graph* presented in Chapter I-2 in [39]. However, there are notable differences. More specifically, the definition is parameterized by a limit ordinal α, such that $\omega < \alpha$; the set of objects is assumed to be a subset of the set V_α in the von Neumann hierarchy of sets (e.g., see [59]). Such digraphs are called *α-digraphs* to make the dependence on the parameter α explicit.[1] This definition was inspired by the ideas expressed in [28], [52] and [57].

In ZFC in HOL, the predicate *small* is used for distinguishing the terms of any type of the form *'a set* that are isomorphic to elements of a term of the type V (the elements can be exposed via the predicate *elts*). Thus, the collection of the elements associated with any term of the type V (e.g., *elts a*) is always small (see the theorem *small-elts* in [51]). Therefore, in this study, in an attempt to avoid confusion, the term "small" is never used to refer to digraphs. Instead, a new terminology is introduced in this body of work.

Thus, in this work, an α-digraph is a tiny α-digraph if and only if the set of its objects and the set of its arrows both belong to the set V_α. This notion is similar to the notion of a small category in the sense of the definition employed in Chapter I-6 in [39], if it is assumed that the "smallness" is determined with respect to the set V_α instead of the universe U. Also, in what follows, any member of the set V_α will be referred to as an α-tiny set.

All of the large (i.e. non-tiny) digraphs that are considered within the scope of this work have a slightly unconventional condition associated with the size of their *Hom*-sets. This condition implies that all *Hom*-sets of a digraph are tiny, but it is not equivalent to all *Hom*-sets being tiny. The condition was introduced in an attempt to resolve some of the issues related to the lack of an analogue of the Axiom Schema of Replacement closed with respect to V_α.

3.2.5 Digraph: definition and elementary properties

locale *digraph* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *Dom*: *vsv* ‹$\mathfrak{C}(\!|Dom|\!)$› + *Cod*: *vsv* ‹$\mathfrak{C}(\!|Cod|\!)$›
for α $\mathfrak{C}$ +
assumes *dg-length*[*dg-cs-simps*]: *vcard* $\mathfrak{C} = 4_{\mathbb{N}}$
and *dg-Dom-vdomain*[*dg-cs-simps*]: $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|Dom|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and *dg-Dom-vrange*: $\mathcal{R}_{\circ}$ ($\mathfrak{C}(\!|Dom|\!)$) $\subseteq_{\circ}$ $\mathfrak{C}(\!|Obj|\!)$
and *dg-Cod-vdomain*[*dg-cs-simps*]: $\mathcal{D}_{\circ}$ ($\mathfrak{C}(\!|Cod|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and *dg-Cod-vrange*: $\mathcal{R}_{\circ}$ ($\mathfrak{C}(\!|Cod|\!)$) $\subseteq_{\circ}$ $\mathfrak{C}(\!|Obj|\!)$
and *dg-Obj-vsubset-Vset*: $\mathfrak{C}(\!|Obj|\!)$ $\subseteq_{\circ}$ *Vset* α
and *dg-Hom-vifunion-in-Vset*[*dg-cs-intros*]:
[[$A \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$; $B \subseteq_{\circ} \mathfrak{C}(\!|Obj|\!)$; $A \in_{\circ}$ *Vset* α; $B \in_{\circ}$ *Vset* α]] ⟹
($\bigcup_{\circ} a \in_{\circ} A.\ \bigcup_{\circ} b \in_{\circ} B.$ *Hom* $\mathfrak{C}$ a b) $\in_{\circ}$ *Vset* α

[1] The prefix "α-" may be omitted whenever it is possible to infer the value of α from the context. This applies not only to the digraphs, but all other entities that are parameterized by a limit ordinal α such that $\omega < \alpha$.

lemmas [*dg-cs-simps*] =
digraph.dg-length
digraph.dg-Dom-vdomain
digraph.dg-Cod-vdomain

lemmas [*dg-cs-intros*] =
digraph.dg-Hom-vifunion-in-Vset

Rules.

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

mk-ide rf *digraph-def*[*unfolded digraph-axioms-def*]
|*intro digraphI*|
|*dest digraphD*[*dest*]|
|*elim digraphE*[*elim*]|

Elementary properties.

lemma *dg-eqI*:
assumes *digraph* α $\mathfrak{A}$
and *digraph* 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 $\mathfrak{A}(\!|Cod|\!) = \mathfrak{B}(\!|Cod|\!)$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

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

lemma (in *digraph*) *dg-Obj-if-Dom-vrange*:
assumes $a \in_\circ \mathcal{R}_\circ (\mathfrak{C}(\!|Dom|\!))$
shows $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma (in *digraph*) *dg-Obj-if-Cod-vrange*:
assumes $a \in_\circ \mathcal{R}_\circ (\mathfrak{C}(\!|Cod|\!))$
shows $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma (in *digraph*) *dg-is-arrD*:
assumes $f : a \mapsto_{\mathfrak{C}} b$
shows $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{C}(\!|Dom|\!)(\!|f|\!) = a$
and $\mathfrak{C}(\!|Cod|\!)(\!|f|\!) = b$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *digraph.dg-is-arrD*(1–3)

lemma (in *digraph*) *dg-is-arrE*[*elim*]:
assumes $f : a \mapsto_{\mathfrak{C}} b$
obtains $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
and $a \in_\circ \mathfrak{C}(\!|Obj|\!)$

and $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
and $\mathfrak{C}(\!|Dom|\!)(\!|f|\!) = a$
and $\mathfrak{C}(\!|Cod|\!)(\!|f|\!) = b$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-in-ArrE*[*elim*]:
assumes $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
obtains $a\ b$ **where** $f : a \mapsto_{\mathfrak{C}} b$ **and** $a \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Hom-in-Vset*[*dg-cs-intros*]:
assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $Hom\ \mathfrak{C}\ a\ b \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *digraph.dg-Hom-in-Vset*

Size.

lemma (**in** *digraph*) *dg-Arr-vsubset-Vset*: $\mathfrak{C}(\!|Arr|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Dom-vsubset-Vset*: $\mathfrak{C}(\!|Dom|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Cod-vsubset-Vset*: $\mathfrak{C}(\!|Cod|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

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

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

lemma (**in** *digraph*) *dg-in-Obj-in-Vset*[*dg-cs-intros*]:
assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $a \in_\circ Vset\ \alpha$
⟨*proof*⟩

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

lemma (**in** *digraph*) *dg-in-Arr-in-Vset*[*dg-cs-intros*]:
assumes $a \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $a \in_\circ Vset\ \alpha$
⟨*proof*⟩

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

lemma (**in** *digraph*) *dg-Cod-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $\mathfrak{C}(\!|Cod|\!) \in_\circ Vset\ \beta$

⟨*proof*⟩

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

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

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

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

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

Further properties.

lemma (**in** *digraph*) *dg-Dom-app-in-Obj*:
assumes $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $\mathfrak{C}(\!|Dom|\!)(\!|f|\!) \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Cod-app-in-Obj*:
assumes $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $\mathfrak{C}(\!|Cod|\!)(\!|f|\!) \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Arr-vempty-if-Obj-vempty*:
assumes $\mathfrak{C}(\!|Obj|\!) = 0$
shows $\mathfrak{C}(\!|Arr|\!) = 0$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Dom-vempty-if-Arr-vempty*:
assumes $\mathfrak{C}(\!|Arr|\!) = 0$
shows $\mathfrak{C}(\!|Dom|\!) = 0$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-Cod-vempty-if-Arr-vempty*:
assumes $\mathfrak{C}(\!|Arr|\!) = 0$
shows $\mathfrak{C}(\!|Cod|\!) = 0$
⟨*proof*⟩

3.2.6 Opposite digraph

Definition and elementary properties

See Chapter II-2 in [39].

definition *op-dg* :: $V \Rightarrow V$
  where *op-dg* $\mathfrak{C}$ = $[\mathfrak{C}(\!|Obj|\!), \mathfrak{C}(\!|Arr|\!), \mathfrak{C}(\!|Cod|\!), \mathfrak{C}(\!|Dom|\!)]_\circ$

Components.

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

lemma *op-dg-component-intros*[*dg-op-intros*]:
  shows $a \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow a \in_\circ$ *op-dg* $\mathfrak{C}(\!|Obj|\!)$
    and $f \in_\circ \mathfrak{C}(\!|Arr|\!) \Longrightarrow f \in_\circ$ *op-dg* $\mathfrak{C}(\!|Arr|\!)$
  ⟨*proof*⟩

Elementary properties.

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

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

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

Further properties

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

lemmas *digraph-op*[*dg-op-intros*] = *digraph.digraph-op*

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

lemmas *dg-op-dg-op-dg*[*dg-op-simps*] = *digraph.dg-op-dg-op-dg*

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

3.3 Smallness for digraphs

3.3.1 Background

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

3.3.2 Tiny digraph

Definition and elementary properties

locale *tiny-digraph* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *Dom*: *vsv* ‹$\mathfrak{C}(\!|Dom|\!)$› + *Cod*: *vsv* ‹$\mathfrak{C}(\!|Cod|\!)$›
for α $\mathfrak{C}$ +
assumes *tiny-dg-length*[*dg-cs-simps*]: *vcard* $\mathfrak{C}$ = $4_{\mathbb{N}}$
and *tiny-dg-Dom-vdomain*[*dg-cs-simps*]: $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and *tiny-dg-Dom-vrange*: $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Dom|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and *tiny-dg-Cod-vdomain*[*dg-cs-simps*]: $\mathcal{D}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) = $\mathfrak{C}(\!|Arr|\!)$
and *tiny-dg-Cod-vrange*: $\mathcal{R}_\circ$ ($\mathfrak{C}(\!|Cod|\!)$) $\subseteq_\circ$ $\mathfrak{C}(\!|Obj|\!)$
and *tiny-dg-Obj-in-Vset*[*dg-small-cs-intros*]: $\mathfrak{C}(\!|Obj|\!)$ $\in_\circ$ *Vset* α
and *tiny-dg-Arr-in-Vset*[*dg-small-cs-intros*]: $\mathfrak{C}(\!|Arr|\!)$ $\in_\circ$ *Vset* α

lemmas [*dg-small-cs-intros*] =
tiny-digraph.tiny-dg-Obj-in-Vset
tiny-digraph.tiny-dg-Arr-in-Vset

Rules.

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

mk-ide rf *tiny-digraph-def*[*unfolded tiny-digraph-axioms-def*]
|*intro tiny-digraphI*|
|*dest tiny-digraphD*[*dest*]|
|*elim tiny-digraphE*[*elim*]|

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

Elementary properties.

sublocale *tiny-digraph* $\subseteq$ *digraph*
⟨*proof*⟩

lemmas (**in** *tiny-digraph*) *tiny-dg-digraph* = *digraph-axioms*

lemmas [*dg-small-cs-intros*] = *tiny-digraph.tiny-dg-digraph*

Size.

lemma (**in** *tiny-digraph*) *tiny-dg-Dom-in-Vset*: $\mathfrak{C}(\!|Dom|\!)$ $\in_\circ$ *Vset* α
⟨*proof*⟩

lemma (**in** *tiny-digraph*) *tiny-dg-Cod-in-Vset*: $\mathfrak{C}(\!|Cod|\!)$ $\in_\circ$ *Vset* α
⟨*proof*⟩

lemma (**in** *tiny-digraph*) *tiny-dg-in-Vset*: $\mathfrak{C}$ $\in_\circ$ *Vset* α
⟨*proof*⟩

lemma *small-tiny-digraphs*[*simp*]: *small* {𝔇. *tiny-digraph* α 𝔇}
⟨*proof*⟩

lemma *tiny-digraphs-vsubset-Vset*: *set* {𝔇. *tiny-digraph* α 𝔇} $\subseteq_\circ$ *Vset* α
⟨*proof*⟩

lemma (**in** *digraph*) *dg-tiny-digraph-if-ge-Limit*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-digraph* β 𝔇
⟨*proof*⟩

Opposite tiny digraph

lemma (**in** *tiny-digraph*) *tiny-digraph-op*: *tiny-digraph* α (*op-dg* 𝔇)
⟨*proof*⟩

lemmas *tiny-digraph-op*[*dg-op-intros*] = *tiny-digraph.tiny-digraph-op*

3.3.3 Finite digraph

Definition and elementary properties

A finite digraph is a generalization of the concept of a finite category, as presented in nLab [3][2].

locale *finite-digraph* = *digraph* α 𝔇 **for** α 𝔇 +
assumes *fin-dg-Obj-vfinite*[*dg-small-cs-intros*]: *vfinite* (𝔇⦇*Obj*⦈)
and *fin-dg-Arr-vfinite*[*dg-small-cs-intros*]: *vfinite* (𝔇⦇*Arr*⦈)

lemmas [*dg-small-cs-intros*] =
finite-digraph.fin-dg-Obj-vfinite
finite-digraph.fin-dg-Arr-vfinite

Rules.

lemma (**in** *finite-digraph*) *finite-digraph-axioms'*[*dg-small-cs-intros*]:
assumes $\alpha' = \alpha$
shows *finite-digraph* α' 𝔇
⟨*proof*⟩

mk-ide rf *finite-digraph-def*[*unfolded finite-digraph-axioms-def*]
|*intro finite-digraphI*|
|*dest finite-digraphD*[*dest*]|
|*elim finite-digraphE*[*elim*]|

Elementary properties.

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

lemmas (**in** *finite-digraph*) *fin-dg-tiny-digraph* = *tiny-digraph-axioms*

lemmas [*dg-small-cs-intros*] = *finite-digraph.fin-dg-tiny-digraph*

Size.

lemma *small-finite-digraphs*[*simp*]: *small* {𝔇. *finite-digraph* α 𝔇}
⟨*proof*⟩

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

lemma *finite-digraphs-vsubset-Vset*: *set* $\{\mathfrak{C}.\ \textit{finite-digraph}\ \alpha\ \mathfrak{C}\} \subseteq_\circ \textit{Vset}\ \alpha$
⟨*proof*⟩

Opposite finite digraph

lemma (**in** *finite-digraph*) *fininte-digraph-op*: *finite-digraph* α (*op-dg* $\mathfrak{C}$)
⟨*proof*⟩

lemmas *fininte-digraph-op*[*dg-op-intros*] = *finite-digraph.fininte-digraph-op*

3.4 Homomorphism of digraphs

3.4.1 Background

named-theorems *dghm-cs-simps*
named-theorems *dghm-cs-intros*

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

named-theorems *dghm-field-simps*

definition *ObjMap* :: *V* **where** [*dghm-field-simps*]: *ObjMap* = 0
definition *ArrMap* :: *V* **where** [*dghm-field-simps*]: *ArrMap* = $1_{\mathbb{N}}$
definition *HomDom* :: *V* **where** [*dghm-field-simps*]: *HomDom* = $2_{\mathbb{N}}$
definition *HomCod* :: *V* **where** [*dghm-field-simps*]: *HomCod* = $3_{\mathbb{N}}$

3.4.2 Definition and elementary properties

A homomorphism of digraphs, as presented in this work, can be seen as a generalization of the concept of a functor between categories, as presented in Chapter I-3 in [39], to digraphs. The generalization is performed by removing the axioms (1) from the definition. It is expected that the resulting definition is consistent with the conventional notion of a homomorphism of digraphs in graph theory, but further details are considered to be outside of the scope of this work.

The definition of a digraph homomorphism is parameterized by a limit ordinal α such that $\omega < \alpha$. Such digraph homomorphisms are referred to either as α-digraph homomorphisms or homomorphisms of α-digraphs.

Following [39], all digraph homomorphisms are covariant (see Chapter II-2). However, a special notation is adapted for the digraph homomorphisms from an opposite digraph. Normally, such digraph homomorphisms will be referred to as the contravariant digraph homomorphisms, but this convention will not be enforced.

locale *is-dghm* =
  $\mathcal{Z}$ α + *vfsequence* $\mathfrak{F}$ + *HomDom*: *digraph* α $\mathfrak{A}$ + *HomCod*: *digraph* α $\mathfrak{B}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *dghm-length*[*dg-cs-simps*]: *vcard* $\mathfrak{F}$ = $4_{\mathbb{N}}$
    and *dghm-HomDom*[*dg-cs-simps*]: $\mathfrak{F}(\lfloor HomDom \rfloor) = \mathfrak{A}$
    and *dghm-HomCod*[*dg-cs-simps*]: $\mathfrak{F}(\lfloor HomCod \rfloor) = \mathfrak{B}$
    and *dghm-ObjMap-vsv*: *vsv* $(\mathfrak{F}(\lfloor ObjMap \rfloor))$
    and *dghm-ArrMap-vsv*: *vsv* $(\mathfrak{F}(\lfloor ArrMap \rfloor))$
    and *dghm-ObjMap-vdomain*[*dg-cs-simps*]: $\mathcal{D}_\circ\ (\mathfrak{F}(\lfloor ObjMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
    and *dghm-ObjMap-vrange*: $\mathcal{R}_\circ\ (\mathfrak{F}(\lfloor ObjMap \rfloor)) \subseteq_\circ \mathfrak{B}(\lfloor Obj \rfloor)$
    and *dghm-ArrMap-vdomain*[*dg-cs-simps*]: $\mathcal{D}_\circ\ (\mathfrak{F}(\lfloor ArrMap \rfloor)) = \mathfrak{A}(\lfloor Arr \rfloor)$
    and *dghm-ArrMap-is-arr*:
      $f : a \mapsto_{\mathfrak{A}} b \Longrightarrow \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) : \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor b \rfloor)$

syntax *-is-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-} \mapsto\mapsto_{DG1} \text{-})\rangle\ [51, 51, 51]\ 51)$
syntax-consts *-is-dghm* $\rightleftharpoons$ *is-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

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

syntax *-is-cn-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle(\text{-} :/ \text{-}\ _{DG}\!\mapsto\!\mapsto_1 \text{-})\rangle\ [51, 51, 51]\ 51)$

syntax-consts *-is-cn-dghm* ⇌ *is-cn-dghm*
translations $\mathfrak{F} : \mathfrak{A}\ {}_{DG}\mapsto\mapsto_{\alpha}\ \mathfrak{B} \rightharpoonup CONST\ is\text{-}cn\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

abbreviation *all-dghms* :: $V \Rightarrow V$
  where $all\text{-}dghms\ \alpha \equiv set\ \{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha}\ \mathfrak{B}\}$

abbreviation *dghms* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where $dghms\ \alpha\ \mathfrak{A}\ \mathfrak{B} \equiv set\ \{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha}\ \mathfrak{B}\}$

sublocale *is-dghm* ⊆ *ObjMap*: *vsv* ‹$\mathfrak{F}(\!|ObjMap|\!)$›
  rewrites $\mathcal{D}_{\circ}\ (\mathfrak{F}(\!|ObjMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
  ⟨*proof*⟩

sublocale *is-dghm* ⊆ *ArrMap*: *vsv* ‹$\mathfrak{F}(\!|ArrMap|\!)$›
  rewrites $\mathcal{D}_{\circ}\ (\mathfrak{F}(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
  ⟨*proof*⟩

lemmas [*dg-cs-simps*] =
  is-dghm.dghm-HomDom
  is-dghm.dghm-HomCod
  is-dghm.dghm-ObjMap-vdomain
  is-dghm.dghm-ArrMap-vdomain

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

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

lemmas [*dg-cs-intros*] = *is-dghm.dghm-ArrMap-is-arr'*

Rules.

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

mk-ide rf *is-dghm-def*[*unfolded is-dghm-axioms-def*]
  |*intro is-dghmI*|
  |*dest is-dghmD*[*dest*]|
  |*elim is-dghmE*[*elim*]|

lemmas [*dg-cs-intros*] = *is-dghmD*(3,4)

Elementary properties.

lemma *dghm-eqI*:
  assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha}\ \mathfrak{B}$
    and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{DG\alpha}\ \mathfrak{D}$
    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 (**in** *is-dghm*) *dghm-def*: $\mathfrak{F} = [\mathfrak{F}(\!|ObjMap|\!), \mathfrak{F}(\!|ArrMap|\!), \mathfrak{F}(\!|HomDom|\!), \mathfrak{F}(\!|HomCod|\!)]_\circ$
⟨*proof*⟩

lemma (**in** *is-dghm*) *dghm-ObjMap-app-in-HomCod-Obj*[*dg-cs-intros*]:
assumes $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \in_\circ \mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *is-dghm.dghm-ObjMap-app-in-HomCod-Obj*

lemma (**in** *is-dghm*) *dghm-ArrMap-vrange*: $\mathcal{R}_\circ\ (\mathfrak{F}(\!|ArrMap|\!)) \subseteq_\circ \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemma (**in** *is-dghm*) *dghm-ArrMap-app-in-HomCod-Arr*[*dg-cs-intros*]:
assumes $a \in_\circ \mathfrak{A}(\!|Arr|\!)$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|a|\!) \in_\circ \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *is-dghm.dghm-ArrMap-app-in-HomCod-Arr*

Size.

lemma (**in** *is-dghm*) *dghm-ObjMap-vsubset-Vset*: $\mathfrak{F}(\!|ObjMap|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-dghm*) *dghm-ArrMap-vsubset-Vset*: $\mathfrak{F}(\!|ArrMap|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *is-dghm*) *dghm-ObjMap-in-Vset*:
assumes $\alpha \in_\circ \beta$
shows $\mathfrak{F}(\!|ObjMap|\!) \in_\circ Vset\ \beta$
⟨*proof*⟩

lemma (**in** *is-dghm*) *dghm-ArrMap-in-Vset*:
assumes $\alpha \in_\circ \beta$
shows $\mathfrak{F}(\!|ArrMap|\!) \in_\circ Vset\ \beta$
⟨*proof*⟩

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

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

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

lemma (**in** *is-dghm*) *dghm-in-Vset-7*: $\mathfrak{F} \in_\circ Vset\ (\alpha + 7_{\mathbb{N}})$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *all-dghms-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$

shows *all-dghms* $\alpha \in_\circ$ *Vset* β
⟨*proof*⟩

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

Further properties.

lemma (in *is-dghm*) *dghm-is-arr-HomCod*:
assumes $f : a \mapsto_{\mathfrak{A}} b$
shows $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) \in_\circ \mathfrak{B}(\!|Arr|\!)\ \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \in_\circ \mathfrak{B}(\!|Obj|\!)\ \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!) \in_\circ \mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

lemma (in *is-dghm*) *dghm-vimage-dghm-ArrMap-vsubset-Hom*:
assumes $a \in_\circ \mathfrak{A}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $\mathfrak{F}(\!|ArrMap|\!)$ ‘$_\circ$ *Hom* $\mathfrak{A}$ a $b \subseteq_\circ$ *Hom* $\mathfrak{B}$ $(\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))$ $(\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$
⟨*proof*⟩

3.4.3 Opposite digraph homomorphism

Definition and elementary properties

See Chapter II-2 in [39].

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

Components.

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

Further properties

lemma (in *is-dghm*) *is-dghm-op*: *op-dghm* $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-dghm*) *is-dghm-op'*[*dg-op-intros*]:
assumes $\mathfrak{A}' = op\text{-}dg\ \mathfrak{A}$ **and** $\mathfrak{B}' = op\text{-}dg\ \mathfrak{B}$ **and** $\alpha' = \alpha$
shows *op-dghm* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{DG\alpha'} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-dghm-op*[*dg-op-intros*] = *is-dghm.is-dghm-op'*

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

lemmas *dghm-op-dghm-op-dghm*[*dg-op-simps*] = *is-dghm.dghm-op-dghm-op-dghm*

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

3.4.4 Composition of covariant digraph homomorphisms

Definition and elementary properties

See Chapter I-3 in [39].

definition *dghm-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\circ_{DGHM}\rangle$ 55)
where $\mathfrak{G} \circ_{DGHM} \mathfrak{F} =$
$[\mathfrak{G}(\!|ObjMap|\!) \circ_{\circ} \mathfrak{F}(\!|ObjMap|\!), \mathfrak{G}(\!|ArrMap|\!) \circ_{\circ} \mathfrak{F}(\!|ArrMap|\!), \mathfrak{F}(\!|HomDom|\!), \mathfrak{G}(\!|HomCod|\!)]_{\circ}$

Components.

lemma *dghm-comp-components*:
shows $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\!|ObjMap|\!) = \mathfrak{G}(\!|ObjMap|\!) \circ_{\circ} \mathfrak{F}(\!|ObjMap|\!)$
and $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\!|ArrMap|\!) = \mathfrak{G}(\!|ArrMap|\!) \circ_{\circ} \mathfrak{F}(\!|ArrMap|\!)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]: $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\!|HomDom|\!) = \mathfrak{F}(\!|HomDom|\!)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]: $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\!|HomCod|\!) = \mathfrak{G}(\!|HomCod|\!)$
⟨*proof*⟩

Object map

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

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

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

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

Arrow map

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

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

lemma *dghm-comp-ArrMap-vrange*[*dg-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathcal{R}_{\circ}((\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\!|ArrMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemma *dghm-comp-ArrMap-app*[*dg-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$ **and** $f \in_{\circ} \mathfrak{A}(\!|Arr|\!)$

shows $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \rfloor)$
⟨*proof*⟩

Opposite of the composition of covariant digraph homomorphisms

lemma *op-dghm-dghm-comp*[*dg-op-simps*]:
op-dghm $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})$ = *op-dghm* $\mathfrak{G} \circ_{DGHM}$ *op-dghm* $\mathfrak{F}$
⟨*proof*⟩

Further properties

lemma *dghm-comp-is-dghm*[*dg-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

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

3.4.5 Composition of contravariant digraph homomorphisms

Definition and elementary properties

See section 1.2 in [15].

definition *dghm-cn-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle_{DGHM^{\circ}}\rangle$ 55)
where $\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F} =$
$[$
$\mathfrak{G}(\lfloor ObjMap \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap \rfloor)$,
$\mathfrak{G}(\lfloor ArrMap \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ArrMap \rfloor)$,
op-dg $(\mathfrak{F}(\lfloor HomDom \rfloor))$,
$\mathfrak{G}(\lfloor HomCod \rfloor)$
$]_{\circ}$

Components.

lemma *dghm-cn-comp-components*:
shows $(\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor ObjMap \rfloor) = \mathfrak{G}(\lfloor ObjMap \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap \rfloor)$
and $(\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor ArrMap \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ArrMap \rfloor)$
and [*dg-cn-cs-simps*]: $(\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor HomDom \rfloor) =$ *op-dg* $(\mathfrak{F}(\lfloor HomDom \rfloor))$
and [*dg-cn-cs-simps*]: $(\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor HomCod \rfloor) = \mathfrak{G}(\lfloor HomCod \rfloor)$
⟨*proof*⟩

Object map: two contravariant digraph homomorphisms

lemma *dghm-cn-comp-ObjMap-vsv*[*dg-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor ObjMap \rfloor))$
⟨*proof*⟩

lemma *dghm-cn-comp-ObjMap-vdomain*[*dg-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}((\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor ObjMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *dghm-cn-comp-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {}_{DG}{\mapsto\mapsto}_{\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}((\mathfrak{G}\ {}_{DGHM^{\circ}}\ \mathfrak{F})(\lfloor ObjMap \rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Obj \rfloor)$

⟨*proof*⟩

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

Arrow map: two contravariant digraph homomorphisms

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

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

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

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

Object map: contravariant and covariant digraph homomorphisms

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

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

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

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

Arrow map: contravariant and covariant digraph homomorphisms

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

lemma *dghm-cn-cov-comp-ArrMap-vdomain*[*dg-cn-cs-simps*]:

assumes $\mathfrak{G} : \mathfrak{B}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ _{DGHM}\!\circ\ \mathfrak{F})(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$
⟨*proof*⟩

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

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

Opposite of the contravariant composition of the digraph homomorphisms

lemma *op-dghm-dghm-cn-comp*[*dg-op-simps*]:
op-dghm $(\mathfrak{G}\ _{DGHM}\!\circ\ \mathfrak{F})$ = *op-dghm* $\mathfrak{G}\ _{DGHM}\!\circ$ *op-dghm* $\mathfrak{F}$
⟨*proof*⟩

Further properties

lemma *dghm-cn-comp-is-dghm*[*dg-cn-cs-intros*]:
— See section 1.2 in [15].
assumes *digraph* α $\mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{B}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathfrak{G}\ _{DGHM}\!\circ\ \mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha}\ \mathfrak{C}$
⟨*proof*⟩

lemma *dghm-cn-cov-comp-is-dghm*[*dg-cn-cs-intros*]:
— See section 1.2 in [15].
assumes $\mathfrak{G} : \mathfrak{B}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha}\ \mathfrak{B}$
shows $\mathfrak{G}\ _{DGHM}\!\circ\ \mathfrak{F} : \mathfrak{A}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{C}$
⟨*proof*⟩

lemma *dghm-cov-cn-comp-is-dghm*:
— See section 1.2 in [15]
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\!\!\mapsto_{DG\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathfrak{G} \circ_{DGHM}\ \mathfrak{F} : \mathfrak{A}\ _{DG}\!\mapsto\!\!\mapsto_{\alpha}\ \mathfrak{C}$
⟨*proof*⟩

3.4.6 Identity digraph homomorphism

Definition and elementary properties

See Chapter I-3 in [39].

definition *dghm-id* :: $V \Rightarrow V$
where *dghm-id* $\mathfrak{C}$ = [*vid-on* $(\mathfrak{C}(\!|Obj|\!))$, *vid-on* $(\mathfrak{C}(\!|Arr|\!))$, $\mathfrak{C}$, $\mathfrak{C}$]$_{\circ}$

Components.

lemma *dghm-id-components*:
shows *dghm-id* $\mathfrak{C}(\!|ObjMap|\!)$ = *vid-on* $(\mathfrak{C}(\!|Obj|\!))$
and *dghm-id* $\mathfrak{C}(\!|ArrMap|\!)$ = *vid-on* $(\mathfrak{C}(\!|Arr|\!))$
and [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-id* $\mathfrak{C}(\!|HomDom|\!) = \mathfrak{C}$
and [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-id* $\mathfrak{C}(\!|HomCod|\!) = \mathfrak{C}$
⟨*proof*⟩

Object map

mk-VLambda *dghm-id-components*(1)[*folded VLambda-vid-on*]

|*vsv dghm-id-ObjMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|
|*vdomain dghm-id-ObjMap-vdomain*[*dg-shared-cs-simps*, *dg-cs-simps*]|
|*app dghm-id-ObjMap-app*[*dg-shared-cs-simps*, *dg-cs-simps*]|

lemma *dghm-id-ObjMap-vrange*[*dg-shared-cs-simps*, *dg-cs-simps*]:
$\mathcal{R}_{\circ}$ (*dghm-id* $\mathfrak{C}$(|*ObjMap*|)) = $\mathfrak{C}$(|*Obj*|)
⟨*proof*⟩

Arrow map

mk-VLambda *dghm-id-components*(2)[*folded VLambda-vid-on*]
|*vsv dghm-id-ArrMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|
|*vdomain dghm-id-ArrMap-vdomain*[*dg-shared-cs-simps*, *dg-cs-simps*]|
|*app dghm-id-ArrMap-app*[*dg-shared-cs-simps*, *dg-cs-simps*]|

lemma *dghm-id-ArrMap-vrange*[*dg-shared-cs-simps*, *dg-cs-simps*]:
$\mathcal{R}_{\circ}$ (*dghm-id* $\mathfrak{C}$(|*ArrMap*|)) = $\mathfrak{C}$(|*Arr*|)
⟨*proof*⟩

Opposite identity digraph homomorphism

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

An identity digraph homomorphism is a digraph homomorphism

lemma (in *digraph*) *dg-dghm-id-is-dghm*: *dghm-id* $\mathfrak{C}$: $\mathfrak{C}$ $\mapsto\mapsto_{DG\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (in *digraph*) *dg-dghm-id-is-dghm′*:
assumes $\mathfrak{A}$ = $\mathfrak{C}$ **and** $\mathfrak{B}$ = $\mathfrak{C}$
shows *dghm-id* $\mathfrak{C}$: $\mathfrak{A}$ $\mapsto\mapsto_{DG\alpha}$ $\mathfrak{B}$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *digraph.dg-dghm-id-is-dghm′*

Further properties

lemma (in *is-dghm*) *dghm-dghm-comp-dghm-id-left*: *dghm-id* $\mathfrak{B}$ $\circ_{DGHM}$ $\mathfrak{F}$ = $\mathfrak{F}$
— See Chapter I-3 in [39]).
⟨*proof*⟩

lemmas [*dg-cs-simps*] = *is-dghm.dghm-dghm-comp-dghm-id-left*

lemma (in *is-dghm*) *dghm-dghm-comp-dghm-id-right*: $\mathfrak{F}$ $\circ_{DGHM}$ *dghm-id* $\mathfrak{A}$ = $\mathfrak{F}$
— See Chapter I-3 in [39]).
⟨*proof*⟩

lemmas [*dg-cs-simps*] = *is-dghm.dghm-dghm-comp-dghm-id-right*

3.4.7 Constant digraph homomorphism

Definition and elementary properties

See Chapter III-3 in [39].

definition *dghm-const* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f* =
[*vconst-on* ($\mathfrak{C}$(|*Obj*|)) *a*, *vconst-on* ($\mathfrak{C}$(|*Arr*|)) *f*, $\mathfrak{C}$, $\mathfrak{D}$]$_{\circ}$

Components.

lemma *dghm-const-components*:
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*(|*ObjMap*|) = *vconst-on* ($\mathfrak{C}$(|*Obj*|)) *a*
and *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*(|*ArrMap*|) = *vconst-on* ($\mathfrak{C}$(|*Arr*|)) *f*
and [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*(|*HomDom*|) = $\mathfrak{C}$
and [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*(|*HomCod*|) = $\mathfrak{D}$
⟨*proof*⟩

Object map

mk-VLambda *dghm-const-components*(1)[*folded VLambda-vconst-on*]
|*vsv dghm-const-ObjMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|
|*vdomain dghm-const-ObjMap-vdomain*[*dg-shared-cs-simps*, *dg-cs-simps*]|
|*app dghm-const-ObjMap-app*[*dg-shared-cs-simps*, *dg-cs-simps*]|

Arrow map

mk-VLambda *dghm-const-components*(2)[*folded VLambda-vconst-on*]
|*vsv dghm-const-ArrMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|
|*vdomain dghm-const-ArrMap-vdomain*[*dg-shared-cs-simps*, *dg-cs-simps*]|
|*app dghm-const-ArrMap-app*[*dg-shared-cs-simps*, *dg-cs-simps*]|

Opposite constant digraph homomorphism

lemma *op-dghm-dghm-const*[*dg-op-simps*]:
op-dghm (*dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*) = *dghm-const* (*op-dg* $\mathfrak{C}$) (*op-dg* $\mathfrak{D}$) *a f*
⟨*proof*⟩

A constant digraph homomorphism is a digraph homomorphism

lemma *dghm-const-is-dghm*:
assumes *digraph* α $\mathfrak{C}$ **and** *digraph* α $\mathfrak{D}$ **and** $f : a \mapsto_{\mathfrak{D}} a$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f* : $\mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *dghm-const-is-dghm'*[*dg-cs-intros*]:
assumes *digraph* α $\mathfrak{C}$
and *digraph* α $\mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} a$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f* : $\mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

Further properties

lemma (**in** *is-dghm*) *dghm-dghm-comp-dghm-const*[*dg-cs-simps*]:
assumes *digraph* α $\mathfrak{C}$ **and** $f : a \mapsto_{\mathfrak{C}} a$
shows *dghm-const* $\mathfrak{B}$ $\mathfrak{C}$ *a f* $\circ_{DGHM}$ $\mathfrak{F}$ = *dghm-const* $\mathfrak{A}$ $\mathfrak{C}$ *a f*
⟨*proof*⟩

lemmas [*dg-cs-simps*] = *is-dghm.dghm-dghm-comp-dghm-const*

3.4.8 Faithful digraph homomorphism

Definition and elementary properties

See Chapter I-3 in [39]).

locale *is-ft-dghm* = *is-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *ft-dghm-v11-on-Hom*:
    $[\![\ a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)\ ]\!] \Longrightarrow v11\ (\mathfrak{F}(\!|ArrMap|\!) \restriction^l_\circ Hom\ \mathfrak{A}\ a\ b)$

syntax *-is-ft-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle (\text{-} :/ \text{-} \mapsto\mapsto_{DG.faithful^1} \text{-})\rangle\ [51, 51, 51]\ 51)$
syntax-consts *-is-ft-dghm* $\rightleftharpoons$ *is-ft-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}ft\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

Rules.

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

mk-ide rf *is-ft-dghm-def*[*unfolded is-ft-dghm-axioms-def*]
  |*intro is-ft-dghmI*|
  |*dest is-ft-dghmD*[*dest*]|
  |*elim is-ft-dghmE*[*elim*]|

lemmas [*dghm-cs-intros*] = *is-ft-dghmD*(1)

lemma *is-ft-dghmI''*:
  assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
    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 $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{B}$
⟨*proof*⟩

Opposite faithful digraph homomorphism

lemma (in *is-ft-dghm*) *ft-dghm-op-dghm-is-ft-dghm*:
  op-dghm $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.faithful\alpha}$ *op-dg* $\mathfrak{B}$
  ⟨*proof*⟩

lemma (in *is-ft-dghm*) *ft-dghm-op-dghm-is-ft-dghm'*[*dg-op-intros*]:
  assumes $\mathfrak{A}'$ = *op-dg* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-dg* $\mathfrak{B}$
  shows *op-dghm* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{B}'$
  ⟨*proof*⟩

lemmas *ft-dghm-op-dghm-is-ft-dghm*[*dg-op-intros*] =
  is-ft-dghm.ft-dghm-op-dghm-is-ft-dghm'

The composition of faithful digraph homomorphisms is a faithful digraph homomorphism.

lemma *dghm-comp-is-ft-dghm*[*dghm-cs-intros*]:
  — See Chapter I-3 in [39].
  assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{B}$
  shows $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma (in *is-ft-dghm*) *ft-dghm-ArrMap-eqD*:
  assumes $g : a \mapsto_{\mathfrak{A}} b$ **and** $f : a \mapsto_{\mathfrak{A}} b$ **and** $\mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)$
  shows $g = f$
⟨*proof*⟩

3.4.9 Full digraph homomorphism

Definition and elementary properties

See Chapter I-3 in [39].

locale *is-fl-dghm* = *is-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *fl-dghm-surj-on-Hom*:
$[\![a \in_\circ \mathfrak{A}(\!|Obj|\!);\ b \in_\circ \mathfrak{A}(\!|Obj|\!)]\!] \Longrightarrow$
$\mathfrak{F}(\!|ArrMap|\!)\ `_\circ\ (Hom\ \mathfrak{A}\ a\ b) = Hom\ \mathfrak{B}\ (\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))\ (\mathfrak{F}(\!|ObjMap|\!)(\!|b|\!))$

syntax *-is-fl-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
$(\langle (-:/ - \mapsto\mapsto_{DG.full^1} -)\rangle\ [51, 51, 51]\ 51)$
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.full_\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}fl\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

Rules.

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

mk-ide rf *is-fl-dghm-def*[*unfolded is-fl-dghm-axioms-def*]
|*intro is-fl-dghmI*|
|*dest is-fl-dghmD*[*dest*]|
|*elim is-fl-dghmE*[*elim*]|

lemmas [*dghm-cs-intros*] = *is-fl-dghmD*(1)

Opposite full digraph homomorphism

lemma (**in** *is-fl-dghm*) *fl-dghm-op-dghm-is-fl-dghm*:
op-dghm $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.full_\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-fl-dghm*) *fl-dghm-op-dghm-is-fl-dghm'*[*dg-op-intros*]:
assumes $\mathfrak{A}' =$ *op-dg* $\mathfrak{A}$ **and** $\mathfrak{B}' =$ *op-dg* $\mathfrak{B}$
shows *op-dghm* $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.full_\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemmas *fl-dghm-op-dghm-is-fl-dghm*[*dg-op-intros*] =
is-fl-dghm.fl-dghm-op-dghm-is-fl-dghm'

The composition of full digraph homomorphisms is a full digraph homomorphism

lemma *dghm-comp-is-fl-dghm*[*dghm-cs-intros*]:
— See Chapter I-3 in [39].
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG.full_\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.full_\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.full_\alpha} \mathfrak{C}$
⟨*proof*⟩

3.4.10 Fully faithful digraph homomorphism

Definition and elementary properties

See Chapter I-3 in [39].

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

syntax *-is-ff-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
$(\langle (-:/ - \mapsto\mapsto_{DG.ff^1} -)\rangle\ [51, 51, 51]\ 51)$

syntax-consts *-is-ff-dghm* ⇌ *is-ff-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.ff\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}ff\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

Rules.

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

mk-ide rf *is-ff-dghm-def*
|*intro is-ff-dghmI*|
|*dest is-ff-dghmD*[*dest*]|
|*elim is-ff-dghmE*[*elim*]|

lemmas [*dghm-cs-intros*] = *is-ff-dghmD*

Opposite fully faithful digraph homomorphism.

lemma (in *is-ff-dghm*) *ff-dghm-op-dghm-is-ff-dghm*:
op-dghm $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.ff\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

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

lemmas *ff-dghm-op-dghm-is-ff-dghm*[*dg-op-intros*] =
is-ff-dghm.ff-dghm-op-dghm-is-ff-dghm

The composition of fully faithful digraph homomorphisms is a fully faithful digraph homomorphism.

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

3.4.11 Isomorphism of digraphs

Definition and elementary properties

See Chapter I-3 in [39].

locale *is-iso-dghm* = *is-dghm* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ **for** $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ +
assumes *iso-dghm-ObjMap-v11*: *v11* ($\mathfrak{F}(\lfloor ObjMap \rfloor)$)
and *iso-dghm-ArrMap-v11*: *v11* ($\mathfrak{F}(\lfloor ArrMap \rfloor)$)
and *iso-dghm-ObjMap-vrange*[*dghm-cs-simps*]: $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor ObjMap \rfloor)$) = $\mathfrak{B}(\lfloor Obj \rfloor)$
and *iso-dghm-ArrMap-vrange*[*dghm-cs-simps*]: $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor ArrMap \rfloor)$) = $\mathfrak{B}(\lfloor Arr \rfloor)$

syntax *-is-iso-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - ↦↦$_{DG.iso1}$ -)› [51, 51, 51] 51)
syntax-consts *-is-iso-dghm* ⇌ *is-iso-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}iso\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

sublocale *is-iso-dghm* ⊆ *ObjMap*: *v11* ‹$\mathfrak{F}(\lfloor ObjMap \rfloor)$›
rewrites $\mathcal{D}_\circ$ ($\mathfrak{F}(\lfloor ObjMap \rfloor)$) = $\mathfrak{A}(\lfloor Obj \rfloor)$ **and** $\mathcal{R}_\circ$ ($\mathfrak{F}(\lfloor ObjMap \rfloor)$) = $\mathfrak{B}(\lfloor Obj \rfloor)$
⟨*proof*⟩

sublocale *is-iso-dghm* ⊆ *ArrMap*: *v*11 ‹$\mathfrak{F}(\!|ArrMap|\!)$›
rewrites $\mathcal{D}_\circ\ (\mathfrak{F}(\!|ArrMap|\!)) = \mathfrak{A}(\!|Arr|\!)$ **and** $\mathcal{R}_\circ\ (\mathfrak{F}(\!|ArrMap|\!)) = \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] =
is-iso-dghm.iso-dghm-ObjMap-vrange
is-iso-dghm.iso-dghm-ArrMap-vrange

Rules.

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

mk-ide rf *is-iso-dghm-def*[*unfolded is-iso-dghm-axioms-def*]
|*intro is-iso-dghmI*|
|*dest is-iso-dghmD*[*dest*]|
|*elim is-iso-dghmE*[*elim*]|

Elementary properties.

lemma (**in** *is-iso-dghm*) *iso-dghm-Obj-HomDom-if-Obj-HomCod*[*elim*]:
assumes $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
obtains a **where** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$ **and** $b = \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemma (**in** *is-iso-dghm*) *iso-dghm-Arr-HomDom-if-Arr-HomCod*[*elim*]:
assumes $g \in_\circ \mathfrak{B}(\!|Arr|\!)$
obtains f **where** $f \in_\circ \mathfrak{A}(\!|Arr|\!)$ **and** $g = \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)$
⟨*proof*⟩

lemma (**in** *is-iso-dghm*) *iso-dghm-ObjMap-eqE*[*elim*]:
assumes $\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) = \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $a = b \Longrightarrow P$
shows P
⟨*proof*⟩

lemma (**in** *is-iso-dghm*) *iso-dghm-ArrMap-eqE*[*elim*]:
assumes $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!)$
and $f \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $f = g \Longrightarrow P$
shows P
⟨*proof*⟩

sublocale *is-iso-dghm* ⊆ *is-ft-dghm*
⟨*proof*⟩

sublocale *is-iso-dghm* ⊆ *is-fl-dghm*
⟨*proof*⟩

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

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

Opposite digraph isomorphisms

lemma (in *is-iso-dghm*) *is-iso-dghm-op*:
op-dghm $\mathfrak{F}$: *op-dg* $\mathfrak{A}$ $\mapsto\mapsto_{DG.iso\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-iso-dghm*) *is-iso-dghm-op′*:
assumes $\mathfrak{A}'$ = *op-dg* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-dg* $\mathfrak{B}$
shows *op-dghm* $\mathfrak{F}$: $\mathfrak{A}'$ $\mapsto\mapsto_{DG.iso\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas *is-iso-dghm-op*[*dg-op-intros*] = *is-iso-dghm.is-iso-dghm-op′*

The composition of isomorphisms of digraphs is an isomorphism of digraphs

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

An identity digraph homomorphism is an isomorphism of digraphs.

lemma (in *digraph*) *dg-dghm-id-is-iso-dghm*: *dghm-id* $\mathfrak{C}$: $\mathfrak{C}$ $\mapsto\mapsto_{DG.iso\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (in *digraph*) *dg-dghm-id-is-iso-dghm′*[*dghm-cs-intros*]:
assumes $\mathfrak{A}'$ = $\mathfrak{C}$ **and** $\mathfrak{B}'$ = $\mathfrak{C}$
shows *dghm-id* $\mathfrak{C}$: $\mathfrak{A}'$ $\mapsto\mapsto_{DG.iso\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*dghm-cs-intros*] = *digraph.dg-dghm-id-is-iso-dghm′*

3.4.12 Inverse digraph homomorphism

Definition and elementary properties

definition *inv-dghm* :: $V \Rightarrow V$
where *inv-dghm* $\mathfrak{F}$ = $[(\mathfrak{F}(\!|ObjMap|\!))^{-1}{}_{\circ}, (\mathfrak{F}(\!|ArrMap|\!))^{-1}{}_{\circ}, \mathfrak{F}(\!|HomCod|\!), \mathfrak{F}(\!|HomDom|\!)]_{\circ}$

Components.

lemma *inv-dghm-components*:
shows *inv-dghm* $\mathfrak{F}(\!|ObjMap|\!)$ = $(\mathfrak{F}(\!|ObjMap|\!))^{-1}{}_{\circ}$
and *inv-dghm* $\mathfrak{F}(\!|ArrMap|\!)$ = $(\mathfrak{F}(\!|ArrMap|\!))^{-1}{}_{\circ}$
and [*dghm-cs-simps*]: *inv-dghm* $\mathfrak{F}(\!|HomDom|\!)$ = $\mathfrak{F}(\!|HomCod|\!)$
and [*dghm-cs-simps*]: *inv-dghm* $\mathfrak{F}(\!|HomCod|\!)$ = $\mathfrak{F}(\!|HomDom|\!)$
⟨*proof*⟩

Object map

lemma (in *is-iso-dghm*) *inv-dghm-ObjMap-v*11[*dghm-cs-intros*]:
*v*11 (*inv-dghm* $\mathfrak{F}(\!|ObjMap|\!)$)
⟨*proof*⟩

lemmas [*dghm-cs-intros*] = *is-iso-dghm.inv-dghm-ObjMap-v*11

lemma (in *is-iso-dghm*) *inv-dghm-ObjMap-vdomain*[*dghm-cs-simps*]:
$\mathcal{D}_{\circ}$ (*inv-dghm* $\mathfrak{F}(\!|ObjMap|\!)$) = $\mathfrak{B}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ObjMap-vdomain*

lemma (**in** *is-iso-dghm*) *inv-dghm-ObjMap-app*[*dghm-cs-simps*]:
assumes $a' = \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$ **and** $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
shows *inv-dghm* $\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a' \rfloor) = a$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ObjMap-app*

lemma (**in** *is-iso-dghm*) *inv-dghm-ObjMap-vrange*[*dghm-cs-simps*]:
$\mathcal{R}_{\circ}$ (*inv-dghm* $\mathfrak{F}(\lfloor ObjMap \rfloor)$) = $\mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ObjMap-vrange*

Arrow map

lemma (**in** *is-iso-dghm*) *inv-dghm-ArrMap-v11*[*dghm-cs-intros*]:
v11 (*inv-dghm* $\mathfrak{F}(\lfloor ArrMap \rfloor)$)
⟨*proof*⟩

lemmas [*dghm-cs-intros*] = *is-iso-dghm.inv-dghm-ArrMap-v11*

lemma (**in** *is-iso-dghm*) *inv-dghm-ArrMap-vdomain*[*dghm-cs-simps*]:
$\mathcal{D}_{\circ}$ (*inv-dghm* $\mathfrak{F}(\lfloor ArrMap \rfloor)$) = $\mathfrak{B}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ArrMap-vdomain*

lemma (**in** *is-iso-dghm*) *inv-dghm-ArrMap-app*[*dghm-cs-simps*]:
assumes $a' = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor a \rfloor)$ **and** $a \in_{\circ} \mathfrak{A}(\lfloor Arr \rfloor)$
shows *inv-dghm* $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor a' \rfloor) = a$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ArrMap-app*

lemma (**in** *is-iso-dghm*) *inv-dghm-ArrMap-vrange*[*dghm-cs-simps*]:
$\mathcal{R}_{\circ}$ (*inv-dghm* $\mathfrak{F}(\lfloor ArrMap \rfloor)$) = $\mathfrak{A}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ArrMap-vrange*

Further properties

lemma (**in** *is-iso-dghm*) *iso-dghm-ObjMap-inv-dghm-ObjMap-app*[*dghm-cs-simps*]:
assumes $a \in_{\circ} \mathfrak{B}(\lfloor Obj \rfloor)$
shows $\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor$*inv-dghm* $\mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)\rfloor) = a$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.iso-dghm-ObjMap-inv-dghm-ObjMap-app*

lemma (**in** *is-iso-dghm*) *iso-dghm-ArrMap-inv-dghm-ArrMap-app*[*dghm-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor$*inv-dghm* $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)\rfloor) = f$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.iso-dghm-ArrMap-inv-dghm-ArrMap-app*

lemma (in *is-iso-dghm*) *iso-dghm-HomDom-is-arr-conv*:
assumes $f \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
shows $f : a \mapsto_{\mathfrak{A}} b$
⟨*proof*⟩

lemma (in *is-iso-dghm*) *iso-dghm-HomCod-is-arr-conv*:
assumes $f \in_\circ \mathfrak{B}(\!|Arr|\!)$
and $a \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and *inv-dghm* $\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!) : inv\text{-}dghm\ \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{A}} inv\text{-}dghm\ \mathfrak{F}(\!|ObjMap|\!)(\!|b|\!)$
shows $f : a \mapsto_{\mathfrak{B}} b$
⟨*proof*⟩

3.4.13 An isomorphism of digraphs is an isomorphism in the category *GRPH*

See Chapter I-3 in [39]).

lemma *is-iso-arr-is-iso-dghm*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{A}$
and $\mathfrak{G} \circ_{DGHM} \mathfrak{F} = dghm\text{-}id\ \mathfrak{A}$
and $\mathfrak{F} \circ_{DGHM} \mathfrak{G} = dghm\text{-}id\ \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-iso-dghm-is-iso-arr*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
shows [*dghm-cs-intros*]: $inv\text{-}dghm\ \mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{A}$
and [*dghm-cs-simps*]: $inv\text{-}dghm\ \mathfrak{F} \circ_{DGHM} \mathfrak{F} = dghm\text{-}id\ \mathfrak{A}$
and [*dghm-cs-simps*]: $\mathfrak{F} \circ_{DGHM} inv\text{-}dghm\ \mathfrak{F} = dghm\text{-}id\ \mathfrak{B}$
⟨*proof*⟩

Further properties

lemma (in *is-iso-dghm*) *iso-inv-dghm-ObjMap-dghm-ObjMap-app*[*dghm-cs-simps*]:
assumes $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
shows $inv\text{-}dghm\ \mathfrak{F}(\!|ObjMap|\!)(\!|\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)|\!) = a$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.iso-inv-dghm-ObjMap-dghm-ObjMap-app*

lemma (in *is-iso-dghm*) *iso-inv-dghm-ArrMap-dghm-ArrMap-app*[*dghm-cs-simps*]:
assumes $f : a \mapsto_{\mathfrak{A}} b$
shows $inv\text{-}dghm\ \mathfrak{F}(\!|ArrMap|\!)(\!|\mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)|\!) = f$
⟨*proof*⟩

lemmas [*dghm-cs-simps*] = *is-iso-dghm.iso-inv-dghm-ArrMap-dghm-ArrMap-app*

3.4.14 Isomorphic digraphs

Definition and elementary properties

See Chapter I-3 in [39]).

locale *iso-digraph* =
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B} :: V$

assumes *iso-digraph-is-iso-dghm*: $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$

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

sublocale *iso-digraph* $\subseteq$ *HomDom*: *digraph* α $\mathfrak{A}$ + *HomCod*: *digraph* α $\mathfrak{B}$
⟨*proof*⟩

Rules.

lemma *iso-digraphI'*:
assumes $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.iso\alpha} \mathfrak{B}$
shows $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

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

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

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

A digraph isomorphism is an equivalence relation

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

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

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

3.5 Smallness for digraph homomorphisms

3.5.1 Digraph homomorphism with tiny maps

Definition and elementary properties

The following construction is based on the concept of a *small functor* used in [57] in the context of the presentation of the set theory *ZFC/S*.

locale *is-tm-dghm* =
  is-dghm α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  HomDom: *digraph* α $\mathfrak{A}$ +
  HomCod: *digraph* α $\mathfrak{B}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *tm-dghm-ObjMap-in-Vset*[*dg-small-cs-intros*]: $\mathfrak{F}(\lfloor ObjMap \rfloor) \in_{\circ} Vset\ \alpha$
    and *tm-dghm-ArrMap-in-Vset*[*dg-small-cs-intros*]: $\mathfrak{F}(\lfloor ArrMap \rfloor) \in_{\circ} Vset\ \alpha$

syntax *-is-tm-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  ($\langle$(- :/ - $\mapsto\mapsto_{DG.tm^1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-tm-dghm* $\rightleftharpoons$ *is-tm-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}tm\text{-}dghm\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

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

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

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

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

lemmas [*dg-small-cs-intros*] =
  is-tm-dghm.tm-dghm-ObjMap-in-Vset
  is-tm-dghm.tm-dghm-ArrMap-in-Vset

Rules.

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

mk-ide rf *is-tm-dghm-def*[*unfolded is-tm-dghm-axioms-def*]
  |*intro is-tm-dghmI*|
  |*dest is-tm-dghmD*[*dest*]|
  |*elim is-tm-dghmE*[*elim*]|

lemmas [*dg-small-cs-intros*] = *is-tm-dghmD*(1)

Elementary properties.

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

lemmas (**in** *is-tm-dghm*)
  tm-dghm-HomDom-is-tiny-digraph = *HomDom.tiny-digraph-axioms*

lemmas [*dg-small-cs-intros*] = *is-tm-dghm.tm-dghm-HomDom-is-tiny-digraph*

Further rules.

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

Size.

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

Opposite digraph homomorphism with tiny maps

lemma (**in** *is-tm-dghm*) *is-tm-dghm-op*: *op-dghm* $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.tm\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

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

lemmas *is-tm-dghm-op*[*dg-op-intros*] = *is-tm-dghm.is-tm-dghm-op'*

Composition of a digraph homomorphism with tiny maps

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

Finite digraphs and digraph homomorphisms with tiny maps

lemma (**in** *is-dghm*) *dghm-is-tm-dghm-if-HomDom-finite-digraph*:
assumes *finite-digraph* α $\mathfrak{A}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

Constant digraph homomorphism

lemma *dghm-const-is-tm-dghm*:
assumes *tiny-digraph* α $\mathfrak{C}$ **and** *digraph* α $\mathfrak{D}$ **and** $f : a \mapsto_{\mathfrak{D}} a$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ a $f : \mathfrak{C} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *dghm-const-is-tm-dghm'*[*dg-small-cs-intros*]:
assumes *tiny-digraph* α $\mathfrak{C}$
and *digraph* α $\mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} a$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ a $f : \mathfrak{C}' \mapsto\mapsto_{DG.tm\alpha} \mathfrak{D}'$
⟨*proof*⟩

3.5.2 Tiny digraph homomorphism

Definition and elementary properties

locale *is-tiny-dghm* =
  is-dghm α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  HomDom: *tiny-digraph* α $\mathfrak{A}$ +
  HomCod: *tiny-digraph* α $\mathfrak{B}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

syntax *-is-tiny-dghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  ($\langle$(- :/ - $\mapsto\mapsto_{DG.tiny^1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-tiny-dghm* $\rightleftharpoons$ *is-tiny-dghm*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG.tiny\alpha} \mathfrak{B} \rightleftharpoons$ *CONST is-tiny-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

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

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

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

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

Rules.

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

mk-ide rf *is-tiny-dghm-def*
  |*intro is-tiny-dghmI*|
  |*dest is-tiny-dghmD*[*dest*]|
  |*elim is-tiny-dghmE*[*elim*]|

lemmas [*dg-small-cs-intros*] = *is-tiny-dghmD*(2,3)

Size.

lemma (**in** *is-tiny-dghm*) *tiny-dghm-ObjMap-in-Vset*[*dg-small-cs-intros*]:
  $\mathfrak{F}(\lfloor$*ObjMap*$\rfloor) \in_\circ$ *Vset* α
⟨*proof*⟩

lemmas [*dg-small-cs-intros*] = *is-tiny-dghm.tiny-dghm-ObjMap-in-Vset*

lemma (**in** *is-tiny-dghm*) *tiny-dghm-ArrMap-in-Vset*[*dg-small-cs-intros*]:
  $\mathfrak{F}(\lfloor$*ArrMap*$\rfloor) \in_\circ$ *Vset* α
⟨*proof*⟩

lemmas [*dg-small-cs-intros*] = *is-tiny-dghm.tiny-dghm-ArrMap-in-Vset*

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

sublocale *is-tiny-dghm* ⊆ *is-tm-dghm*

⟨*proof*⟩

lemmas (**in** *is-tiny-dghm*) *tiny-dghm-is-tm-dghm* = *is-tm-dghm-axioms*

lemmas [*dg-small-cs-intros*] = *is-tiny-dghm.tiny-dghm-is-tm-dghm*

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

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

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

Opposite tiny digraph homomorphism

lemma (**in** *is-tiny-dghm*) *is-tiny-dghm-op*:
op-dghm $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\mapsto_{DG.tiny\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

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

lemmas *is-tiny-dghm-op*[*dg-op-intros*] = *is-tiny-dghm.is-tiny-dghm-op'*

Composition of tiny digraph homomorphisms

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

Tiny constant digraph homomorphism

lemma *dghm-const-is-tiny-dghm*:
assumes *tiny-digraph* α $\mathfrak{C}$ **and** *tiny-digraph* α $\mathfrak{D}$ **and** $f : a \mapsto_{\mathfrak{D}} a$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ a f : $\mathfrak{C} \mapsto\mapsto_{DG.tiny\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *dghm-const-is-tiny-dghm'*[*dg-small-cs-intros*]:
assumes *tiny-digraph* α $\mathfrak{C}$
and *tiny-digraph* α $\mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} a$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows *dghm-const* $\mathfrak{C}$ $\mathfrak{D}$ a f : $\mathfrak{C}' \mapsto\mapsto_{DG.tiny\alpha} \mathfrak{D}'$
⟨*proof*⟩

3.6 Transformation of digraph homomorphisms

3.6.1 Background

named-theorems *tdghm-cs-simps*
named-theorems *tdghm-cs-intros*
named-theorems *nt-field-simps*

definition $NTMap :: V$ **where** [*nt-field-simps*]: $NTMap = 0$
definition $NTDom :: V$ **where** [*nt-field-simps*]: $NTDom = 1_{\mathbb{N}}$
definition $NTCod :: V$ **where** [*nt-field-simps*]: $NTCod = 2_{\mathbb{N}}$
definition $NTDGDom :: V$ **where** [*nt-field-simps*]: $NTDGDom = 3_{\mathbb{N}}$
definition $NTDGCod :: V$ **where** [*nt-field-simps*]: $NTDGCod = 4_{\mathbb{N}}$

3.6.2 Definition and elementary properties

A transformation of digraph homomorphisms, as presented in this work, is a generalization of the concept of a natural transformation, as presented in Chapter I-4 in [39], to digraphs and digraph homomorphisms. The generalization is performed by excluding the commutativity axiom from the definition.

The definition of a transformation of digraph homomorphisms is parameterized by a limit ordinal α such that $\omega < \alpha$. Such transformations of digraph homomorphisms are referred to either as α-transformations of digraph homomorphisms or transformations of α-digraph homomorphisms.

locale *is-tdghm* =
  $\mathcal{Z}\ \alpha$ +
  vfsequence $\mathfrak{N}$ +
  NTDom: *is-dghm* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$ +
  NTCod: *is-dghm* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{G}$
  for $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$ +
  assumes *tdghm-length*[*dg-cs-simps*]: *vcard* $\mathfrak{N} = 5_{\mathbb{N}}$
    and *tdghm-NTMap-vsv*: *vsv* $(\mathfrak{N}(\!|NTMap|\!))$
    and *tdghm-NTMap-vdomain*[*dg-cs-simps*]: $\mathcal{D}_{\circ}\ (\mathfrak{N}(\!|NTMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
    and *tdghm-NTDom*[*dg-cs-simps*]: $\mathfrak{N}(\!|NTDom|\!) = \mathfrak{F}$
    and *tdghm-NTCod*[*dg-cs-simps*]: $\mathfrak{N}(\!|NTCod|\!) = \mathfrak{G}$
    and *tdghm-NTDGDom*[*dg-cs-simps*]: $\mathfrak{N}(\!|NTDGDom|\!) = \mathfrak{A}$
    and *tdghm-NTDGCod*[*dg-cs-simps*]: $\mathfrak{N}(\!|NTDGCod|\!) = \mathfrak{B}$
    and *tdghm-NTMap-is-arr*:
      $a \in_{\circ} \mathfrak{A}(\!|Obj|\!) \Longrightarrow \mathfrak{N}(\!|NTMap|\!)(\!|a|\!) : \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!) \mapsto_{\mathfrak{B}} \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$

syntax *-is-tdghm* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  $(\langle (-:/ - \mapsto_{DGHM} -:/ - \mapsto\mapsto_{DG^1} -)\rangle\ [51, 51, 51, 51, 51]\ 51)$
syntax-consts *-is-tdghm* $\rightleftharpoons$ *is-tdghm*
translations $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B} \rightleftharpoons$
  CONST is-tdghm $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$

abbreviation *all-tdghms* :: $V \Rightarrow V$
  where *all-tdghms* $\alpha \equiv$ *set* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}\}$

abbreviation *tdghms* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *tdghms* $\alpha\ \mathfrak{A}\ \mathfrak{B} \equiv$ *set* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}\}$

abbreviation *these-tdghms* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *these-tdghms* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \equiv$ *set* $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}\}$

sublocale *is-tdghm* $\subseteq$ *NTMap*: *vsv* $\langle\mathfrak{N}(\!|NTMap|\!)\rangle$
  rewrites $\mathcal{D}_{\circ}\ (\mathfrak{N}(\!|NTMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
  ⟨*proof*⟩

lemmas [*dg-cs-simps*] =
is-tdghm.tdghm-length
is-tdghm.tdghm-NTMap-vdomain
is-tdghm.tdghm-NTDom
is-tdghm.tdghm-NTCod
is-tdghm.tdghm-NTDGDom
is-tdghm.tdghm-NTDGCod

lemma (in *is-tdghm*) *tdghm-NTMap-is-arr'*[*dg-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
and $A = \mathfrak{F}(\!|ObjMap|\!)(\!|a|\!)$
and $B = \mathfrak{G}(\!|ObjMap|\!)(\!|a|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|a|\!) : A \mapsto_{\mathfrak{B}} B$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *is-tdghm.tdghm-NTMap-is-arr'*

Rules.

lemma (in *is-tdghm*) *is-tdghm-axioms'*[*dg-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 $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-tdghm-def*[*unfolded is-tdghm-axioms-def*]
|*intro is-tdghmI*|
|*dest is-tdghmD*[*dest*]|
|*elim is-tdghmE*[*elim*]|

lemmas [*dg-cs-intros*] =
is-tdghmD(3,4)

Elementary properties.

lemma *tdghm-eqI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG\alpha} \mathfrak{B}'$
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 (in *is-tdghm*) *tdghm-def*:
$\mathfrak{N} = [\mathfrak{N}(\!|NTMap|\!), \mathfrak{N}(\!|NTDom|\!), \mathfrak{N}(\!|NTCod|\!), \mathfrak{N}(\!|NTDGDom|\!), \mathfrak{N}(\!|NTDGCod|\!)]_{\circ}$
⟨*proof*⟩

lemma (in *is-tdghm*) *tdghm-NTMap-app-in-Arr*[*dg-cs-intros*]:
assumes $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
shows $\mathfrak{N}(\!|NTMap|\!)(\!|a|\!) \in_{\circ} \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *is-tdghm.tdghm-NTMap-app-in-Arr*

lemma (in *is-tdghm*) *tdghm-NTMap-vrange-vifunion*:
$\mathcal{R}_{\circ}(\mathfrak{N}(\!|NTMap|\!)) \subseteq_{\circ} (\bigcup_{\circ} a \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{F}(\!|ObjMap|\!)).\ \bigcup_{\circ} b \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{G}(\!|ObjMap|\!)).\ Hom\ \mathfrak{B}\ a\ b)$
⟨*proof*⟩

lemma (in *is-tdghm*) *tdghm-NTMap-vrange*: $\mathcal{R}_{\circ}\ (\mathfrak{N}(\!|NTMap|\!)) \subseteq_{\circ} \mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

Size.

lemma (in *is-tdghm*) *tdghm-NTMap-vsubset-Vset*: $\mathfrak{N}(\!|NTMap|\!) \subseteq_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (in *is-tdghm*) *tdghm-NTMap-in-Vset*:
assumes $\alpha \in_{\circ} \beta$
shows $\mathfrak{N}(\!|NTMap|\!) \in_{\circ} Vset\ \beta$
⟨*proof*⟩

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

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

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

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

lemma *small-these-tdghms*[*simp*]: $small\ \{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}\}$
⟨*proof*⟩

Further elementary results.

lemma *these-tdghms-iff*:
$\mathfrak{N} \in_{\circ} these\text{-}tdghms\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \longleftrightarrow \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

3.6.3 Opposite transformation of digraph homomorphisms

Definition and elementary properties

See section 1.5 in [15].

definition *op-tdghm* :: $V \Rightarrow V$
where *op-tdghm* $\mathfrak{N}$ =
$[$
$\mathfrak{N}(\!|NTMap|\!)$,
op-dghm $(\mathfrak{N}(\!|NTCod|\!))$,
op-dghm $(\mathfrak{N}(\!|NTDom|\!))$,
op-dg $(\mathfrak{N}(\!|NTDGDom|\!))$,
op-dg $(\mathfrak{N}(\!|NTDGCod|\!))$
$]_{\circ}$

Components.

lemma *op-tdghm-components*[*dg-op-simps*]:
shows *op-tdghm* $\mathfrak{N}(\!|NTMap|\!)$ = $\mathfrak{N}(\!|NTMap|\!)$
and *op-tdghm* $\mathfrak{N}(\!|NTDom|\!)$ = *op-dghm* $(\mathfrak{N}(\!|NTCod|\!))$

and *op-tdghm* $\mathfrak{N}(\!|NTCod|\!)$ = *op-dghm* $(\mathfrak{N}(\!|NTDom|\!))$
and *op-tdghm* $\mathfrak{N}(\!|NTDGDom|\!)$ = *op-dg* $(\mathfrak{N}(\!|NTDGDom|\!))$
and *op-tdghm* $\mathfrak{N}(\!|NTDGCod|\!)$ = *op-dg* $(\mathfrak{N}(\!|NTDGCod|\!))$
⟨*proof*⟩

Further properties

lemma (in *is-tdghm*) *is-tdghm-op*:
op-tdghm $\mathfrak{N}$: *op-dghm* $\mathfrak{G} \mapsto_{DGHM}$ *op-dghm* $\mathfrak{F}$: *op-dg* $\mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tdghm*) *is-tdghm-op'*[*dg-op-intros*]:
assumes $\mathfrak{G}'$ = *op-dghm* $\mathfrak{G}$
and $\mathfrak{F}'$ = *op-dghm* $\mathfrak{F}$
and $\mathfrak{A}'$ = *op-dg* $\mathfrak{A}$
and $\mathfrak{B}'$ = *op-dg* $\mathfrak{B}$
shows *op-tdghm* $\mathfrak{N}$: $\mathfrak{G}' \mapsto_{DGHM} \mathfrak{F}' : \mathfrak{A}' \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-tdghm-op*[*dg-op-intros*] = *is-tdghm.is-tdghm-op'*

lemma (in *is-tdghm*) *tdghm-op-tdghm-op-tdghm*[*dg-op-simps*]:
op-tdghm (*op-tdghm* $\mathfrak{N}$) = $\mathfrak{N}$
⟨*proof*⟩

lemmas *tdghm-op-tdghm-op-tdghm*[*dg-op-simps*] =
is-tdghm.tdghm-op-tdghm-op-tdghm

lemma *eq-op-tdghm-iff*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}'$
shows *op-tdghm* $\mathfrak{N}$ = *op-tdghm* $\mathfrak{N}' \longleftrightarrow \mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

3.6.4 Composition of a transformation of digraph homomorphisms and a digraph homomorphism

Definition and elementary properties

definition *tdghm-dghm-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{TDGHM-DGHM}$› 55)
where $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H}$ =
[
$(\lambda a \in_\circ \mathfrak{H}(\!|HomDom|\!)(\!|Obj|\!).\ \mathfrak{N}(\!|NTMap|\!)(\!|\mathfrak{H}(\!|ObjMap|\!)(\!|a|\!)|\!))$,
$\mathfrak{N}(\!|NTDom|\!) \circ_{DGHM} \mathfrak{H}$,
$\mathfrak{N}(\!|NTCod|\!) \circ_{DGHM} \mathfrak{H}$,
$\mathfrak{H}(\!|HomDom|\!)$,
$\mathfrak{N}(\!|NTDGCod|\!)$
]$_\circ$

Components.

lemma *tdghm-dghm-comp-components*:
shows $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\!|NTMap|\!)$ =
$(\lambda a \in_\circ \mathfrak{H}(\!|HomDom|\!)(\!|Obj|\!).\ \mathfrak{N}(\!|NTMap|\!)(\!|\mathfrak{H}(\!|ObjMap|\!)(\!|a|\!)|\!))$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\!|NTDom|\!) = \mathfrak{N}(\!|NTDom|\!) \circ_{DGHM} \mathfrak{H}$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\!|NTCod|\!) = \mathfrak{N}(\!|NTCod|\!) \circ_{DGHM} \mathfrak{H}$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:

$(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\lfloor NTDGDom \rfloor) = \mathfrak{H}(\lfloor HomDom \rfloor)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\lfloor NTDGCod \rfloor) = \mathfrak{N}(\lfloor NTDGCod \rfloor)$
⟨*proof*⟩

Transformation map

mk-VLambda *tdghm-dghm-comp-components*(1)
|*vsv tdghm-dghm-comp-NTMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|

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

lemmas [*dg-cs-simps*] =
is-dghm.tdghm-dghm-comp-NTMap-vdomain
is-dghm.tdghm-dghm-comp-NTMap-app

lemma *tdghm-dghm-comp-NTMap-vrange*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ\ ((\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

Opposite of the composition of a transformation of digraph homomorphisms and a digraph homomorphism

lemma *op-tdghm-tdghm-dghm-comp*[*dg-op-simps*]:
op-tdghm $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})$ = *op-tdghm* $\mathfrak{N} \circ_{TDGHM-DGHM}$ *op-dghm* $\mathfrak{H}$
⟨*proof*⟩

Composition of a transformation of digraph homomorphisms and a digraph homomorphism is a transformation of digraph homomorphisms

lemma *tdghm-dghm-comp-is-tdghm*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F} \circ_{DGHM} \mathfrak{H} \mapsto_{DGHM} \mathfrak{G} \circ_{DGHM} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *tdghm-dghm-comp-is-tdghm'*[*dg-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{DGHM} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{DGHM} \mathfrak{H}$
shows $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma *tdghm-dghm-comp-tdghm-dghm-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{H} \mapsto_{DGHM} \mathfrak{H}' : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{G}) \circ_{TDGHM-DGHM} \mathfrak{F} = \mathfrak{N} \circ_{TDGHM-DGHM} (\mathfrak{G} \circ_{DGHM} \mathfrak{F})$
⟨*proof*⟩

lemma (**in** *is-tdghm*) *tdghm-tdghm-dghm-comp-dghm-id*[*dg-cs-simps*]:
$\mathfrak{N} \circ_{TDGHM-DGHM}$ *dghm-id* $\mathfrak{A} = \mathfrak{N}$

⟨*proof*⟩

lemmas [*dg-cs-simps*] = *is-tdghm.tdghm-tdghm-dghm-comp-dghm-id*

3.6.5 Composition of a digraph homomorphism and a transformation of digraph homomorphisms

Definition and elementary properties

definition *dghm-tdghm-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{DGHM-TDGHM}$› 55)
where $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N}$ =
[
($\lambda a \in_\circ \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!)$. $\mathfrak{H}(\!|ArrMap|\!)(\!|\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)|\!)$),
$\mathfrak{H} \circ_{DGHM} \mathfrak{N}(\!|NTDom|\!)$,
$\mathfrak{H} \circ_{DGHM} \mathfrak{N}(\!|NTCod|\!)$,
$\mathfrak{N}(\!|NTDGDom|\!)$,
$\mathfrak{H}(\!|HomCod|\!)$
]$_\circ$

Components.

lemma *dghm-tdghm-comp-components*:
shows $(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(\!|NTMap|\!)$ =
($\lambda a \in_\circ \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!)$. $\mathfrak{H}(\!|ArrMap|\!)(\!|\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)|\!)$)
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(\!|NTDom|\!) = \mathfrak{H} \circ_{DGHM} \mathfrak{N}(\!|NTDom|\!)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(\!|NTCod|\!) = \mathfrak{H} \circ_{DGHM} \mathfrak{N}(\!|NTCod|\!)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(\!|NTDGDom|\!) = \mathfrak{N}(\!|NTDGDom|\!)$
and [*dg-shared-cs-simps*, *dg-cs-simps*]:
$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(\!|NTDGCod|\!) = \mathfrak{H}(\!|HomCod|\!)$
⟨*proof*⟩

Transformation map

mk-VLambda *dghm-tdghm-comp-components*(1)
|*vsv dghm-tdghm-comp-NTMap-vsv*[*dg-shared-cs-intros*, *dg-cs-intros*]|

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

lemmas [*dg-cs-simps*] =
is-tdghm.dghm-tdghm-comp-NTMap-vdomain
is-tdghm.dghm-tdghm-comp-NTMap-app

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

Opposite of the composition of a digraph homomorphism and a transformation of digraph homomorphisms

lemma *op-tdghm-dghm-tdghm-comp*[*dg-op-simps*]:
op-tdghm $(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})$ = *op-dghm* $\mathfrak{H} \circ_{DGHM-TDGHM}$ *op-tdghm* $\mathfrak{N}$
⟨*proof*⟩

Composition of a digraph homomorphism and a transformation of digraph homomorphisms is a transformation of digraph homomorphisms

lemma *dghm-tdghm-comp-is-tdghm*:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{H} \circ_{DGHM} \mathfrak{F} \mapsto_{DGHM} \mathfrak{H} \circ_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *dghm-tdghm-comp-is-tdghm'*[*dg-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{H} \circ_{DGHM} \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{H} \circ_{DGHM} \mathfrak{G}$
shows $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma *dghm-comp-dghm-tdghm-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{H} \mapsto_{DGHM} \mathfrak{H}' : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
shows $(\mathfrak{G} \circ_{DGHM} \mathfrak{F}) \circ_{DGHM-TDGHM} \mathfrak{N} = \mathfrak{G} \circ_{DGHM-TDGHM} (\mathfrak{F} \circ_{DGHM-TDGHM} \mathfrak{N})$
⟨*proof*⟩

lemma (**in** *is-tdghm*) *tdghm-dghm-tdghm-comp-dghm-id*[*dg-cs-simps*]:
dghm-id $\mathfrak{B} \circ_{DGHM-TDGHM} \mathfrak{N} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*dg-cs-simps*] = *is-tdghm.tdghm-dghm-tdghm-comp-dghm-id*

lemma *dghm-tdghm-comp-tdghm-dghm-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
and $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
shows $(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N}) \circ_{TDGHM-DGHM} \mathfrak{K} = \mathfrak{H} \circ_{DGHM-TDGHM} (\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{K})$
⟨*proof*⟩

3.7 Smallness for transformations of digraph homomorphisms

3.7.1 Transformation of digraph homomorphisms with tiny maps

Definition and elementary properties

locale *is-tm-tdghm* =
  $\mathcal{Z}$ α +
  NTDom: *is-tm-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  NTCod: *is-tm-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ +
  is-tdghm α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

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

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

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

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

Rules.

lemma (in *is-tm-tdghm*) *is-tm-tdghm-axioms'*[*dg-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 $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG.tm_{\alpha'}} \mathfrak{B}'$
  ⟨*proof*⟩

mk-ide rf *is-tm-tdghm-def*
  |*intro is-tm-tdghmI*|
  |*dest is-tm-tdghmD*[*dest*]|
  |*elim is-tm-tdghmE*[*elim*]|

lemmas [*dg-small-cs-intros*] = *is-tm-tdghmD*(2,3,4)

Size.

lemma (in *is-tm-tdghm*) *tm-tdghm-NTMap-in-Vset*: $\mathfrak{N}(\lfloor NTMap \rfloor) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

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

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

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

Further elementary results.

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

Opposite transformation of digraph homomorphisms with tiny maps

lemma (in *is-tm-tdghm*) *is-tm-tdghm-op*:
op-tdghm $\mathfrak{N}$: *op-dghm* $\mathfrak{G}$ $\mapsto_{DGHM.tm}$ *op-dghm* $\mathfrak{F}$: *op-dg* $\mathfrak{A}$ $\mapsto\mapsto_{DG.tm\alpha}$ *op-dg* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-tm-tdghm*) *is-tm-tdghm-op'*[*dg-op-intros*]:
assumes $\mathfrak{G}'$ = *op-dghm* $\mathfrak{G}$
and $\mathfrak{F}'$ = *op-dghm* $\mathfrak{F}$
and $\mathfrak{A}'$ = *op-dg* $\mathfrak{A}$
and $\mathfrak{B}'$ = *op-dg* $\mathfrak{B}$
shows *op-tdghm* $\mathfrak{N}$: $\mathfrak{G}' \mapsto_{DGHM.tm} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-tm-tdghm-op*[*dg-op-intros*] = *is-tm-tdghm.is-tm-tdghm-op'*

Composition of a transformation of digraph homomorphisms with tiny maps and a digraph homomorphism with tiny maps

lemma *tdghm-dghm-comp-is-tm-tdghm*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F} \circ_{DGHM} \mathfrak{H} \mapsto_{DGHM.tm} \mathfrak{G} \circ_{DGHM} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *tdghm-dghm-comp-is-tm-tdghm'*[*dg-small-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{DGHM} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{DGHM} \mathfrak{H}$
shows $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

Composition of a digraph homomorphism with tiny maps and a transformation of digraph homomorphisms with tiny maps

lemma *dghm-tdghm-comp-is-tm-tdghm*:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}$
shows $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{H} \circ_{DGHM} \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{H} \circ_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *dghm-tdghm-comp-is-tm-tdghm'*[*dg-small-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{H} \circ_{DGHM} \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{H} \circ_{DGHM} \mathfrak{G}$
shows $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

3.7.2 Transformation of homomorphisms of tiny digraphs

Definition and elementary properties

locale *is-tiny-tdghm* =

$\mathcal{Z}$ α +
NTDom: *is-tiny-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
NTCod: *is-tiny-dghm* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$ +
is-tdghm α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$
for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

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

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

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

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

Rules.

lemmas (**in** *is-tiny-tdghm*) [*dg-small-cs-intros*] = *is-tiny-tdghm-axioms*

mk-ide rf *is-tiny-tdghm-def*
|*intro is-tiny-tdghmI*[*intro*]|
|*dest is-tiny-tdghmD*[*dest*]|
|*elim is-tiny-tdghmE*[*elim*]|

lemmas [*dg-small-cs-intros*] = *is-tiny-tdghmD*(2,3,4)

Elementary properties.

sublocale *is-tiny-tdghm* $\subseteq$ *is-tm-tdghm*
⟨*proof*⟩

lemmas (**in** *is-tiny-tdghm*) *tiny-tdghm-is-tm-tdghm* = *is-tm-tdghm-axioms*

lemmas [*dg-small-cs-intros*] = *is-tiny-tdghm.tiny-tdghm-is-tm-tdghm*

Size.

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

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

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

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

lemma *tiny-tdghms-vsubset-Vset*[*simp*]:
set {𝔑. ∃𝔉 𝔊. 𝔑 : 𝔉 $\mapsto_{DGHM.tiny}$ 𝔊 : 𝔄 $\mapsto\mapsto_{DG.tiny\alpha}$ 𝔅} $\subseteq_\circ$ *Vset* α
(**is** ‹*set ?tdghms* $\subseteq_\circ$ -›)
⟨*proof*⟩

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

Further elementary results.

lemma *these-tiny-tdghms-iff*:
$\mathfrak{N} \in_\circ$ *these-tiny-tdghms* α 𝔄 𝔅 𝔉 𝔊 ⟷
𝔑 : 𝔉 $\mapsto_{DGHM.tiny}$ 𝔊 : 𝔄 $\mapsto\mapsto_{DG.tiny\alpha}$ 𝔅
⟨*proof*⟩

Opposite transformation of homomorphisms of tiny digraphs

lemma (**in** *is-tiny-tdghm*) *is-tm-tdghm-op*: *op-tdghm* 𝔑 :
op-dghm 𝔊 $\mapsto_{DGHM.tiny}$ *op-dghm* 𝔉 : *op-dg* 𝔄 $\mapsto\mapsto_{DG.tiny\alpha}$ *op-dg* 𝔅
⟨*proof*⟩

lemma (**in** *is-tiny-tdghm*) *is-tiny-tdghm-op'*[*dg-op-intros*]:
assumes 𝔊′ = *op-dghm* 𝔊
and 𝔉′ = *op-dghm* 𝔉
and 𝔄′ = *op-dg* 𝔄
and 𝔅′ = *op-dg* 𝔅
shows *op-tdghm* 𝔑 : 𝔊′ $\mapsto_{DGHM.tiny}$ 𝔉′ : 𝔄′ $\mapsto\mapsto_{DG.tiny\alpha}$ 𝔅′
⟨*proof*⟩

lemmas *is-tiny-tdghm-op*[*dg-op-intros*] = *is-tiny-tdghm.is-tiny-tdghm-op'*

3.8 Product digraph

3.8.1 Background

The concept of a product digraph, as presented in this work, is a generalization of the concept of a product category, as presented in Chapter II-3 in [39].

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

3.8.2 Product digraph: definition and elementary properties

definition *dg-prod* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  where *dg-prod* $I\ \mathfrak{A} =$
    [
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$,
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!))(\!|f(\!|i|\!)|\!)))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!))(\!|f(\!|i|\!)|\!)))$
    ]$_{\circ}$

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

Components.

lemma *dg-prod-components*:
  shows $(\prod_{DG} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Obj|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$
    and $(\prod_{DG} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Arr|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$
    and $(\prod_{DG} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Dom|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!))(\!|f(\!|i|\!)|\!)))$
    and $(\prod_{DG} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Cod|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!))(\!|f(\!|i|\!)|\!)))$
  $\langle$*proof*$\rangle$

3.8.3 Local assumptions for a product digraph

locale *pdigraph-base* = $\mathcal{Z}\ \alpha$ **for** $\alpha\ I$ **and** $\mathfrak{A} :: V \Rightarrow V$ +
  assumes *pdg-digraphs*[*dg-prod-cs-intros*]: $i \in_{\circ} I \Longrightarrow$ *digraph* $\alpha\ (\mathfrak{A}\ i)$
    and *pdg-index-in-Vset*[*dg-cs-intros*]: $I \in_{\circ}$ *Vset* α

Rules.

lemma (**in** *pdigraph-base*) *pdigraph-base-axioms'*[*dg-prod-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $I' = I$
  shows *pdigraph-base* $\alpha'\ I'\ \mathfrak{A}$
  $\langle$*proof*$\rangle$

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

Elementary properties.

lemma (**in** *pdigraph-base*) *pdg-Obj-in-Vset*:
  assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_{\circ} \beta$
  shows $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!)) \in_{\circ}$ *Vset* β
$\langle$*proof*$\rangle$

lemma (in *pdigraph-base*) *pdg-Arr-in-Vset*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows $(\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!)) \in_\circ Vset\ \beta$
⟨*proof*⟩

lemmas-with (in *pdigraph-base*) [*folded dg-prod-components*]:
pdg-dg-prod-Obj-in-Vset[*dg-cs-intros*] = *pdg-Obj-in-Vset*
and *pdg-dg-prod-Arr-in-Vset*[*dg-cs-intros*] = *pdg-Arr-in-Vset*

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

Object

lemma *dg-prod-ObjI*:
assumes *vsv a* **and** $\mathcal{D}_\circ\ a = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow a(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Obj|\!)$
shows $a \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-ObjD*:
assumes $a \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
shows *vsv a* **and** $\mathcal{D}_\circ\ a = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow a(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-ObjE*:
assumes $a \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
obtains *vsv a* **and** $\mathcal{D}_\circ\ a = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow a(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-Obj-cong*:
assumes $g \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
and $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow g(\!|i|\!) = f(\!|i|\!)$
shows $g = f$
⟨*proof*⟩

Arrow

lemma *dg-prod-ArrI*:
assumes *vsv f* **and** $\mathcal{D}_\circ\ f = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Arr|\!)$
shows $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-ArrD*:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
shows *vsv f* **and** $\mathcal{D}_\circ\ f = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-ArrE*:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
obtains *vsv f* **and** $\mathcal{D}_\circ\ f = I$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) \in_\circ \mathfrak{A}\ i(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-Arr-cong*:
assumes $g \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$

and $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow g(\!|i|\!) = f(\!|i|\!)$
shows $g = f$
⟨*proof*⟩

Domain

mk-VLambda *dg-prod-components*(3)
|*vsv dg-prod-Dom-vsv*[*dg-cs-intros*]|
|*vdomain dg-prod-Dom-vdomain*[*folded dg-prod-components*, *dg-cs-simps*]|
|*app dg-prod-Dom-app*[*folded dg-prod-components*]|

lemma (**in** *pdigraph-base*) *dg-prod-Dom-app-in-Obj*[*dg-cs-intros*]:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
shows $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Dom|\!)(\!|f|\!) \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-Dom-app-component-app*[*dg-cs-simps*]:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$ **and** $i \in_\circ I$
shows $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Dom|\!)(\!|f|\!)(\!|i|\!) = \mathfrak{A}\ i(\!|Dom|\!)(\!|f(\!|i|\!)|\!)$
⟨*proof*⟩

Codomain

mk-VLambda *dg-prod-components*(4)
|*vsv dg-prod-Cod-vsv*[*dg-cs-intros*]|
|*vdomain dg-prod-Cod-vdomain*[*folded dg-prod-components*, *dg-cs-simps*]|
|*app dg-prod-Cod-app*[*folded dg-prod-components*]|

lemma (**in** *pdigraph-base*) *dg-prod-Cod-app-in-Obj*[*dg-cs-intros*]:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
shows $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Cod|\!)(\!|f|\!) \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-Cod-app-component-app*[*dg-cs-simps*]:
assumes $f \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$ **and** $i \in_\circ I$
shows $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Cod|\!)(\!|f|\!)(\!|i|\!) = \mathfrak{A}\ i(\!|Cod|\!)(\!|f(\!|i|\!)|\!)$
⟨*proof*⟩

A product α-digraph is a tiny β-digraph

lemma (**in** *pdigraph-base*) *pdg-tiny-digraph-dg-prod*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *tiny-digraph* β $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

lemma (**in** *pdigraph-base*) *pdg-tiny-digraph-dg-prod′*:
tiny-digraph $(\alpha + \omega)$ $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Arrow with a domain and a codomain

lemma (**in** *pdigraph-base*) *dg-prod-is-arrI*:
assumes *vsv f*
and $\mathcal{D}_\circ\ f = I$
and *vsv a*
and $\mathcal{D}_\circ\ a = I$
and *vsv b*

and $\mathcal{D}_\circ\ b = I$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) : a(\!|i|\!) \mapsto_{\mathfrak{A}\ i} b(\!|i|\!)$
shows $f : a \mapsto_{\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i} b$
⟨*proof*⟩

lemma (**in** *pdigraph-base*) *dg-prod-is-arrD*[*dest*]:
assumes $f : a \mapsto_{\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i} b$
shows *vsv f*
and $\mathcal{D}_\circ\ f = I$
and *vsv a*
and $\mathcal{D}_\circ\ a = I$
and *vsv b*
and $\mathcal{D}_\circ\ b = I$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) : a(\!|i|\!) \mapsto_{\mathfrak{A}\ i} b(\!|i|\!)$
⟨*proof*⟩

lemma (**in** *pdigraph-base*) *dg-prod-is-arrE*[*elim*]:
assumes $f : a \mapsto_{\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i} b$
obtains *vsv f*
and $\mathcal{D}_\circ\ f = I$
and *vsv a*
and $\mathcal{D}_\circ\ a = I$
and *vsv b*
and $\mathcal{D}_\circ\ b = I$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow f(\!|i|\!) : a(\!|i|\!) \mapsto_{\mathfrak{A}\ i} b(\!|i|\!)$
⟨*proof*⟩

3.8.4 Further local assumptions for product digraphs

Definition and elementary properties

locale *pdigraph* = *pdigraph-base* $\alpha\ I\ \mathfrak{A}$ **for** $\alpha\ I\ \mathfrak{A}$ +
assumes *pdg-Obj-vsubset-Vset*: $J \subseteq_\circ I \Longrightarrow (\prod_{DG} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and *pdg-Hom-vifunion-in-Vset*:
⟦
$J \subseteq_\circ I$;
$A \subseteq_\circ (\prod_{DG} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$B \subseteq_\circ (\prod_{DG} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$A \in_\circ Vset\ \alpha$;
$B \in_\circ Vset\ \alpha$
⟧ $\Longrightarrow (\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ (\prod_{DG} i \in_\circ J.\ \mathfrak{A}\ i)\ a\ b) \in_\circ Vset\ \alpha$

Rules.

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

mk-ide rf *pdigraph-def*[*unfolded pdigraph-axioms-def*]
|*intro pdigraphI*|
|*dest pdigraphD*[*dest*]|
|*elim pdigraphE*[*elim*]|

lemmas [*dg-prod-cs-intros*] = *pdigraphD*(1)

Elementary properties.

lemma (**in** *pdigraph*) *pdg-Obj-vsubset-Vset'*: $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in *pdigraph*) *pdg-Hom-vifunion-in-Vset'*:
assumes $A \subseteq_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
and $B \subseteq_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
and $A \in_\circ Vset\ \alpha$
and $B \in_\circ Vset\ \alpha$
shows $(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)\ a\ b) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in *pdigraph*) *pdg-vsubset-index-pdigraph*:
assumes $J \subseteq_\circ I$
shows *pdigraph* $\alpha\ J\ \mathfrak{A}$
⟨*proof*⟩

A product α-digraph is an α-digraph

lemma (in *pdigraph*) *pdg-digraph-dg-prod*: *digraph* $\alpha\ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

3.8.5 Local assumptions for a finite product digraph

Definition and elementary properties

locale *finite-pdigraph* = *pdigraph-base* $\alpha\ I\ \mathfrak{A}$ **for** $\alpha\ I\ \mathfrak{A}$ +
assumes *fin-pdg-index-vfinite*: *vfinite* I

Rules.

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

mk-ide rf *finite-pdigraph-def*[*unfolded finite-pdigraph-axioms-def*]
|*intro finite-pdigraphI*|
|*dest finite-pdigraphD*[*dest*]|
|*elim finite-pdigraphE*[*elim*]|

lemmas [*dg-prod-cs-intros*] = *finite-pdigraphD*(1)

Local assumptions for a finite product digraph and local assumptions for an arbitrary product digraph

sublocale *finite-pdigraph* ⊆ *pdigraph* $\alpha\ I\ \mathfrak{A}$
⟨*proof*⟩

3.8.6 Binary union and complement

Application-specific methods

method *vdiff-of-vunion* **uses** *rule assms subset* =
(
rule
rule
[
OF vintersection-complement assms,
unfolded vunion-complement[*OF subset*]
]
)

method *vdiff-of-vunion′* **uses** *rule assms subset* =
(
rule
rule
[
OF vintersection-complement complement-vsubset subset assms,
unfolded vunion-complement[*OF subset*]
]
)

Results

lemma *dg-prod-vunion-Obj-in-Obj*:
assumes *vdisjnt J K*
and $b \in_\circ (\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
and $c \in_\circ (\prod_{DG} k \in_\circ K.\ \mathfrak{A}\ k)(\!|Obj|\!)$
shows $b \cup_\circ c \in_\circ (\prod_{DG} i \in_\circ J \cup_\circ K.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-vdiff-vunion-Obj-in-Obj*:
assumes $J \subseteq_\circ I$
and $b \in_\circ (\prod_{DG} k \in_\circ I -_\circ J.\ \mathfrak{A}\ k)(\!|Obj|\!)$
and $c \in_\circ (\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)(\!|Obj|\!)$
shows $b \cup_\circ c \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-vunion-Arr-in-Arr*:
assumes *vdisjnt J K*
and $b \in_\circ (\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)(\!|Arr|\!)$
and $c \in_\circ (\prod_{DG} k \in_\circ K.\ \mathfrak{A}\ k)(\!|Arr|\!)$
shows $b \cup_\circ c \in_\circ (\prod_{DG} i \in_\circ J \cup_\circ K.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-vdiff-vunion-Arr-in-Arr*:
assumes $J \subseteq_\circ I$
and $b \in_\circ (\prod_{DG} k \in_\circ I -_\circ J.\ \mathfrak{A}\ k)(\!|Arr|\!)$
and $c \in_\circ (\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)(\!|Arr|\!)$
shows $b \cup_\circ c \in_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma (**in** *pdigraph*) *pdg-dg-prod-vunion-is-arr*:
assumes *vdisjnt J K*
and $J \subseteq_\circ I$
and $K \subseteq_\circ I$
and $g : a \mapsto_{(\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)} b$
and $f : c \mapsto_{(\prod_{DG} k \in_\circ K.\ \mathfrak{A}\ k)} d$
shows $g \cup_\circ f : a \cup_\circ c \mapsto_{(\prod_{DG} i \in_\circ J \cup_\circ K.\ \mathfrak{A}\ i)} b \cup_\circ d$
⟨*proof*⟩

lemma (**in** *pdigraph*) *pdg-dg-prod-vdiff-vunion-is-arr*:
assumes $J \subseteq_\circ I$
and $g : a \mapsto_{(\prod_{DG} k \in_\circ I -_\circ J.\ \mathfrak{A}\ k)} b$
and $f : c \mapsto_{(\prod_{DG} j \in_\circ J.\ \mathfrak{A}\ j)} d$
shows $g \cup_\circ f : a \cup_\circ c \mapsto_{\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i} b \cup_\circ d$
⟨*proof*⟩

3.8.7 Projection

Definition and elementary properties

See Chapter II-3 in [39].

definition *dghm-proj* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$ ($\langle\pi_{DG}\rangle$)
  where $\pi_{DG}\ I\ \mathfrak{A}\ i =$
    [
      $(\lambda a \in_\circ ((\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\lfloor Obj \rfloor)).\ a(\lfloor i \rfloor))$,
      $(\lambda f \in_\circ ((\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\lfloor Arr \rfloor)).\ f(\lfloor i \rfloor))$,
      $(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$,
      $\mathfrak{A}\ i$
    ]$_\circ$

Components.

lemma *dghm-proj-components*:
  shows $\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor ObjMap \rfloor) = (\lambda a \in_\circ ((\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\lfloor Obj \rfloor)).\ a(\lfloor i \rfloor))$
    and $\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor ArrMap \rfloor) = (\lambda f \in_\circ ((\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\lfloor Arr \rfloor)).\ f(\lfloor i \rfloor))$
    and $\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor HomDom \rfloor) = (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
    and $\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor HomCod \rfloor) = \mathfrak{A}\ i$
  ⟨*proof*⟩

Object map.

mk-VLambda *dghm-proj-components*(1)
  |*vsv dghm-proj-ObjMap-vsv*[*dg-cs-intros*]|
  |*vdomain dghm-proj-ObjMap-vdomain*[*dg-cs-simps*]|
  |*app dghm-proj-ObjMap-app*[*dg-cs-simps*]|

lemma (**in** *pdigraph*) *dghm-proj-ObjMap-vrange*:
  assumes $i \in_\circ I$
  shows $\mathcal{R}_\circ\ (\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor ObjMap \rfloor)) \subseteq_\circ \mathfrak{A}\ i(\lfloor Obj \rfloor)$
  ⟨*proof*⟩

Arrow map.

mk-VLambda *dghm-proj-components*(2)
  |*vsv dghm-proj-ArrMap-vsv*[*dg-cs-intros*]|
  |*vdomain dghm-proj-ArrMap-vdomain*[*dg-cs-simps*]|
  |*app dghm-proj-ArrMap-app*[*dg-cs-simps*]|

lemma (**in** *pdigraph*) *dghm-proj-ArrMap-vrange*:
  assumes $i \in_\circ I$
  shows $\mathcal{R}_\circ\ (\pi_{DG}\ I\ \mathfrak{A}\ i(\lfloor ArrMap \rfloor)) \subseteq_\circ \mathfrak{A}\ i(\lfloor Arr \rfloor)$
  ⟨*proof*⟩

A projection digraph homomorphism is a digraph homomorphism

lemma (**in** *pdigraph*) *pdg-dghm-proj-is-dghm*:
  assumes $i \in_\circ I$
  shows $\pi_{DG}\ I\ \mathfrak{A}\ i : (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i) \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
⟨*proof*⟩

lemma (**in** *pdigraph*) *pdg-dghm-proj-is-dghm′*:
  assumes $i \in_\circ I$ **and** $\mathfrak{C} = (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$ **and** $\mathfrak{D} = \mathfrak{A}\ i$
  shows $\pi_{DG}\ I\ \mathfrak{A}\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
  ⟨*proof*⟩

lemmas [*dg-cs-intros*] = *pdigraph.pdg-dghm-proj-is-dghm′*

3.8.8 Digraph product universal property digraph homomorphism

Definition and elementary properties

The following digraph homomorphism is used in the proof of the universal property of the product digraph later in this work.

definition *dghm-up* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi =$
$[$
$(\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ObjMap|\!)(\!|a|\!)))$,
$(\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ (\lambda i \in_\circ I.\ \varphi\ i(\!|ArrMap|\!)(\!|f|\!)))$,
$\mathfrak{C}$,
$(\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
$]_\circ$

Components.

lemma *dghm-up-components*:
shows *dghm-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 *dghm-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 *dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomDom|\!) = \mathfrak{C}$
and *dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomCod|\!) = (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Object map

mk-VLambda *dghm-up-components*(1)
|*vsv dghm-up-ObjMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-up-ObjMap-vdomain*[*dg-cs-simps*]|
|*app dghm-up-ObjMap-app*|

lemma *dghm-up-ObjMap-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ$ (*dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)) \subseteq_\circ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Obj|\!)$
⟨*proof*⟩

lemma *dghm-up-ObjMap-app-vdomain*[*dg-cs-simps*]:
assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $\mathcal{D}_\circ$ (*dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)(\!|a|\!)) = I$
⟨*proof*⟩

lemma *dghm-up-ObjMap-app-component*[*dg-cs-simps*]:
assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $i \in_\circ I$
shows *dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)(\!|a|\!)(\!|i|\!) = \varphi\ i(\!|ObjMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemma *dghm-up-ObjMap-app-vrange*:
assumes $a \in_\circ \mathfrak{C}(\!|Obj|\!)$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ$ (*dghm-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)(\!|a|\!)) \subseteq_\circ (\bigcup_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Obj|\!))$
⟨*proof*⟩

Arrow map

mk-VLambda *dghm-up-components*(2)
|*vsv dghm-up-ArrMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-up-ArrMap-vdomain*[*dg-cs-simps*]|
|*app dghm-up-ArrMap-app*|

lemma (in *pdigraph*) *dghm-up-ArrMap-vrange*:

assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!))\ \subseteq_\circ\ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *dghm-up-ArrMap-vrange*:
assumes $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!))\ \subseteq_\circ\ (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *dghm-up-ArrMap-app-vdomain*[*dg-cs-simps*]:
assumes $a \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $\mathcal{D}_\circ\ (dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!)(\!|a|\!)) = I$
⟨*proof*⟩

lemma *dghm-up-ArrMap-app-component*[*dg-cs-simps*]:
assumes $a \in_\circ \mathfrak{C}(\!|Arr|\!)$ **and** $i \in_\circ I$
shows $dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!)(\!|a|\!)(\!|i|\!) = \varphi\ i(\!|ArrMap|\!)(\!|a|\!)$
⟨*proof*⟩

lemma *dghm-up-ArrMap-app-vrange*:
assumes $a \in_\circ \mathfrak{C}(\!|Arr|\!)$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_\circ\ (dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!)(\!|a|\!))\ \subseteq_\circ\ (\bigcup_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!))$
⟨*proof*⟩

Digraph product universal property digraph homomorphism is a digraph homomorphism

lemma (**in** *pdigraph*) *pdg-dghm-up-is-dghm*:
assumes *digraph* $\alpha\ \mathfrak{C}$ **and** $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
shows $dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi : \mathfrak{C} \mapsto\mapsto_{DG\alpha} (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Further properties

lemma (**in** *pdigraph*) *pdg-dghm-comp-dghm-proj-dghm-up*:
assumes *digraph* $\alpha\ \mathfrak{C}$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{A}\ i$
and $i \in_\circ I$
shows $\varphi\ i = \pi_{DG}\ I\ \mathfrak{A}\ i \circ_{DGHM} dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi$
⟨*proof*⟩

lemma (**in** *pdigraph*) *pdg-dghm-up-eq-dghm-proj*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{DG\alpha} (\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i)$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow \varphi\ i = \pi_{DG}\ I\ \mathfrak{A}\ i \circ_{DGHM} \mathfrak{F}$
shows $dghm\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi = \mathfrak{F}$
⟨*proof*⟩

3.8.9 Singleton digraph

Object

lemma *dg-singleton-ObjI*:
assumes $A = set\ \{\langle j, a\rangle\}$ **and** $a \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $A \in_\circ (\prod_{DG} i \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-singleton-ObjE*:
assumes $A \in_\circ (\prod_{DG} i \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|Obj|\!)$
obtains a **where** $A = set\ \{\langle j, a\rangle\}$ **and** $a \in_\circ \mathfrak{C}(\!|Obj|\!)$

⟨*proof*⟩

Arrow

lemma *dg-singleton-ArrI*:
assumes $F = set\ \{\langle j,\ a\rangle\}$ **and** $a \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $F \in_\circ (\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-singleton-ArrE*:
assumes $F \in_\circ (\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})(\!|Arr|\!)$
obtains a **where** $F = set\ \{\langle j,\ a\rangle\}$ **and** $a \in_\circ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

Singleton digraph is a digraph

lemma (in *digraph*) *dg-finite-pdigraph-dg-singleton*:
assumes $j \in_\circ Vset\ \alpha$
shows *finite-pdigraph* α $(set\ \{j\})$ $(\lambda i.\ \mathfrak{C})$
⟨*proof*⟩

lemma (in *digraph*) *dg-digraph-dg-singleton*:
assumes $j \in_\circ Vset\ \alpha$
shows *digraph* α $(\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})$
⟨*proof*⟩

Arrow with a domain and a codomain

lemma (in *digraph*) *dg-singleton-is-arrI*:
assumes $j \in_\circ Vset\ \alpha$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows $set\ \{\langle j,\ f\rangle\} : set\ \{\langle j,\ a\rangle\} \mapsto_{(\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})} set\ \{\langle j,\ b\rangle\}$
⟨*proof*⟩

lemma (in *digraph*) *dg-singleton-is-arrD*:
assumes $set\ \{\langle j,\ f\rangle\} : set\ \{\langle j,\ a\rangle\} \mapsto_{(\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})} set\ \{\langle j,\ b\rangle\}$
and $j \in_\circ Vset\ \alpha$
shows $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (in *digraph*) *dg-singleton-is-arrE*:
assumes $set\ \{\langle j,\ f\rangle\} : set\ \{\langle j,\ a\rangle\} \mapsto_{(\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})} set\ \{\langle j,\ b\rangle\}$
and $j \in_\circ Vset\ \alpha$
obtains $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

3.8.10 Singleton digraph homomorphism

definition *dghm-singleton* :: $V \Rightarrow V \Rightarrow V$
where *dghm-singleton* j $\mathfrak{C}$ =
[
$(\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!).\ set\ \{\langle j,\ a\rangle\})$,
$(\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!).\ set\ \{\langle j,\ f\rangle\})$,
$\mathfrak{C}$,
$(\prod_{DG} j \in_\circ set\ \{j\}.\ \mathfrak{C})$
]$_\circ$

Components.

lemma *dghm-singleton-components*:

shows *dghm-singleton j* $\mathfrak{C}(\!|ObjMap|\!) = (\lambda a \in_{\circ} \mathfrak{C}(\!|Obj|\!).\ set\ \{\langle j,\ a\rangle\})$
and *dghm-singleton j* $\mathfrak{C}(\!|ArrMap|\!) = (\lambda f \in_{\circ} \mathfrak{C}(\!|Arr|\!).\ set\ \{\langle j,\ f\rangle\})$
and *dghm-singleton j* $\mathfrak{C}(\!|HomDom|\!) = \mathfrak{C}$
and *dghm-singleton j* $\mathfrak{C}(\!|HomCod|\!) = (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})$
⟨*proof*⟩

Object map

mk-VLambda *dghm-singleton-components*(1)
|*vsv dghm-singleton-ObjMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-singleton-ObjMap-vdomain*[*dg-cs-simps*]|
|*app dghm-singleton-ObjMap-app*[*dg-prod-cs-simps*]|

lemma *dghm-singleton-ObjMap-vrange*[*dg-cs-simps*]:
$\mathcal{R}_{\circ}$ (*dghm-singleton j* $\mathfrak{C}(\!|ObjMap|\!)$) = $(\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

Arrow map

mk-VLambda *dghm-singleton-components*(2)
|*vsv dghm-singleton-ArrMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-singleton-ArrMap-vdomain*[*dg-cs-simps*]|
|*app dghm-singleton-ArrMap-app*[*dg-prod-cs-simps*]|

lemma *dghm-singleton-ArrMap-vrange*[*dg-cs-simps*]:
$\mathcal{R}_{\circ}$ (*dghm-singleton j* $\mathfrak{C}(\!|ArrMap|\!)$) = $(\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

Singleton digraph homomorphism is an isomorphism of digraphs

lemma (in *digraph*) *dg-dghm-singleton-is-dghm*:
assumes $j \in_{\circ} Vset\ \alpha$
shows *dghm-singleton j* $\mathfrak{C} : \mathfrak{C} \mapsto\mapsto_{DG.iso\alpha} (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})$
⟨*proof*⟩

3.8.11 Product of two digraphs

Definition and elementary properties

See Chapter II-3 in [39].

definition *dg-prod-2* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\times_{DG}$› 80)
where $\mathfrak{A} \times_{DG} \mathfrak{B} \equiv$ *dg-prod* $(2_{\mathbb{N}})$ (*if2* $\mathfrak{A}$ $\mathfrak{B}$)

Product of two digraphs is a digraph

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *finite-pdigraph-dg-prod-2*: *finite-pdigraph* α $(2_{\mathbb{N}})$ (*if2* $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

interpretation *finite-pdigraph* α ‹$2_{\mathbb{N}}$› ‹*if2* $\mathfrak{A}$ $\mathfrak{B}$›

⟨*proof*⟩

lemma *digraph-dg-prod-2*[*dg-cs-intros*]: *digraph* α ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)
⟨*proof*⟩

end

Object

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

lemma *dg-prod-2-ObjI*:
assumes a $\in_\circ$ $\mathfrak{A}$(|*Obj*|) **and** b $\in_\circ$ $\mathfrak{B}$(|*Obj*|)
shows $[a, b]_\circ$ $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Obj*|)
⟨*proof*⟩

lemma *dg-prod-2-ObjI'*[*dg-prod-cs-intros*]:
assumes $ab = [a, b]_\circ$ **and** a $\in_\circ$ $\mathfrak{A}$(|*Obj*|) **and** b $\in_\circ$ $\mathfrak{B}$(|*Obj*|)
shows ab $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Obj*|)
⟨*proof*⟩

lemma *dg-prod-2-ObjE*:
assumes ab $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Obj*|)
obtains a b **where** $ab = [a, b]_\circ$ **and** a $\in_\circ$ $\mathfrak{A}$(|*Obj*|) **and** b $\in_\circ$ $\mathfrak{B}$(|*Obj*|)
⟨*proof*⟩

end

Arrow

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

lemma *dg-prod-2-ArrI*:
assumes g $\in_\circ$ $\mathfrak{A}$(|*Arr*|) **and** f $\in_\circ$ $\mathfrak{B}$(|*Arr*|)
shows $[g, f]_\circ$ $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Arr*|)
⟨*proof*⟩

lemma *dg-prod-2-ArrI'*[*dg-prod-cs-intros*]:
assumes $gf = [g, f]_\circ$ **and** g $\in_\circ$ $\mathfrak{A}$(|*Arr*|) **and** f $\in_\circ$ $\mathfrak{B}$(|*Arr*|)
shows $[g, f]_\circ$ $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Arr*|)
⟨*proof*⟩

lemma *dg-prod-2-ArrE*:
assumes gf $\in_\circ$ ($\mathfrak{A}$ $\times_{DG}$ $\mathfrak{B}$)(|*Arr*|)
obtains g f **where** $gf = [g, f]_\circ$ **and** g $\in_\circ$ $\mathfrak{A}$(|*Arr*|) **and** f $\in_\circ$ $\mathfrak{B}$(|*Arr*|)
⟨*proof*⟩

end

Arrow with a domain and a codomain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$

assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation *finite-pdigraph* α ‹$2_{\mathbb{N}}$› ‹*if*2 $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemma *dg-prod-2-is-arrI*:
assumes $g : a \mapsto_{\mathfrak{A}} c$ **and** $f : b \mapsto_{\mathfrak{B}} d$
shows $[g, f]_\circ : [a, b]_\circ \mapsto_{\mathfrak{A} \times_{DG} \mathfrak{B}} [c, d]_\circ$
⟨*proof*⟩

lemma *dg-prod-2-is-arrI'*[*dg-prod-cs-intros*]:
assumes $gf = [g, f]_\circ$
and $ab = [a, b]_\circ$
and $cd = [c, d]_\circ$
and $g : a \mapsto_{\mathfrak{A}} c$
and $f : b \mapsto_{\mathfrak{B}} d$
shows $gf : ab \mapsto_{\mathfrak{A} \times_{DG} \mathfrak{B}} cd$
⟨*proof*⟩

lemma *dg-prod-2-is-arrE*:
assumes $gf : ab \mapsto_{\mathfrak{A} \times_{DG} \mathfrak{B}} cd$
obtains $g\ f\ a\ b\ c\ d$
where $gf = [g, f]_\circ$
and $ab = [a, b]_\circ$
and $cd = [c, d]_\circ$
and $g : a \mapsto_{\mathfrak{A}} c$
and $f : b \mapsto_{\mathfrak{B}} d$
⟨*proof*⟩

end

Domain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

lemma *dg-prod-2-Dom-vsv*: *vsv* $((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Dom|\!))$
⟨*proof*⟩

lemma *dg-prod-2-Dom-vdomain*[*dg-cs-simps*]:
$\mathcal{D}_\circ\ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Dom|\!)) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-2-Dom-app*[*dg-prod-cs-simps*]:
assumes $[g, f]_\circ \in_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
shows $(\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Dom|\!)(\!|g, f|\!)_\bullet = [\mathfrak{A}(\!|Dom|\!)(\!|g|\!), \mathfrak{B}(\!|Dom|\!)(\!|f|\!)]_\circ$
⟨*proof*⟩

lemma *dg-prod-2-Dom-vrange*: $\mathcal{R}_\circ\ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Dom|\!)) \subseteq_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

end

Codomain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

lemma *dg-prod-2-Cod-vsv*: *vsv* $((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Cod|\!))$
⟨*proof*⟩

lemma *dg-prod-2-Cod-vdomain*[*dg-cs-simps*]:
$\mathcal{D}_\circ\ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Cod|\!)) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-2-Cod-app*[*dg-prod-cs-simps*]:
assumes $[g, f]_\circ \in_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
shows $(\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Cod|\!)(\!|g, f|\!)_\bullet = [\mathfrak{A}(\!|Cod|\!)(\!|g|\!), \mathfrak{B}(\!|Cod|\!)(\!|f|\!)]_\circ$
⟨*proof*⟩

lemma *dg-prod-2-Cod-vrange*: $\mathcal{R}_\circ\ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Cod|\!)) \subseteq_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

end

Opposite product digraph

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *dg-prod-2-op-dg-dg-Obj*[*dg-op-simps*]:
$(op\text{-}dg\ \mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-2-dg-op-dg-Obj*[*dg-op-simps*]:
$(\mathfrak{A} \times_{DG} op\text{-}dg\ \mathfrak{B})(\!|Obj|\!) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-2-op-dg-dg-Arr*[*dg-op-simps*]:
$(op\text{-}dg\ \mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-2-dg-op-dg-Arr*[*dg-op-simps*]:
$(\mathfrak{A} \times_{DG} op\text{-}dg\ \mathfrak{B})(\!|Arr|\!) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
⟨*proof*⟩

end

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$
begin

lemma *op-dg-dg-prod-2*[*dg-op-simps*]: $op\text{-}dg\ (\mathfrak{A} \times_{DG} \mathfrak{B}) = op\text{-}dg\ \mathfrak{A} \times_{DG} op\text{-}dg\ \mathfrak{B}$
⟨*proof*⟩

end

3.8.12 Projections for the product of two digraphs

Definition and elementary properties

definition *dghm-proj-fst* :: $V \Rightarrow V \Rightarrow V$ ($\langle \pi_{DG.1} \rangle$)
  where $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B}$ = *dghm-proj* ($2_{\mathbb{N}}$) (*if*2 $\mathfrak{A}\ \mathfrak{B}$) 0
definition *dghm-proj-snd* :: $V \Rightarrow V \Rightarrow V$ ($\langle \pi_{DG.2} \rangle$)
  where $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B}$ = *dghm-proj* ($2_{\mathbb{N}}$) (*if*2 $\mathfrak{A}\ \mathfrak{B}$) ($1_{\mathbb{N}}$)

Object map for a projection of a product of two digraphs

context
  fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}$
  assumes $\mathfrak{A}$: *digraph* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{B}$
begin

lemma *dghm-proj-fst-ObjMap-app*[*dg-cs-simps*]:
  assumes $[a, b]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
  shows $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B}(\!|ObjMap|\!)(\!|a, b|\!)_{\bullet} = a$
⟨*proof*⟩

lemma *dghm-proj-snd-ObjMap-app*[*dg-cs-simps*]:
  assumes $[a, b]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Obj|\!)$
  shows $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B}(\!|ObjMap|\!)(\!|a, b|\!)_{\bullet} = b$
⟨*proof*⟩

end

Arrow map for a projection of a product of two digraphs

context
  fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}$
  assumes $\mathfrak{A}$: *digraph* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{B}$
begin

lemma *dghm-proj-fst-ArrMap-app*[*dg-cs-simps*]:
  assumes $[g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
  shows $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B}(\!|ArrMap|\!)(\!|g, f|\!)_{\bullet} = g$
⟨*proof*⟩

lemma *dghm-proj-snd-ArrMap-app*[*dg-cs-simps*]:
  assumes $[g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\!|Arr|\!)$
  shows $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B}(\!|ArrMap|\!)(\!|g, f|\!)_{\bullet} = f$
⟨*proof*⟩

end

Domain and codomain of a projection of a product of two digraphs

lemma *dghm-proj-fst-HomDom*: $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B}(\!|HomDom|\!) = \mathfrak{A} \times_{DG} \mathfrak{B}$
  ⟨*proof*⟩

lemma *dghm-proj-fst-HomCod*: $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B}(\!|HomCod|\!) = \mathfrak{A}$
  ⟨*proof*⟩

lemma *dghm-proj-snd-HomDom*: $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B}(\!|HomDom|\!) = \mathfrak{A} \times_{DG} \mathfrak{B}$

⟨*proof*⟩

lemma *dghm-proj-snd-HomCod*: $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B}(\lfloor HomCod \rfloor) = \mathfrak{B}$
⟨*proof*⟩

Projection of a product of two digraphs is a digraph homomorphism

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}$
assumes $\mathfrak{A}$: *digraph* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{B}$
begin

interpretation *finite-pdigraph* α ‹$2_{\mathbb{N}}$› ‹*if*2 $\mathfrak{A}\ \mathfrak{B}$›
⟨*proof*⟩

lemma *dghm-proj-fst-is-dghm*:
assumes $i \in_\circ I$
shows $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{A} \times_{DG} \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *dghm-proj-fst-is-dghm'*[*dg-cs-intros*]:
assumes $i \in_\circ I$ **and** $\mathfrak{C} = \mathfrak{A} \times_{DG} \mathfrak{B}$ **and** $\mathfrak{D} = \mathfrak{A}$
shows $\pi_{DG.1}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *dghm-proj-snd-is-dghm*:
assumes $i \in_\circ I$
shows $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{A} \times_{DG} \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dghm-proj-snd-is-dghm'*[*dg-cs-intros*]:
assumes $i \in_\circ I$ **and** $\mathfrak{C} = \mathfrak{A} \times_{DG} \mathfrak{B}$ **and** $\mathfrak{D} = \mathfrak{B}$
shows $\pi_{DG.2}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathfrak{D}$
⟨*proof*⟩

end

3.8.13 Product of three digraphs

definition *dg-prod-3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (‹(- $\times_{DG3}$ - $\times_{DG3}$ -)› [81, 81, 81] 80)
where $\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C} = (\prod_{DG} i \in_\circ 3_{\mathbb{N}}.\ if3\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ i)$

Product of three digraphs is a digraph

context
fixes $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}$
assumes $\mathfrak{A}$: *digraph* $\alpha\ \mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{B}$ **and** $\mathfrak{C}$: *digraph* $\alpha\ \mathfrak{C}$
begin

interpretation $\mathcal{Z}\ \alpha$ ⟨*proof*⟩
interpretation $\mathfrak{A}$: *digraph* $\alpha\ \mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* $\alpha\ \mathfrak{C}$ ⟨*proof*⟩

lemma *finite-pdigraph-dg-prod-3*:
finite-pdigraph α ($3_{\mathbb{N}}$) (*if*3 $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}$)
⟨*proof*⟩

interpretation *finite-pdigraph* α ‹$3_{\mathbb{N}}$› ‹*if*3 $\mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}$›

⟨*proof*⟩

lemma *digraph-dg-prod-3*[*dg-cs-intros*]: *digraph* α $(\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})$
⟨*proof*⟩

end

Object

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *digraph* α $\mathfrak{C}$
begin

lemma *dg-prod-3-ObjI*:
assumes $a \in_\circ \mathfrak{A}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $[a, b, c]_\circ \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-3-ObjI'*[*dg-prod-cs-intros*]:
assumes $abc = [a, b, c]_\circ$ **and** $a \in_\circ \mathfrak{A}(\!|Obj|\!)$ **and** $b \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
shows $abc \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Obj|\!)$
⟨*proof*⟩

lemma *dg-prod-3-ObjE*:
assumes $abc \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Obj|\!)$
obtains a b c
where $abc = [a, b, c]_\circ$
and $a \in_\circ \mathfrak{A}(\!|Obj|\!)$
and $b \in_\circ \mathfrak{B}(\!|Obj|\!)$
and $c \in_\circ \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

end

Arrow

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *digraph* α $\mathfrak{C}$
begin

lemma *dg-prod-3-ArrI*:
assumes $h \in_\circ \mathfrak{A}(\!|Arr|\!)$ **and** $g \in_\circ \mathfrak{B}(\!|Arr|\!)$ **and** $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $[h, g, f]_\circ \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-3-ArrI'*[*dg-prod-cs-intros*]:
assumes $hgf = [h, g, f]_\circ$
and $h \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{B}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
shows $[h, g, f]_\circ \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Arr|\!)$
⟨*proof*⟩

lemma *dg-prod-3-ArrE*:
assumes $hgf \in_\circ (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\!|Arr|\!)$
obtains h g f

where $hgf = [h,\ g,\ f]_\circ$
and $h \in_\circ \mathfrak{A}(\!|Arr|\!)$
and $g \in_\circ \mathfrak{B}(\!|Arr|\!)$
and $f \in_\circ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

end

Arrow with a domain and a codomain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *digraph* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *digraph* α $\mathfrak{C}$ ⟨*proof*⟩
interpretation *finite-pdigraph* α ‹$3_{\mathbf{N}}$› ‹*if*3 $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemma *dg-prod-3-is-arrI*:
assumes $f : a \mapsto_{\mathfrak{A}} b$ **and** $f' : a' \mapsto_{\mathfrak{B}} b'$ **and** $f'' : a'' \mapsto_{\mathfrak{C}} b''$
shows $[f,\ f',\ f'']_\circ : [a,\ a',\ a'']_\circ \mapsto_{\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C}} [b,\ b',\ b'']_\circ$
⟨*proof*⟩

lemma *dg-prod-3-is-arrI'*[*dg-prod-cs-intros*]:
assumes $F = [f,\ f',\ f'']_\circ$
and $A = [a,\ a',\ a'']_\circ$
and $B = [b,\ b',\ b'']_\circ$
and $f : a \mapsto_{\mathfrak{A}} b$
and $f' : a' \mapsto_{\mathfrak{B}} b'$
and $f'' : a'' \mapsto_{\mathfrak{C}} b''$
shows $F : A \mapsto_{\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C}} B$
⟨*proof*⟩

lemma *dg-prod-3-is-arrE*:
assumes $F : A \mapsto_{\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C}} B$
obtains $f\ f'\ f''\ a\ a'\ a''\ b\ b'\ b''$
where $F = [f,\ f',\ f'']_\circ$
and $A = [a,\ a',\ a'']_\circ$
and $B = [b,\ b',\ b'']_\circ$
and $f : a \mapsto_{\mathfrak{A}} b$
and $f' : a' \mapsto_{\mathfrak{B}} b'$
and $f'' : a'' \mapsto_{\mathfrak{C}} b''$
⟨*proof*⟩

end

Domain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ **and** $\mathfrak{C}$: *digraph* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩

interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *digraph* α $\mathfrak{C}$ ⟨*proof*⟩

lemma *dg-prod-3-Dom-vsv*: *vsv* ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Dom*|))
⟨*proof*⟩

lemma *dg-prod-3-Dom-vdomain*[*dg-cs-simps*]:
$\mathcal{D}_\circ$ ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Dom*|)) = (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Arr*|)
⟨*proof*⟩

lemma *dg-prod-3-Dom-app*[*dg-prod-cs-simps*]:
assumes [*f*, *f'*, *f''*]$_\circ$ ∈$_\circ$ (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Arr*|)
shows (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Dom*|)(|*f*, *f'*, *f''*|)$_\bullet$ =
[𝔄(|*Dom*|)(|*f*|), 𝔅(|*Dom*|)(|*f'*|), ℭ(|*Dom*|)(|*f''*|)]$_\circ$
⟨*proof*⟩

lemma *dg-prod-3-Dom-vrange*:
$\mathcal{R}_\circ$ ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Dom*|)) ⊆$_\circ$ (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Obj*|)
⟨*proof*⟩

end

Codomain

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes 𝔄: *digraph* α $\mathfrak{A}$ **and** 𝔅: *digraph* α $\mathfrak{B}$ **and** ℭ: *digraph* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *digraph* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *digraph* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *digraph* α $\mathfrak{C}$ ⟨*proof*⟩

lemma *dg-prod-3-Cod-vsv*: *vsv* ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Cod*|))
⟨*proof*⟩

lemma *dg-prod-3-Cod-vdomain*[*dg-cs-simps*]:
$\mathcal{D}_\circ$ ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Cod*|)) = (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Arr*|)
⟨*proof*⟩

lemma *dg-prod-3-Cod-app*[*dg-prod-cs-simps*]:
assumes [*f*, *f'*, *f''*]$_\circ$ ∈$_\circ$ (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Arr*|)
shows
(𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Cod*|)(|*f*, *f'*, *f''*|)$_\bullet$ =
[𝔄(|*Cod*|)(|*f*|), 𝔅(|*Cod*|)(|*f'*|), ℭ(|*Cod*|)(|*f''*|)]$_\circ$
⟨*proof*⟩

lemma *dg-prod-3-Cod-vrange*:
$\mathcal{R}_\circ$ ((𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Cod*|)) ⊆$_\circ$ (𝔄 ×$_{DG3}$ 𝔅 ×$_{DG3}$ ℭ)(|*Obj*|)
⟨*proof*⟩

end

3.9 Subdigraph

3.9.1 Background

In this body of work, a subdigraph is a natural generalization of the concept of a subcategory, as defined in Chapter I-3 in [39], to digraphs. It should be noted that a similar concept also exists in the conventional graph theory, but further details are considered to be outside of the scope of this work.

named-theorems *dg-sub-cs-intros*
named-theorems *dg-sub-bw-cs-intros*
named-theorems *dg-sub-fw-cs-intros*
named-theorems *dg-sub-bw-cs-simps*

3.9.2 Simple subdigraph

Definition and elementary properties

locale *subdigraph* = *sdg*: *digraph* α $\mathfrak{B}$ + *dg*: *digraph* α $\mathfrak{C}$ **for** α $\mathfrak{B}$ $\mathfrak{C}$ +
  assumes *subdg-Obj-vsubset*[*dg-sub-fw-cs-intros*]:
    $a \in_\circ \mathfrak{B}(\!|Obj|\!) \Longrightarrow a \in_\circ \mathfrak{C}(\!|Obj|\!)$
    and *subdg-is-arr-vsubset*[*dg-sub-fw-cs-intros*]:
      $f : a \mapsto_{\mathfrak{B}} b \Longrightarrow f : a \mapsto_{\mathfrak{C}} b$

abbreviation *is-subdigraph* ($\langle$(-/ $\subseteq_{DG^1}$ -)$\rangle$ [51, 51] 50)
  where $\mathfrak{B} \subseteq_{DG\alpha} \mathfrak{C} \equiv$ *subdigraph* α $\mathfrak{B}$ $\mathfrak{C}$

lemmas [*dg-sub-fw-cs-intros*] =
  subdigraph.subdg-Obj-vsubset
  subdigraph.subdg-is-arr-vsubset

Rules.

lemma (**in** *subdigraph*) *subdigraph-axioms'*[*dg-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
  shows $\mathfrak{B}' \subseteq_{DG_{\alpha'}} \mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *subdigraph*) *subdigraph-axioms''*[*dg-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $\mathfrak{B} \subseteq_{DG_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *subdigraph-def*[*unfolded subdigraph-axioms-def*]
  |*intro subdigraphI*|
  |*dest subdigraphD*[*dest*]|
  |*elim subdigraphE*[*elim*!]|

lemmas [*dg-sub-cs-intros*] = *subdigraphD*(1,2)

The opposite subdigraph.

lemma (**in** *subdigraph*) *subdg-subdigraph-op-dg-op-dg*: *op-dg* $\mathfrak{B} \subseteq_{DG\alpha}$ *op-dg* $\mathfrak{C}$
⟨*proof*⟩

lemmas *subdg-subdigraph-op-dg-op-dg*[*dg-op-intros*] =
  subdigraph.subdg-subdigraph-op-dg-op-dg

Further rules.

lemma (**in** *subdigraph*) *subdg-objD*:

assumes $a \in_{\circ} \mathfrak{B}(\!|Obj|\!)$
shows $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$
⟨*proof*⟩

lemmas [*dg-sub-fw-cs-intros*] = *subdigraph.subdg-objD*

lemma (in *subdigraph*) *subdg-arrD*[*dg-sub-fw-cs-intros*]:
assumes $f \in_{\circ} \mathfrak{B}(\!|Arr|\!)$
shows $f \in_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

lemmas [*dg-sub-fw-cs-intros*] = *subdigraph.subdg-arrD*

lemma (in *subdigraph*) *subdg-dom-simp*:
assumes $f \in_{\circ} \mathfrak{B}(\!|Arr|\!)$
shows $\mathfrak{B}(\!|Dom|\!)(\!|f|\!) = \mathfrak{C}(\!|Dom|\!)(\!|f|\!)$
⟨*proof*⟩

lemmas [*dg-sub-bw-cs-simps*] = *subdigraph.subdg-dom-simp*

lemma (in *subdigraph*) *subdg-cod-simp*:
assumes $f \in_{\circ} \mathfrak{B}(\!|Arr|\!)$
shows $\mathfrak{B}(\!|Cod|\!)(\!|f|\!) = \mathfrak{C}(\!|Cod|\!)(\!|f|\!)$
⟨*proof*⟩

lemmas [*dg-sub-bw-cs-simps*] = *subdigraph.subdg-cod-simp*

lemma (in *subdigraph*) *subdg-is-arrD*:
assumes $f : a \mapsto_{\mathfrak{B}} b$
shows $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*dg-sub-fw-cs-intros*] = *subdigraph.subdg-is-arrD*

The subdigraph relation is a partial order

lemma *subdg-refl*:
assumes *digraph* α $\mathfrak{A}$
shows $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *subdg-trans*[*trans*]:
assumes $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{DG\alpha} \mathfrak{C}$
shows $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *subdg-antisym*:
assumes $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{DG\alpha} \mathfrak{A}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

3.9.3 Inclusion digraph homomorphism

Definition and elementary properties

See Chapter I-3 in [39].

definition *dghm-inc* :: $V \Rightarrow V \Rightarrow V$
where *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}$ = [*vid-on* ($\mathfrak{B}(\!|Obj|\!)$), *vid-on* ($\mathfrak{B}(\!|Arr|\!)$), $\mathfrak{B}$, $\mathfrak{C}$]$_{\circ}$

Components.

lemma *dghm-inc-components*:
shows *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}(\!|ObjMap|\!)$ = *vid-on* $(\mathfrak{B}(\!|Obj|\!))$
and *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}(\!|ArrMap|\!)$ = *vid-on* $(\mathfrak{B}(\!|Arr|\!))$
and [*dg-cs-simps*]: *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}(\!|HomDom|\!)$ = $\mathfrak{B}$
and [*dg-cs-simps*]: *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}(\!|HomCod|\!)$ = $\mathfrak{C}$
⟨*proof*⟩

Object map

mk-VLambda *dghm-inc-components*(1)[*folded VLambda-vid-on*]
|*vsv dghm-inc-ObjMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-inc-ObjMap-vdomain*[*dg-cs-simps*]|
|*app dghm-inc-ObjMap-app*[*dg-cs-simps*]|

Arrow map

mk-VLambda *dghm-inc-components*(2)[*folded VLambda-vid-on*]
|*vsv dghm-inc-ArrMap-vsv*[*dg-cs-intros*]|
|*vdomain dghm-inc-ArrMap-vdomain*[*dg-cs-simps*]|
|*app dghm-inc-ArrMap-app*[*dg-cs-simps*]|

Canonical inclusion digraph homomorphism associated with a subdigraph

sublocale *subdigraph* ⊆ *inc*: *is-ft-dghm* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*dghm-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemmas (**in** *subdigraph*) *subdg-dghm-inc-is-ft-dghm* = *inc.is-ft-dghm-axioms*

The inclusion digraph homomorphism for the opposite digraphs

lemma (**in** *subdigraph*) *subdg-dghm-inc-op-dg-is-dghm*[*dg-sub-cs-intros*]:
dghm-inc (*op-dg* $\mathfrak{B}$) (*op-dg* $\mathfrak{C}$) : *op-dg* $\mathfrak{B}$ $\mapsto\!\!\mapsto_{DG.faithful\alpha}$ *op-dg* $\mathfrak{C}$
⟨*proof*⟩

lemmas [*dg-sub-cs-intros*] = *subdigraph.subdg-dghm-inc-op-dg-is-dghm*

lemma (**in** *subdigraph*) *subdg-op-dg-dghm-inc*[*dg-op-simps*]:
op-dghm (*dghm-inc* $\mathfrak{B}$ $\mathfrak{C}$) = *dghm-inc* (*op-dg* $\mathfrak{B}$) (*op-dg* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*dg-op-simps*] = *subdigraph.subdg-op-dg-dghm-inc*

3.9.4 Full subdigraph

See Chapter I-3 in [39].

locale *fl-subdigraph* = *subdigraph* +
assumes *fl-subdg-is-fl-dghm-inc*: *dghm-inc* $\mathfrak{B}$ $\mathfrak{C}$: $\mathfrak{B}$ $\mapsto\!\!\mapsto_{DG.full\alpha}$ $\mathfrak{C}$

abbreviation *is-fl-subdigraph* (‹(-/ $\subseteq_{DG.full_1}$ -)› [51, 51] 50)
where $\mathfrak{B}$ $\subseteq_{DG.full\alpha}$ $\mathfrak{C}$ ≡ *fl-subdigraph* α $\mathfrak{B}$ $\mathfrak{C}$

sublocale *fl-subdigraph* ⊆ *inc*: *is-fl-dghm* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*dghm-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

Rules.

lemma (**in** *fl-subdigraph*) *fl-subdigraph-axioms'*[*dg-cs-intros*]:

assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{B}' \subseteq_{DG.full_{\alpha'}} \mathfrak{C}$
⟨*proof*⟩

lemma (in *fl-subdigraph*) *fl-subdigraph-axioms''*[*dg-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{B} \subseteq_{DG.full_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *fl-subdigraph-def*[*unfolded fl-subdigraph-axioms-def*]
|*intro fl-subdigraphI*|
|*dest fl-subdigraphD*[*dest*]|
|*elim fl-subdigraphE*[*elim*!]|

lemmas [*dg-sub-cs-intros*] = *fl-subdigraphD*(1)

Elementary properties.

lemma (in *fl-subdigraph*) *fl-subdg-Hom-eq*:
assumes $A \in_\circ \mathfrak{B}(\!|Obj|\!)$ **and** $B \in_\circ \mathfrak{B}(\!|Obj|\!)$
shows $Hom\ \mathfrak{B}\ A\ B = Hom\ \mathfrak{C}\ A\ B$
⟨*proof*⟩

3.9.5 Wide subdigraph

Definition and elementary properties

See [3][3]).

locale *wide-subdigraph* = *subdigraph* +
assumes *wide-subdg-Obj*[*dg-sub-bw-cs-intros*]: $a \in_\circ \mathfrak{C}(\!|Obj|\!) \Longrightarrow a \in_\circ \mathfrak{B}(\!|Obj|\!)$

abbreviation *is-wide-subdigraph* (‹(-/ $\subseteq_{DG.wide_1}$ -)› [51, 51] 50)
where $\mathfrak{B} \subseteq_{DG.wide_\alpha} \mathfrak{C} \equiv$ *wide-subdigraph* $\alpha\ \mathfrak{B}\ \mathfrak{C}$

lemmas [*dg-sub-bw-cs-intros*] = *wide-subdigraph.wide-subdg-Obj*

Rules.

lemma (in *wide-subdigraph*) *wide-subdigraph-axioms'*[*dg-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{B}' \subseteq_{DG.wide_{\alpha'}} \mathfrak{C}$
⟨*proof*⟩

lemma (in *wide-subdigraph*) *wide-subdigraph-axioms''*[*dg-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{B} \subseteq_{DG.wide_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *wide-subdigraph-def*[*unfolded wide-subdigraph-axioms-def*]
|*intro wide-subdigraphI*|
|*dest wide-subdigraphD*[*dest*]|
|*elim wide-subdigraphE*[*elim*!]|

lemmas [*dg-sub-cs-intros*] = *wide-subdigraphD*(1)

Elementary properties.

lemma (in *wide-subdigraph*) *wide-subdg-obj-eq*[*dg-sub-bw-cs-simps*]:

[3] https://ncatlab.org/nlab/show/wide+subcategory

$\mathfrak{B}(|Obj|) = \mathfrak{C}(|Obj|)$
⟨*proof*⟩

lemmas [*dg-sub-bw-cs-simps*] = *wide-subdigraph.wide-subdg-obj-eq*

The wide subdigraph relation is a partial order

lemma *wide-subdg-refl*:
assumes *digraph* α $\mathfrak{A}$
shows $\mathfrak{A} \subseteq_{DG.wide\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *wide-subdg-trans*[*trans*]:
assumes $\mathfrak{A} \subseteq_{DG.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{DG.wide\alpha}$ $\mathfrak{C}$
shows $\mathfrak{A} \subseteq_{DG.wide\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *wide-subdg-antisym*:
assumes $\mathfrak{A} \subseteq_{DG.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{DG.wide\alpha}$ $\mathfrak{A}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

3.10 Simple digraphs

3.10.1 Background

The section presents a variety of simple digraphs, such as the empty digraph 0 and a digraph with one object and one arrow 1. All of the entities presented in this section are generalizations of certain simple categories, whose definitions can be found in [39].

3.10.2 Empty digraph 0

Definition and elementary properties

See Chapter I-2 in [39].

definition *dg*-0 :: *V*
where *dg*-0 = [0, 0, 0, 0]$_\circ$

Components.

lemma *dg*-0-*components*:
shows *dg*-0(|*Obj*|) = 0
and *dg*-0(|*Arr*|) = 0
and *dg*-0(|*Dom*|) = 0
and *dg*-0(|*Cod*|) = 0
⟨*proof*⟩

0 is a digraph

lemma (**in** $\mathcal{Z}$) *digraph-dg*-0[*dg-cs-intros*]: *digraph* α *dg*-0
⟨*proof*⟩

lemmas [*dg-cs-intros*] = $\mathcal{Z}$.*digraph-dg*-0

Opposite of the digraph 0

lemma *op-dg-dg*-0[*dg-op-simps*]: *op-dg* (*dg*-0) = *dg*-0
⟨*proof*⟩

Arrow with a domain and a codomain

lemma *dg*-0-*is-arr-iff*[*simp*]: $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}0} \mathfrak{B} \longleftrightarrow$ *False*
⟨*proof*⟩

A digraph without objects is empty

lemma (**in** *digraph*) *dg-dg*-0-*if-Obj*-0:
assumes $\mathfrak{C}$(|*Obj*|) = 0
shows $\mathfrak{C}$ = *dg*-0
⟨*proof*⟩

3.10.3 Empty digraph homomorphism

Definition and elementary properties

definition *dghm*-0 :: *V* ⇒ *V*
where *dghm*-0 $\mathfrak{A}$ = [0, 0, *dg*-0, $\mathfrak{A}$]$_\circ$

Components.

lemma *dghm*-0-*components*:
shows *dghm*-0 $\mathfrak{A}$(|*ObjMap*|) = 0

and *dghm*-0 $\mathfrak{A}(\!|ArrMap|\!) = 0$
and *dghm*-0 $\mathfrak{A}(\!|HomDom|\!) = dg\text{-}0$
and *dghm*-0 $\mathfrak{A}(\!|HomCod|\!) = \mathfrak{A}$
⟨*proof*⟩

Opposite empty digraph homomorphism.

lemma *op-dghm-dghm*-0: *op-dghm* (*dghm*-0 $\mathfrak{C}$) = *dghm*-0 (*op-dg* $\mathfrak{C}$)
⟨*proof*⟩

Object map

lemma *dghm*-0-*ObjMap-vsv*[*dg-cs-intros*]: *vsv* (*dghm*-0 $\mathfrak{C}(\!|ObjMap|\!)$)
⟨*proof*⟩

Arrow map

lemma *dghm*-0-*ArrMap-vsv*[*dg-cs-intros*]: *vsv* (*dghm*-0 $\mathfrak{C}(\!|ArrMap|\!)$)
⟨*proof*⟩

Empty digraph homomorphism is a faithful digraph homomorphism

lemma (**in** $\mathcal{Z}$) *dghm*-0-*is-ft-dghm*:
assumes *digraph* α $\mathfrak{A}$
shows *dghm*-0 $\mathfrak{A}$: *dg*-0 $\mapsto\!\!\mapsto_{DG.faithful\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *dghm*-0-*is-ft-dghm'*[*dghm-cs-intros*]:
assumes *digraph* α $\mathfrak{A}$
and $\mathfrak{B}' = \mathfrak{A}$
and $\mathfrak{A}' =$ *dg*-0
shows *dghm*-0 $\mathfrak{A}$: $\mathfrak{A}' \mapsto\!\!\mapsto_{DG.faithful\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*dghm-cs-intros*] = $\mathcal{Z}$.*dghm*-0-*is-ft-dghm'*

lemma (**in** $\mathcal{Z}$) *dghm*-0-*is-dghm*:
assumes *digraph* α $\mathfrak{A}$
shows *dghm*-0 $\mathfrak{A}$: *dg*-0 $\mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *dghm*-0-*is-dghm'*[*dg-cs-intros*]:
assumes *digraph* α $\mathfrak{A}$
and $\mathfrak{B}' = \mathfrak{A}$
and $\mathfrak{A}' =$ *dg*-0
shows *dghm*-0 $\mathfrak{A}$: $\mathfrak{A}' \mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = $\mathcal{Z}$.*dghm*-0-*is-dghm'*

Further properties

lemma *is-dghm-is-dghm*-0-*if-dg*-0:
assumes $\mathfrak{F}$: *dg*-0 $\mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{C}$
shows $\mathfrak{F}$ = *dghm*-0 $\mathfrak{C}$
⟨*proof*⟩

3.10.4 Empty transformation of digraph homomorphisms

Definition and elementary properties

definition *tdghm*-0 :: $V \Rightarrow V$
where *tdghm*-0 $\mathfrak{C}$ = [0, *dghm*-0 $\mathfrak{C}$, *dghm*-0 $\mathfrak{C}$, *dg*-0, $\mathfrak{C}$]$_\circ$

Components.

lemma *tdghm*-0-*components*:
shows *tdghm*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$ = 0
and [*dg-cs-simps*]: *tdghm*-0 $\mathfrak{C}(\lfloor NTDom \rfloor)$ = *dghm*-0 $\mathfrak{C}$
and [*dg-cs-simps*]: *tdghm*-0 $\mathfrak{C}(\lfloor NTCod \rfloor)$ = *dghm*-0 $\mathfrak{C}$
and [*dg-cs-simps*]: *tdghm*-0 $\mathfrak{C}(\lfloor NTDGDom \rfloor)$ = *dg*-0
and [*dg-cs-simps*]: *tdghm*-0 $\mathfrak{C}(\lfloor NTDGCod \rfloor)$ = $\mathfrak{C}$
⟨*proof*⟩

Duality.

lemma *op-tdghm-tdghm*-0: *op-tdghm* (*tdghm*-0 $\mathfrak{C}$) = *tdghm*-0 (*op-dg* $\mathfrak{C}$)
⟨*proof*⟩

Transformation map

lemma *tdghm*-0-*NTMap-vsv*[*dg-cs-intros*]: *vsv* (*tdghm*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$)
⟨*proof*⟩

lemma *tdghm*-0-*NTMap-vdomain*[*dg-cs-simps*]: $\mathcal{D}_\circ$ (*tdghm*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$) = 0
⟨*proof*⟩

lemma *tdghm*-0-*NTMap-vrange*[*dg-cs-simps*]: $\mathcal{R}_\circ$ (*tdghm*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$) = 0
⟨*proof*⟩

Empty transformation of digraph homomorphisms is a transformation of digraph homomorphisms

lemma (**in** *digraph*) *dg-tdghm*-0-*is-tdghmI*:
tdghm-0 $\mathfrak{C}$: *dghm*-0 $\mathfrak{C}$ $\mapsto_{DGHM}$ *dghm*-0 $\mathfrak{C}$: *dg*-0 $\mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *digraph*) *dg-tdghm*-0-*is-tdghmI'*[*dg-cs-intros*]:
assumes $\mathfrak{F}'$ = *dghm*-0 $\mathfrak{C}$
and $\mathfrak{G}'$ = *dghm*-0 $\mathfrak{C}$
and $\mathfrak{A}'$ = *dg*-0
and $\mathfrak{B}'$ = $\mathfrak{C}$
and $\mathfrak{F}'$ = $\mathfrak{F}$
and $\mathfrak{G}'$ = $\mathfrak{G}$
shows *tdghm*-0 $\mathfrak{C}$: $\mathfrak{F}'$ $\mapsto_{DGHM}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

lemmas [*dg-cs-intros*] = *digraph.dg-tdghm*-0-*is-tdghmI'*

lemma *is-tdghm-is-tdghm*-0-*if-dg*-0:
assumes $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{DGHM}$ $\mathfrak{G}$: *dg*-0 $\mapsto\!\!\mapsto_{DG\alpha}$ $\mathfrak{C}$
shows $\mathfrak{N}$ = *tdghm*-0 $\mathfrak{C}$ **and** $\mathfrak{F}$ = *dghm*-0 $\mathfrak{C}$ **and** $\mathfrak{G}$ = *dghm*-0 $\mathfrak{C}$
⟨*proof*⟩

3.10.5 10: digraph with one object and no arrows

Definition and elementary properties

definition *dg*-10 :: $V \Rightarrow V$

where *dg*-10 𝔞 = [*set* {𝔞}, 0, 0, 0]∘

Components.

lemma *dg*-10-*components*:
shows *dg*-10 𝔞(|*Obj*|) = *set* {𝔞}
and *dg*-10 𝔞(|*Arr*|) = 0
and *dg*-10 𝔞(|*Dom*|) = 0
and *dg*-10 𝔞(|*Cod*|) = 0
⟨*proof*⟩

10 is a digraph

lemma (**in** 𝒵) *digraph-dg*-10:
assumes 𝔞 ∈∘ *Vset* α
shows *digraph* α (*dg*-10 𝔞)
⟨*proof*⟩

Arrow with a domain and a codomain

lemma *dg*-10-*is-arr-iff*: 𝔉 : 𝔄 ↦*dg*-10 𝔞 𝔅 ⟷ *False*
⟨*proof*⟩

3.10.6 1: digraph with one object and one arrow

Definition and elementary properties

definition *dg*-1 :: *V* ⇒ *V* ⇒ *V*
where *dg*-1 𝔞 𝔣 = [*set* {𝔞}, *set* {𝔣}, *set* {⟨𝔣, 𝔞⟩}, *set* {⟨𝔣, 𝔞⟩}]∘

Components.

lemma *dg*-1-*components*:
shows *dg*-1 𝔞 𝔣(|*Obj*|) = *set* {𝔞}
and *dg*-1 𝔞 𝔣(|*Arr*|) = *set* {𝔣}
and *dg*-1 𝔞 𝔣(|*Dom*|) = *set* {⟨𝔣, 𝔞⟩}
and *dg*-1 𝔞 𝔣(|*Cod*|) = *set* {⟨𝔣, 𝔞⟩}
⟨*proof*⟩

1 is a digraph

lemma (**in** 𝒵) *digraph-dg*-1:
assumes 𝔞 ∈∘ *Vset* α **and** 𝔣 ∈∘ *Vset* α
shows *digraph* α (*dg*-1 𝔞 𝔣)
⟨*proof*⟩

Arrow with a domain and a codomain

lemma *dg*-1-*is-arrI*:
assumes *a* = 𝔞 **and** *b* = 𝔞 **and** *f* = 𝔣
shows *f* : *a* ↦*dg*-1 𝔞 𝔣 *b*
⟨*proof*⟩

lemma *dg*-1-*is-arrD*:
assumes *f* : *a* ↦*dg*-1 𝔞 𝔣 *b*
shows *a* = 𝔞 **and** *b* = 𝔞 **and** *f* = 𝔣
⟨*proof*⟩

lemma *dg*-1-*is-arrE*:
assumes *f* : *a* ↦*dg*-1 𝔞 𝔣 *b*

obtains $a = \mathfrak{a}$ **and** $b = \mathfrak{a}$ **and** $f = \mathfrak{f}$
⟨*proof*⟩

lemma *dg-1-is-arr-iff*: $f : a \mapsto_{dg\text{-}1\ \mathfrak{a}\ \mathfrak{f}} b \longleftrightarrow (a = \mathfrak{a} \wedge b = \mathfrak{a} \wedge f = \mathfrak{f})$
⟨*proof*⟩

3.11 *GRPH* as a digraph

3.11.1 Background

Conventionally, *GRPH* defined as a category of digraphs and digraph homomorphisms (e.g., see Chapter II-7 in [39]). However, there is little that can prevent one from exposing *GRPH* as a digraph and provide additional structure gradually later. Thus, in this section, α-*GRPH* is defined as a digraph of digraphs and digraph homomorphisms in V_α.

named-theorems *GRPH-cs-simps*
named-theorems *GRPH-cs-intros*

3.11.2 Definition and elementary properties

definition *dg-GRPH* :: $V \Rightarrow V$
  where *dg-GRPH* α =
    [
      set $\{\mathfrak{C}.$ *digraph* α $\mathfrak{C}\}$,
      all-dghms α,
      $(\lambda\mathfrak{F}{\in_\circ}$*all-dghms* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$,
      $(\lambda\mathfrak{F}{\in_\circ}$*all-dghms* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$
    ]$_\circ$

Components.

lemma *dg-GRPH-components*:
  shows *dg-GRPH* $\alpha(\!|$*Obj*$|\!)$ = *set* $\{\mathfrak{C}.$ *digraph* α $\mathfrak{C}\}$
    and *dg-GRPH* $\alpha(\!|$*Arr*$|\!)$ = *all-dghms* α
    and *dg-GRPH* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda\mathfrak{F}{\in_\circ}$*all-dghms* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$
    and *dg-GRPH* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda\mathfrak{F}{\in_\circ}$*all-dghms* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$
  ⟨*proof*⟩

3.11.3 Object

lemma *dg-GRPH-ObjI*:
  assumes *digraph* α $\mathfrak{A}$
  shows $\mathfrak{A}$ $\in_\circ$ *dg-GRPH* $\alpha(\!|$*Obj*$|\!)$
  ⟨*proof*⟩

lemma *dg-GRPH-ObjD*:
  assumes $\mathfrak{A}$ $\in_\circ$ *dg-GRPH* $\alpha(\!|$*Obj*$|\!)$
  shows *digraph* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-GRPH-ObjE*:
  assumes $\mathfrak{A}$ $\in_\circ$ *dg-GRPH* $\alpha(\!|$*Obj*$|\!)$
  obtains *digraph* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-GRPH-Obj-iff*[*GRPH-cs-simps*]:
  $\mathfrak{A}$ $\in_\circ$ *dg-GRPH* $\alpha(\!|$*Obj*$|\!)$ ⟷ *digraph* α $\mathfrak{A}$
  ⟨*proof*⟩

3.11.4 Domain

mk-VLambda *dg-GRPH-components*(3)
  |*vsv dg-GRPH-Dom-vsv*[*GRPH-cs-intros*]|
  |*vdomain dg-GRPH-Dom-vdomain*[*GRPH-cs-simps*]|
  |*app dg-GRPH-Dom-app*[*GRPH-cs-simps*]|

lemma *dg-GRPH-Dom-vrange*: $\mathcal{R}_{\circ}$ (*dg-GRPH* $\alpha(\!|Dom|\!)$) $\subseteq_{\circ}$ *dg-GRPH* $\alpha(\!|Obj|\!)$
⟨*proof*⟩

3.11.5 Codomain

mk-VLambda *dg-GRPH-components*(4)
|*vsv dg-GRPH-Cod-vsv*[*GRPH-cs-intros*]|
|*vdomain dg-GRPH-Cod-vdomain*[*GRPH-cs-simps*]|
|*app dg-GRPH-Cod-app*[*GRPH-cs-simps*]|

lemma *dg-GRPH-Cod-vrange*: $\mathcal{R}_{\circ}$ (*dg-GRPH* $\alpha(\!|Cod|\!)$) $\subseteq_{\circ}$ *dg-GRPH* $\alpha(\!|Obj|\!)$
⟨*proof*⟩

3.11.6 *GRPH* is a digraph

lemma (**in** $\mathcal{Z}$) *tiny-digraph-dg-GRPH*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$
shows *tiny-digraph* β (*dg-GRPH* α)
⟨*proof*⟩

3.11.7 Arrow with a domain and a codomain

lemma *dg-GRPH-is-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}GRPH\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-GRPH-is-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}GRPH\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-GRPH-is-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}GRPH\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-GRPH-is-arr-iff*[*GRPH-cs-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}GRPH\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{DG\alpha} \mathfrak{B}$
⟨*proof*⟩

3.12 *Rel* as a digraph

3.12.1 Background

Rel is usually defined as a category of sets and binary relations (e.g., see Chapter I-7 in [39]). However, there is little that can prevent one from exposing *Rel* as a digraph and provide additional structure gradually later. Thus, in this section, α-*Rel* is defined as a digraph of sets and binary relations in V_α.

named-theorems *dg-Rel-shared-cs-simps*
named-theorems *dg-Rel-shared-cs-intros*

named-theorems *dg-Rel-cs-simps*
named-theorems *dg-Rel-cs-intros*

3.12.2 Canonical arrow for V

named-theorems *arr-field-simps*

definition *ArrVal* :: *V* **where** [*arr-field-simps*]: *ArrVal* = 0
definition *ArrDom* :: *V* **where** [*arr-field-simps*]: *ArrDom* = $1_{\mathbb{N}}$
definition *ArrCod* :: *V* **where** [*arr-field-simps*]: *ArrCod* = $2_{\mathbb{N}}$

lemma *ArrVal-eq-helper*:
assumes *f* = *g*
shows *f*(|*ArrVal*|)(|*a*|) = *g*(|*ArrVal*|)(|*a*|)
⟨*proof*⟩

3.12.3 Arrow for *Rel*

Definition and elementary properties

locale *arr-Rel* = $\mathcal{Z}$ α + *vfsequence T* + *ArrVal*: *vbrelation* ‹*T*(|*ArrVal*|)› **for** α *T* +
assumes *arr-Rel-length*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
vcard T = $3_{\mathbb{N}}$
and *arr-Rel-ArrVal-vdomain*: $\mathcal{D}_\circ$ (*T*(|*ArrVal*|)) $\subseteq_\circ$ *T*(|*ArrDom*|)
and *arr-Rel-ArrVal-vrange*: $\mathcal{R}_\circ$ (*T*(|*ArrVal*|)) $\subseteq_\circ$ *T*(|*ArrCod*|)
and *arr-Rel-ArrDom-in-Vset*: *T*(|*ArrDom*|) $\in_\circ$ *Vset* α
and *arr-Rel-ArrCod-in-Vset*: *T*(|*ArrCod*|) $\in_\circ$ *Vset* α

lemmas [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*] = *arr-Rel.arr-Rel-length*

Components.

lemma *arr-Rel-components*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
shows [*f*, *A*, *B*]$_\circ$(|*ArrVal*|) = *f*
and [*f*, *A*, *B*]$_\circ$(|*ArrDom*|) = *A*
and [*f*, *A*, *B*]$_\circ$(|*ArrCod*|) = *B*
⟨*proof*⟩

Rules.

lemma (**in** *arr-Rel*) *arr-Rel-axioms'*[*dg-cs-intros*, *dg-Rel-cs-intros*]:
assumes $\alpha' = \alpha$
shows *arr-Rel* α' *T*
⟨*proof*⟩

mk-ide rf *arr-Rel-def*[*unfolded arr-Rel-axioms-def*]
|*intro arr-RelI*|
|*dest arr-RelD*[*dest*]|

|*elim arr-RelE*[*elim*!]|

lemma (in $\mathcal{Z}$**)** *arr-Rel-vfsequenceI*:
assumes *vbrelation r*
and $\mathcal{D}_{\circ}\ r \subseteq_{\circ} a$
and $\mathcal{R}_{\circ}\ r \subseteq_{\circ} b$
and $a \in_{\circ} Vset\ \alpha$
and $b \in_{\circ} Vset\ \alpha$
shows *arr-Rel* $\alpha\ [r, a, b]_{\circ}$
⟨*proof*⟩

Elementary properties.

lemma *arr-Rel-eqI*:
assumes *arr-Rel* $\alpha\ S$
and *arr-Rel* $\alpha\ T$
and $S(\!|ArrVal|\!) = T(\!|ArrVal|\!)$
and $S(\!|ArrDom|\!) = T(\!|ArrDom|\!)$
and $S(\!|ArrCod|\!) = T(\!|ArrCod|\!)$
shows $S = T$
⟨*proof*⟩

lemma (in *arr-Rel*) *arr-Rel-def*: $T = [T(\!|ArrVal|\!), T(\!|ArrDom|\!), T(\!|ArrCod|\!)]_{\circ}$
⟨*proof*⟩

Size.

lemma (in *arr-Rel*) *arr-Rel-ArrVal-in-Vset*: $T(\!|ArrVal|\!) \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (in *arr-Rel*) *arr-Rel-in-Vset*: $T \in_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma *small-arr-Rel*[*simp*]: *small* $\{T.\ arr\text{-}Rel\ \alpha\ T\}$
⟨*proof*⟩

Other elementary properties.

lemma *set-Collect-arr-Rel*[*simp*]:
$x \in_{\circ} set\ (Collect\ (arr\text{-}Rel\ \alpha)) \longleftrightarrow arr\text{-}Rel\ \alpha\ x$
⟨*proof*⟩

lemma (in *arr-Rel*) *arr-Rel-ArrVal-vsubset-ArrDom-ArrCod*:
$T(\!|ArrVal|\!) \subseteq_{\circ} T(\!|ArrDom|\!) \times_{\circ} T(\!|ArrCod|\!)$
⟨*proof*⟩

Composition

See Chapter I-7 in [39].

definition *comp-Rel* :: $V \Rightarrow V \Rightarrow V$ **(infixl** $\langle \circ_{Rel} \rangle$ 55)
where $comp\text{-}Rel\ S\ T = [S(\!|ArrVal|\!) \circ_{\circ} T(\!|ArrVal|\!), T(\!|ArrDom|\!), S(\!|ArrCod|\!)]_{\circ}$

Components.

lemma *comp-Rel-components*:
shows $(S \circ_{Rel} T)(\!|ArrVal|\!) = S(\!|ArrVal|\!) \circ_{\circ} T(\!|ArrVal|\!)$
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
$(S \circ_{Rel} T)(\!|ArrDom|\!) = T(\!|ArrDom|\!)$
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
$(S \circ_{Rel} T)(\!|ArrCod|\!) = S(\!|ArrCod|\!)$

⟨*proof*⟩

Elementary properties.

lemma *comp-Rel-vsv*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:
vsv (*S* ∘$_{Rel}$ *T*)
⟨*proof*⟩

lemma *arr-Rel-comp-Rel*[*dg-Rel-cs-intros*]:
assumes *arr-Rel* α *S* **and** *arr-Rel* α *T*
shows *arr-Rel* α (*S* ∘$_{Rel}$ *T*)
⟨*proof*⟩

lemma *arr-Rel-comp-Rel-assoc*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
(*H* ∘$_{Rel}$ *G*) ∘$_{Rel}$ *F* = *H* ∘$_{Rel}$ (*G* ∘$_{Rel}$ *F*)
⟨*proof*⟩

Inclusion arrow

The definition of the inclusion arrow is based on the concept of the inclusion map, e.g., see [5][4]

definition *incl-Rel A B* = [*vid-on A*, *A*, *B*]$_{\circ}$

Components.

lemma *incl-Rel-components*:
shows *incl-Rel A B*(|*ArrVal*|) = *vid-on A*
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *incl-Rel A B*(|*ArrDom*|) = *A*
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *incl-Rel A B*(|*ArrCod*|) = *B*
⟨*proof*⟩

Arrow value.

lemma *incl-Rel-ArrVal-vsv*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:
vsv (*incl-Rel A B*(|*ArrVal*|))
⟨*proof*⟩

lemma *incl-Rel-ArrVal-vdomain*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
$\mathcal{D}_{\circ}$ (*incl-Rel A B*(|*ArrVal*|)) = *A*
⟨*proof*⟩

lemma *incl-Rel-ArrVal-app*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
assumes *a* ∈$_{\circ}$ *A*
shows *incl-Rel A B*(|*ArrVal*|)(|*a*|) = *a*
⟨*proof*⟩

Elementary properties.

lemma *incl-Rel-vfsequence*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:
vfsequence (*incl-Rel A B*)
⟨*proof*⟩

lemma *incl-Rel-vcard*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
vcard (*incl-Rel A B*) = $3_{\mathbb{N}}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *arr-Rel-incl-RelI*:
assumes *A* ∈$_{\circ}$ *Vset* α **and** *B* ∈$_{\circ}$ *Vset* α **and** *A* ⊆$_{\circ}$ *B*
shows *arr-Rel* α (*incl-Rel A B*)
⟨*proof*⟩

[4] https://en.wikipedia.org/wiki/Inclusion_map

Identity

See Chapter I-7 in [39].

definition *id-Rel* :: $V \Rightarrow V$
where *id-Rel A* = *incl-Rel A A*

Components.

lemma *id-Rel-components*:
shows *id-Rel A*(|*ArrVal*|) = *vid-on A*
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *id-Rel A*(|*ArrDom*|) = *A*
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *id-Rel A*(|*ArrCod*|) = *A*
⟨*proof*⟩

Elementary properties.

lemma *id-Rel-vfsequence*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:
vfsequence (*id-Rel A*)
⟨*proof*⟩

lemma *id-Rel-vcard*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
vcard (*id-Rel A*) = $3_{\mathbb{N}}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *arr-Rel-id-RelI*:
assumes $A \in_{\circ}$ *Vset* α
shows *arr-Rel* α (*id-Rel A*)
⟨*proof*⟩

lemma *id-Rel-ArrVal-app*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
assumes $a \in_{\circ} A$
shows *id-Rel A*(|*ArrVal*|)(|*a*|) = *a*
⟨*proof*⟩

lemma *arr-Rel-comp-Rel-id-Rel-left*[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α *F* **and** *F*(|*ArrCod*|) = *A*
shows *id-Rel A* $\circ_{Rel}$ *F* = *F*
⟨*proof*⟩

lemma *arr-Rel-comp-Rel-id-Rel-right*[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α *F* **and** *F*(|*ArrDom*|) = *A*
shows *F* $\circ_{Rel}$ *id-Rel A* = *F*
⟨*proof*⟩

Converse

As mentioned in Chapter I-7 in [39], the category *Rel* is usually equipped with an additional structure that is the operation of taking a converse of a relation. The operation is meant to be used almost exclusively as part of the dagger functor for *Rel*.

definition *converse-Rel* :: $V \Rightarrow V$ (‹(-$^{-1}{}_{Rel}$)› [1000] 999)
where *converse-Rel T* = [(*T*(|*ArrVal*|))$^{-1}{}_{\circ}$, *T*(|*ArrCod*|), *T*(|*ArrDom*|)]$_{\circ}$

lemma *converse-Rel-components*:
shows $T^{-1}{}_{Rel}$(|*ArrVal*|) = (*T*(|*ArrVal*|))$^{-1}{}_{\circ}$
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: $T^{-1}{}_{Rel}$(|*ArrDom*|) = *T*(|*ArrCod*|)
and [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: $T^{-1}{}_{Rel}$(|*ArrCod*|) = *T*(|*ArrDom*|)
⟨*proof*⟩

Elementary properties.

lemma (**in** *arr-Rel*) *arr-Rel-converse-Rel*: *arr-Rel* α $(T^{-1}{}_{Rel})$
⟨*proof*⟩

lemmas [*dg-Rel-cs-intros*] =
arr-Rel.arr-Rel-converse-Rel

lemma (**in** *arr-Rel*)
arr-Rel-converse-Rel-converse-Rel[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:
$(T^{-1}{}_{Rel})^{-1}{}_{Rel} = T$
⟨*proof*⟩

lemmas [*dg-Rel-cs-simps*] =
arr-Rel.arr-Rel-converse-Rel-converse-Rel

lemma *arr-Rel-converse-Rel-eq-iff*[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α F **and** *arr-Rel* α G
shows $F^{-1}{}_{Rel} = G^{-1}{}_{Rel} \longleftrightarrow F = G$
⟨*proof*⟩

lemma *arr-Rel-converse-Rel-comp-Rel*[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α G **and** *arr-Rel* α F
shows $(F \circ_{Rel} G)^{-1}{}_{Rel} = G^{-1}{}_{Rel} \circ_{Rel} F^{-1}{}_{Rel}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *arr-Rel-converse-Rel-id-Rel*:
assumes $c \in_{\circ}$ *Vset* α
shows *arr-Rel* α $((id\text{-}Rel\ c)^{-1}{}_{Rel})$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *arr-Rel-converse-Rel-id-Rel-eq-id-Rel*[
dg-Rel-shared-cs-simps, *dg-Rel-cs-simps*
]:
assumes $c \in_{\circ}$ *Vset* α
shows $(id\text{-}Rel\ c)^{-1}{}_{Rel} = id\text{-}Rel\ c$
⟨*proof*⟩

lemmas [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*] =
$\mathcal{Z}$.*arr-Rel-converse-Rel-id-Rel-eq-id-Rel*

lemma *arr-Rel-comp-Rel-converse-Rel-left-if-v*11[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α T
and $\mathcal{D}_{\circ}$ $(T(\!|ArrVal|\!)) = A$
and $T(\!|ArrDom|\!) = A$
and *v*11 $(T(\!|ArrVal|\!))$
and $A \in_{\circ}$ *Vset* α
shows $T^{-1}{}_{Rel} \circ_{Rel} T = id\text{-}Rel\ A$
⟨*proof*⟩

lemma *arr-Rel-comp-Rel-converse-Rel-right-if-v*11[*dg-Rel-cs-simps*]:
assumes *arr-Rel* α T
and $\mathcal{R}_{\circ}$ $(T(\!|ArrVal|\!)) = A$
and $T(\!|ArrCod|\!) = A$
and *v*11 $(T(\!|ArrVal|\!))$
and $A \in_{\circ}$ *Vset* α
shows $T \circ_{Rel} T^{-1}{}_{Rel} = id\text{-}Rel\ A$
⟨*proof*⟩

3.12.4 *Rel* as a digraph

Definition and elementary properties

definition *dg-Rel* :: $V \Rightarrow V$
where *dg-Rel* α =
[
Vset α,
set $\{T.$ *arr-Rel* α $T\}$,
$(\lambda T{\in_\circ}$*set* $\{T.$ *arr-Rel* α $T\}.$ $T(\!|$*ArrDom*$|\!))$,
$(\lambda T{\in_\circ}$*set* $\{T.$ *arr-Rel* α $T\}.$ $T(\!|$*ArrCod*$|\!))$
$]_\circ$

Components.

lemma *dg-Rel-components*:
shows *dg-Rel* $\alpha(\!|$*Obj*$|\!)$ = *Vset* α
and *dg-Rel* $\alpha(\!|$*Arr*$|\!)$ = *set* $\{T.$ *arr-Rel* α $T\}$
and *dg-Rel* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda T{\in_\circ}$*set* $\{T.$ *arr-Rel* α $T\}.$ $T(\!|$*ArrDom*$|\!))$
and *dg-Rel* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda T{\in_\circ}$*set* $\{T.$ *arr-Rel* α $T\}.$ $T(\!|$*ArrCod*$|\!))$
⟨*proof*⟩

Object

lemma *dg-Rel-Obj-iff*: $x \in_\circ$ *dg-Rel* $\alpha(\!|$*Obj*$|\!) \longleftrightarrow x \in_\circ$ *Vset* α
⟨*proof*⟩

Arrow

lemma *dg-Rel-Arr-iff*[*dg-Rel-cs-simps*]: $x \in_\circ$ *dg-Rel* $\alpha(\!|$*Arr*$|\!) \longleftrightarrow$ *arr-Rel* α x
⟨*proof*⟩

Domain

mk-VLambda *dg-Rel-components*(3)
|*vsv dg-Rel-Dom-vsv*[*dg-Rel-cs-intros*]|
|*vdomain dg-Rel-Dom-vdomain*[*dg-Rel-cs-simps*]|
|*app dg-Rel-Dom-app*[*unfolded set-Collect-arr-Rel*, *dg-Rel-cs-simps*]|

lemma *dg-Rel-Dom-vrange*: $\mathcal{R}_\circ$ (*dg-Rel* $\alpha(\!|$*Dom*$|\!)) \subseteq_\circ$ *dg-Rel* $\alpha(\!|$*Obj*$|\!)$
⟨*proof*⟩

Codomain

mk-VLambda *dg-Rel-components*(4)
|*vsv dg-Rel-Cod-vsv*[*dg-Rel-cs-intros*]|
|*vdomain dg-Rel-Cod-vdomain*[*dg-Rel-cs-simps*]|
|*app dg-Rel-Cod-app*[*unfolded set-Collect-arr-Rel*, *dg-Rel-cs-simps*]|

lemma *dg-Rel-Cod-vrange*: $\mathcal{R}_\circ$ (*dg-Rel* $\alpha(\!|$*Cod*$|\!)) \subseteq_\circ$ *dg-Rel* $\alpha(\!|$*Obj*$|\!)$
⟨*proof*⟩

Arrow with a domain and a codomain

Rules.

lemma *dg-Rel-is-arrI*[*dg-Rel-cs-intros*]:
assumes *arr-Rel* α S **and** $S(\!|$*ArrDom*$|\!) = A$ **and** $S(\!|$*ArrCod*$|\!) = B$
shows $S : A \mapsto_{dg\text{-}Rel\ \alpha} B$
⟨*proof*⟩

lemma *dg-Rel-is-arrD*:
assumes $S : A \mapsto_{dg\text{-}Rel\ \alpha} B$
shows *arr-Rel* α S
and [*dg-cs-simps*]: $S(\!|ArrDom|\!) = A$
and [*dg-cs-simps*]: $S(\!|ArrCod|\!) = B$
⟨*proof*⟩

lemma *dg-Rel-is-arrE*:
assumes $S : A \mapsto_{dg\text{-}Rel\ \alpha} B$
obtains *arr-Rel* α S **and** $S(\!|ArrDom|\!) = A$ **and** $S(\!|ArrCod|\!) = B$
⟨*proof*⟩

Elementary properties.

lemma (in $\mathcal{Z}$**)** *dg-Rel-incl-Rel-is-arr*:
assumes $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$ **and** $A \subseteq_\circ B$
shows *incl-Rel* $A\ B : A \mapsto_{dg\text{-}Rel\ \alpha} B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *dg-Rel-incl-Rel-is-arr'*[*dg-Rel-cs-intros*]:
assumes $A \in_\circ Vset\ \alpha$
and $B \in_\circ Vset\ \alpha$
and $A \subseteq_\circ B$
and $A' = A$
and $B' = B$
shows *incl-Rel* $A\ B : A' \mapsto_{dg\text{-}Rel\ \alpha} B'$
⟨*proof*⟩

lemmas [*dg-Rel-cs-intros*] = $\mathcal{Z}$.*dg-Rel-incl-Rel-is-arr'*

lemma *dg-Rel-is-arr-ArrValE*:
assumes $T : A \mapsto_{dg\text{-}Rel\ \alpha} B$ **and** $ab \in_\circ T(\!|ArrVal|\!)$
obtains $a\ b$
where $ab = \langle a, b \rangle$ **and** $a \in_\circ \mathcal{D}_\circ\ (T(\!|ArrVal|\!))$ **and** $b \in_\circ \mathcal{R}_\circ\ (T(\!|ArrVal|\!))$
⟨*proof*⟩

Rel is a digraph

lemma (in $\mathcal{Z}$**)** *dg-Rel-Hom-vifunion-in-Vset*:
assumes $X \in_\circ Vset\ \alpha$ **and** $Y \in_\circ Vset\ \alpha$
shows $(\bigcup_\circ A \in_\circ X.\ \bigcup_\circ B \in_\circ Y.\ Hom\ (dg\text{-}Rel\ \alpha)\ A\ B) \in_\circ Vset\ \alpha$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *digraph-dg-Rel*: *digraph* α (*dg-Rel* α)
⟨*proof*⟩

3.12.5 Canonical dagger for *Rel*

Dagger categories are exposed explicitly later. In the context of this section, the "dagger" is viewed merely as an explicitly defined homomorphism. A definition of a dagger functor, upon which the definition presented in this section is based, can be found in nLab [3][5]. This reference also contains the majority of the results that are presented in this subsection.

Definition and elementary properties

definition *dghm-dag-Rel* :: $V \Rightarrow V$ ($\langle \dagger_{DG.Rel} \rangle$)

[5] https://ncatlab.org/nlab/show/Rel)

where $\dagger_{DG.Rel}\ \alpha =$
[
vid-on (*dg-Rel* $\alpha(\!|Obj|\!)$),
VLambda (*dg-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*,
op-dg (*dg-Rel* α),
dg-Rel α
]$_\circ$

Components.

lemma *dghm-dag-Rel-components*:
shows $\dagger_{DG.Rel}\ \alpha(\!|ObjMap|\!) =$ *vid-on* (*dg-Rel* $\alpha(\!|Obj|\!)$)
and $\dagger_{DG.Rel}\ \alpha(\!|ArrMap|\!) =$ *VLambda* (*dg-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*
and $\dagger_{DG.Rel}\ \alpha(\!|HomDom|\!) =$ *op-dg* (*dg-Rel* α)
and $\dagger_{DG.Rel}\ \alpha(\!|HomCod|\!) =$ *dg-Rel* α
⟨*proof*⟩

Object map

mk-VLambda *dghm-dag-Rel-components*(1)[*folded VLambda-vid-on*]
|*vsv dghm-dag-Rel-ObjMap-vsv*[*dg-Rel-cs-intros*]|
|*vdomain*
dghm-dag-Rel-ObjMap-vdomain[*unfolded dg-Rel-components*, *dg-Rel-cs-simps*]
|
|*app dghm-dag-Rel-ObjMap-app*[*unfolded dg-Rel-components*, *dg-Rel-cs-simps*]|

lemma *dghm-dag-Rel-ObjMap-vrange*[*dg-cs-simps*]: $\mathcal{R}_\circ\ (\dagger_{DG.Rel}\ \alpha(\!|ObjMap|\!)) =$ *Vset* α
⟨*proof*⟩

Arrow map

mk-VLambda *dghm-dag-Rel-components*(2)
|*vsv dghm-dag-Rel-ArrMap-vsv*[*dg-Rel-cs-intros*]|
|*vdomain dghm-dag-Rel-ArrMap-vdomain*[*dg-Rel-cs-simps*]|
|*app dghm-dag-Rel-ArrMap-app*[*unfolded dg-Rel-cs-simps*, *dg-Rel-cs-simps*]|

lemma *dghm-dag-Rel-ArrMap-app-vdomain*[*dg-cs-simps*]:
assumes $T : A \mapsto_{dg\text{-}Rel\ \alpha} B$
shows $\mathcal{D}_\circ\ (\dagger_{DG.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!)(\!|ArrVal|\!)) = \mathcal{R}_\circ\ (T(\!|ArrVal|\!))$
⟨*proof*⟩

lemma *dghm-dag-Rel-ArrMap-app-vrange*[*dg-cs-simps*]:
assumes $T : A \mapsto_{dg\text{-}Rel\ \alpha} B$
shows $\mathcal{R}_\circ\ (\dagger_{DG.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!)(\!|ArrVal|\!)) = \mathcal{D}_\circ\ (T(\!|ArrVal|\!))$
⟨*proof*⟩

lemma *dghm-dag-Rel-ArrMap-app-iff*[*dg-cs-simps*]:
assumes $T : A \mapsto_{dg\text{-}Rel\ \alpha} B$
shows $\langle a, b\rangle \in_\circ \dagger_{DG.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!)(\!|ArrVal|\!) \longleftrightarrow \langle b, a\rangle \in_\circ T(\!|ArrVal|\!)$
⟨*proof*⟩

Further properties

lemma *dghm-dag-Rel-ArrMap-vrange*[*dg-Rel-cs-simps*]:
$\mathcal{R}_\circ\ (\dagger_{DG.Rel}\ \alpha(\!|ArrMap|\!)) =$ *dg-Rel* $\alpha(\!|Arr|\!)$
⟨*proof*⟩

lemma *dghm-dag-Rel-ArrMap-app-is-arr*:
assumes $T : b \mapsto_{dg\text{-}Rel\ \alpha} a$

shows
$\dagger_{DG.Rel}\ \alpha(\llbracket ArrMap \rrbracket)(\llbracket T \rrbracket) : \dagger_{DG.Rel}\ \alpha(\llbracket ObjMap \rrbracket)(\llbracket a \rrbracket) \mapsto_{dg\text{-}Rel\ \alpha}\ \dagger_{DG.Rel}\ \alpha(\llbracket ObjMap \rrbracket)(\llbracket b \rrbracket)$
⟨*proof*⟩

Canonical dagger for *Rel* is a digraph isomorphism

lemma (in $\mathcal{Z}$**)** *dghm-dag-Rel-is-iso-dghm*:
$\dagger_{DG.Rel}\ \alpha : op\text{-}dg\ (dg\text{-}Rel\ \alpha) \mapsto\mapsto_{DG.iso\alpha}\ dg\text{-}Rel\ \alpha$
⟨*proof*⟩

Further properties of the canonical dagger

lemma (in $\mathcal{Z}$**)** *dghm-cn-comp-dghm-dag-Rel-dghm-dag-Rel*:
$\dagger_{DG.Rel}\ \alpha\ _{DGHM}\circ\ \dagger_{DG.Rel}\ \alpha = dghm\text{-}id\ (dg\text{-}Rel\ \alpha)$
⟨*proof*⟩

3.13 *Par* as a digraph

3.13.1 Background

Par is usually defined as a category of sets and partial functions (see nLab [3][6]). However, there is little that can prevent one from exposing *Par* as a digraph and provide additional structure gradually in subsequent installments of this work. Thus, in this section, α-*Par* is defined as a digraph of sets and partial functions in V_α

named-theorems *dg-Par-cs-simps*
named-theorems *dg-Par-cs-intros*

lemmas [*dg-Par-cs-simps*] = *dg-Rel-shared-cs-simps*
lemmas [*dg-Par-cs-intros*] = *dg-Rel-shared-cs-intros*

3.13.2 Arrow for *Par*

Definition and elementary properties

locale *arr-Par* = $\mathcal{Z}$ α + *vfsequence* T + *ArrVal*: *vsv* ‹$T(\lfloor ArrVal \rfloor)$› **for** α T +
  assumes *arr-Par-length*[*dg-Rel-shared-cs-simps*, *dg-Par-cs-simps*]:
    vcard $T = 3_{\mathbb{N}}$
    and *arr-Par-ArrVal-vdomain*: $\mathcal{D}_\circ\ (T(\lfloor ArrVal \rfloor)) \subseteq_\circ T(\lfloor ArrDom \rfloor)$
    and *arr-Par-ArrVal-vrange*: $\mathcal{R}_\circ\ (T(\lfloor ArrVal \rfloor)) \subseteq_\circ T(\lfloor ArrCod \rfloor)$
    and *arr-Par-ArrDom-in-Vset*: $T(\lfloor ArrDom \rfloor) \in_\circ Vset\ \alpha$
    and *arr-Par-ArrCod-in-Vset*: $T(\lfloor ArrCod \rfloor) \in_\circ Vset\ \alpha$

Elementary properties.

sublocale *arr-Par* ⊆ *arr-Rel*
  ⟨*proof*⟩

lemmas (**in** *arr-Par*) [*dg-Par-cs-simps*] = *dg-Rel-shared-cs-simps*

Rules.

lemma (**in** *arr-Par*) *arr-Par-axioms'*[*dg-cs-intros*, *dg-Par-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *arr-Par* α' T
  ⟨*proof*⟩

mk-ide rf *arr-Par-def*[*unfolded arr-Par-axioms-def*]
  |*intro arr-ParI*|
  |*dest arr-ParD*[*dest*]|
  |*elim arr-ParE*[*elim*!]|

lemma (**in** $\mathcal{Z}$) *arr-Par-vfsequenceI*:
  assumes *vsv* r
    and $\mathcal{D}_\circ\ r \subseteq_\circ a$
    and $\mathcal{R}_\circ\ r \subseteq_\circ b$
    and $a \in_\circ Vset\ \alpha$
    and $b \in_\circ Vset\ \alpha$
  shows *arr-Par* α $[r, a, b]_\circ$
  ⟨*proof*⟩

lemma *arr-Par-arr-RelI*:
  assumes *arr-Rel* α T **and** *vsv* $(T(\lfloor ArrVal \rfloor))$
  shows *arr-Par* α T
⟨*proof*⟩

[6] https://ncatlab.org/nlab/show/partial+function

lemma *arr-Par-arr-RelD*:
assumes *arr-Par* α T
shows *arr-Rel* α T **and** *vsv* ($T(\!|ArrVal|\!)$)
⟨*proof*⟩

lemma *arr-Par-arr-RelE*:
assumes *arr-Par* α T
obtains *arr-Rel* α T **and** *vsv* ($T(\!|ArrVal|\!)$)
⟨*proof*⟩

Further properties.

lemma *arr-Par-eqI*:
assumes *arr-Par* α S
and *arr-Par* α T
and $S(\!|ArrVal|\!) = T(\!|ArrVal|\!)$
and $S(\!|ArrDom|\!) = T(\!|ArrDom|\!)$
and $S(\!|ArrCod|\!) = T(\!|ArrCod|\!)$
shows $S = T$
⟨*proof*⟩

lemma *small-arr-Par*[*simp*]: *small* $\{T.\ arr\text{-}Par\ \alpha\ T\}$
⟨*proof*⟩

lemma *set-Collect-arr-Par*[*simp*]:
$T \in_\circ set\ (Collect\ (arr\text{-}Par\ \alpha)) \longleftrightarrow arr\text{-}Par\ \alpha\ T$
⟨*proof*⟩

Composition

abbreviation (*input*) *comp-Par* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{Par}$› 55)
where *comp-Par* ≡ *comp-Rel*

lemma *arr-Par-comp-Par*[*dg-Par-cs-intros*]:
assumes *arr-Par* α S **and** *arr-Par* α T
shows *arr-Par* α ($S \circ_{Par} T$)
⟨*proof*⟩

lemma *arr-Par-comp-Par-ArrVal-app*:
assumes *arr-Par* α S
and *arr-Par* α T
and $x \in_\circ \mathcal{D}_\circ\ (T(\!|ArrVal|\!))$
and $T(\!|ArrVal|\!)(\!|x|\!) \in_\circ \mathcal{D}_\circ\ (S(\!|ArrVal|\!))$
shows $(S \circ_{Par} T)(\!|ArrVal|\!)(\!|x|\!) = S(\!|ArrVal|\!)(\!|T(\!|ArrVal|\!)(\!|x|\!)|\!)$
⟨*proof*⟩

Inclusion

abbreviation (*input*) *incl-Par* :: $V \Rightarrow V \Rightarrow V$
where *incl-Par* ≡ *incl-Rel*

lemma (**in** $\mathcal{Z}$) *arr-Par-incl-ParI*:
assumes $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$ **and** $A \subseteq_\circ B$
shows *arr-Par* α (*incl-Par* A B)
⟨*proof*⟩

Identity

abbreviation (*input*) *id-Par* :: $V \Rightarrow V$

where *id-Par* ≡ *id-Rel*

lemma (in $\mathcal{Z}$**)** *arr-Par-id-ParI*:
assumes $A \in_\circ Vset\ \alpha$
shows *arr-Par* α (*id-Par* A)
⟨*proof*⟩

lemma *arr-Par-comp-Par-id-Par-left*[*dg-Par-cs-simps*]:
assumes *arr-Par* $\alpha\ f$ **and** $f(\!|ArrCod|\!) = A$
shows *id-Par* $A \circ_{Par} f = f$
⟨*proof*⟩

lemma *arr-Par-comp-Par-id-Par-right*[*dg-Par-cs-simps*]:
assumes *arr-Par* $\alpha\ f$ **and** $f(\!|ArrDom|\!) = A$
shows $f \circ_{Par}$ *id-Par* $A = f$
⟨*proof*⟩

3.13.3 *Par* as a digraph

Definition and elementary properties

definition *dg-Par* :: $V \Rightarrow V$
where *dg-Par* α =
[
Vset α,
set $\{T.$ *arr-Par* $\alpha\ T\}$,
$(\lambda T \in_\circ set\ \{T.$ *arr-Par* $\alpha\ T\}.\ T(\!|ArrDom|\!))$,
$(\lambda T \in_\circ set\ \{T.$ *arr-Par* $\alpha\ T\}.\ T(\!|ArrCod|\!))$
]$_\circ$

Components.

lemma *dg-Par-components*:
shows *dg-Par* $\alpha(\!|Obj|\!) = Vset\ \alpha$
and *dg-Par* $\alpha(\!|Arr|\!) = set\ \{T.$ *arr-Par* $\alpha\ T\}$
and *dg-Par* $\alpha(\!|Dom|\!) = (\lambda T \in_\circ set\ \{T.$ *arr-Par* $\alpha\ T\}.\ T(\!|ArrDom|\!))$
and *dg-Par* $\alpha(\!|Cod|\!) = (\lambda T \in_\circ set\ \{T.$ *arr-Par* $\alpha\ T\}.\ T(\!|ArrCod|\!))$
⟨*proof*⟩

Object

lemma *dg-Par-Obj-iff*: $x \in_\circ$ *dg-Par* $\alpha(\!|Obj|\!) \longleftrightarrow x \in_\circ Vset\ \alpha$
⟨*proof*⟩

Arrow

lemma *dg-Par-Arr-iff*[*dg-Par-cs-simps*]: $x \in_\circ$ *dg-Par* $\alpha(\!|Arr|\!) \longleftrightarrow$ *arr-Par* $\alpha\ x$
⟨*proof*⟩

Domain

mk-VLambda *dg-Par-components*(3)
|*vsv dg-Par-Dom-vsv*[*dg-Par-cs-intros*]|
|*vdomain dg-Par-Dom-vdomain*[*dg-Par-cs-simps*]|
|*app dg-Par-Dom-app*[*unfolded set-Collect-arr-Par*, *dg-Par-cs-simps*]|

lemma *dg-Par-Dom-vrange*: $\mathcal{R}_\circ$ (*dg-Par* $\alpha(\!|Dom|\!)) \subseteq_\circ$ *dg-Par* $\alpha(\!|Obj|\!)$
⟨*proof*⟩

Codomain

mk-VLambda *dg-Par-components*(4)
|*vsv dg-Par-Cod-vsv*[*dg-Par-cs-intros*]|
|*vdomain dg-Par-Cod-vdomain*[*dg-Par-cs-simps*]|
|*app dg-Par-Cod-app*[*unfolded set-Collect-arr-Par*, *dg-Par-cs-simps*]|

lemma *dg-Par-Cod-vrange*: $\mathcal{R}_{\circ}\ (\textit{dg-Par}\ \alpha(\!|\textit{Cod}|\!))\subseteq_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$
⟨*proof*⟩

Arrow with a domain and a codomain

Rules.

lemma *dg-Par-is-arrI*:
assumes *arr-Par* $\alpha\ S$ **and** $S(\!|\textit{ArrDom}|\!)=A$ **and** $S(\!|\textit{ArrCod}|\!)=B$
shows $S:A\mapsto_{\textit{dg-Par}\ \alpha}B$
⟨*proof*⟩

lemmas [*dg-Par-cs-intros*] = *dg-Par-is-arrI*

lemma *dg-Par-is-arrD*:
assumes $S:A\mapsto_{\textit{dg-Par}\ \alpha}B$
shows *arr-Par* $\alpha\ S$
and [*dg-cs-simps*]: $S(\!|\textit{ArrDom}|\!)=A$
and [*dg-cs-simps*]: $S(\!|\textit{ArrCod}|\!)=B$
⟨*proof*⟩

lemma *dg-Par-is-arrE*:
assumes $S:A\mapsto_{\textit{dg-Par}\ \alpha}B$
obtains *arr-Par* $\alpha\ S$ **and** $S(\!|\textit{ArrDom}|\!)=A$ **and** $S(\!|\textit{ArrCod}|\!)=B$
⟨*proof*⟩

Elementary properties.

lemma *dg-Par-is-arr-dg-Rel-is-arr*:
assumes $r:a\mapsto_{\textit{dg-Par}\ \alpha}b$
shows $r:a\mapsto_{\textit{dg-Rel}\ \alpha}b$
⟨*proof*⟩

lemma *dg-Par-Hom-vsubset-dg-Rel-Hom*:
assumes $a\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)\ b\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$
shows $\textit{Hom}\ (\textit{dg-Par}\ \alpha)\ a\ b\subseteq_{\circ}\textit{Hom}\ (\textit{dg-Rel}\ \alpha)\ a\ b$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *dg-Par-incl-Par-is-arr*:
assumes $A\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$ **and** $B\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$ **and** $A\subseteq_{\circ}B$
shows $\textit{incl-Par}\ A\ B:A\mapsto_{\textit{dg-Par}\ \alpha}B$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *dg-Par-incl-Par-is-arr'*[*dg-Par-cs-intros*]:
assumes $A\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $B\in_{\circ}\textit{dg-Par}\ \alpha(\!|\textit{Obj}|\!)$
and $A\subseteq_{\circ}B$
and $A'=A$
and $B'=B$
shows $\textit{incl-Par}\ A\ B:A'\mapsto_{\textit{dg-Par}\ \alpha}B'$
⟨*proof*⟩

lemmas [*dg-Par-cs-intros*] = $\mathcal{Z}$.*dg-Par-incl-Par-is-arr'*

Par is a digraph

lemma (in $\mathcal{Z}$**)** *dg-Par-Hom-vifunion-in-Vset*:
assumes $X \in_\circ$ *Vset* α **and** $Y \in_\circ$ *Vset* α
shows $(\bigcup_\circ A \in_\circ X.\ \bigcup_\circ B \in_\circ Y.$ *Hom* (*dg-Par* α) $A\ B) \in_\circ$ *Vset* α
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *digraph-dg-Par*: *digraph* α (*dg-Par* α)
⟨*proof*⟩

Par is a wide subdigraph of *Rel*

lemma (in $\mathcal{Z}$**)** *wide-subdigraph-dg-Par-dg-Rel*: *dg-Par* $\alpha \subseteq_{DG.wide\alpha}$ *dg-Rel* α
⟨*proof*⟩

3.14 *Set* as a digraph

3.14.1 Background

Set is usually defined as a category of sets and total functions (see Chapter I-2 in [39]). However, there is little that can prevent one from exposing *Set* as a digraph and provide additional structure gradually later. Thus, in this section, α-*Set* is defined as a digraph of sets and binary relations in the set V_α.

named-theorems *dg-Set-cs-simps*
named-theorems *dg-Set-cs-intros*

lemmas [*dg-Set-cs-simps*] = *dg-Rel-shared-cs-simps*
lemmas [*dg-Set-cs-intros*] = *dg-Rel-shared-cs-intros*

3.14.2 Arrow for *Set*

Definition and elementary properties

locale *arr-Set* = $\mathcal{Z}$ α + *vfsequence* T + *ArrVal*: *vsv* ‹$T(\!|ArrVal|\!)$› **for** α T +
  assumes *arr-Set-length*[*dg-Rel-shared-cs-simps*, *dg-Set-cs-simps*]:
    vcard $T = 3_{\mathbb{N}}$
   and *arr-Set-ArrVal-vdomain*[*dg-Rel-shared-cs-simps*, *dg-Set-cs-simps*]:
    $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = T(\!|ArrDom|\!)$
   and *arr-Set-ArrVal-vrange*: $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) \subseteq_\circ T(\!|ArrCod|\!)$
   and *arr-Set-ArrDom-in-Vset*: $T(\!|ArrDom|\!) \in_\circ Vset\ \alpha$
   and *arr-Set-ArrCod-in-Vset*: $T(\!|ArrCod|\!) \in_\circ Vset\ \alpha$

lemmas [*dg-Set-cs-simps*] = *arr-Set.arr-Set-ArrVal-vdomain*

Elementary properties.

sublocale *arr-Set* ⊆ *arr-Par*
  ⟨*proof*⟩

Rules.

lemma (**in** *arr-Set*) *arr-Set-axioms'*[*dg-cs-intros*, *dg-Set-cs-intros*]:
  assumes $\alpha' = \alpha$
  shows *arr-Set* α' T
  ⟨*proof*⟩

mk-ide rf *arr-Set-def*[*unfolded arr-Set-axioms-def*]
  |*intro arr-SetI*|
  |*dest arr-SetD*[*dest*]|
  |*elim arr-SetE*[*elim*!]|

lemma (**in** $\mathcal{Z}$) *arr-Set-vfsequenceI*:
  assumes *vsv* r
    and $\mathcal{D}_\circ\ r = a$
    and $\mathcal{R}_\circ\ r \subseteq_\circ b$
    and $a \in_\circ Vset\ \alpha$
    and $b \in_\circ Vset\ \alpha$
  shows *arr-Set* α $[r, a, b]_\circ$
  ⟨*proof*⟩

lemma *arr-Set-arr-ParI*:
  assumes *arr-Par* α T **and** $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = T(\!|ArrDom|\!)$
  shows *arr-Set* α T
⟨*proof*⟩

lemma *arr-Set-arr-ParD*:
assumes *arr-Set* α T
shows *arr-Par* α T **and** $\mathcal{D}_\circ$ $(T(\!|ArrVal|\!)) = T(\!|ArrDom|\!)$
⟨*proof*⟩

lemma *arr-Set-arr-ParE*:
assumes *arr-Set* α T
obtains *arr-Par* α T **and** $\mathcal{D}_\circ$ $(T(\!|ArrVal|\!)) = T(\!|ArrDom|\!)$
⟨*proof*⟩

Further properties.

lemma *arr-Set-eqI*:
assumes *arr-Set* α S
and *arr-Set* α T
and $S(\!|ArrVal|\!) = T(\!|ArrVal|\!)$
and $S(\!|ArrDom|\!) = T(\!|ArrDom|\!)$
and $S(\!|ArrCod|\!) = T(\!|ArrCod|\!)$
shows $S = T$
⟨*proof*⟩

lemma *small-arr-Set*[*simp*]: *small* $\{T.\ arr\text{-}Set\ \alpha\ T\}$
⟨*proof*⟩

lemma *set-Collect-arr-Set*[*simp*]:
$T \in_\circ set\ (Collect\ (arr\text{-}Set\ \alpha)) \longleftrightarrow arr\text{-}Set\ \alpha\ T$
⟨*proof*⟩

Composition

See [39]).

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

lemma *arr-Set-comp-Set*[*dg-Set-cs-intros*]:
assumes *arr-Set* α S **and** *arr-Set* α T **and** $\mathcal{R}_\circ$ $(T(\!|ArrVal|\!)) \subseteq_\circ \mathcal{D}_\circ\ (S(\!|ArrVal|\!))$
shows *arr-Set* α $(S \circ_{Set} T)$
⟨*proof*⟩

lemma *arr-Set-comp-Set-ArrVal-app*:
assumes *arr-Set* α S
and *arr-Set* α T
and $x \in_\circ \mathcal{D}_\circ\ (T(\!|ArrVal|\!))$
and $T(\!|ArrVal|\!)(\!|x|\!) \in_\circ \mathcal{D}_\circ\ (S(\!|ArrVal|\!))$
shows $(S \circ_{Set} T)(\!|ArrVal|\!)(\!|x|\!) = S(\!|ArrVal|\!)(\!|T(\!|ArrVal|\!)(\!|x|\!)|\!)$
⟨*proof*⟩

Inclusion

abbreviation (*input*) *incl-Set* :: $V \Rightarrow V \Rightarrow V$
where *incl-Set* ≡ *incl-Rel*

lemma (**in** $\mathcal{Z}$) *arr-Set-incl-SetI*:
assumes $A \in_\circ Vset\ \alpha$ **and** $B \in_\circ Vset\ \alpha$ **and** $A \subseteq_\circ B$
shows *arr-Set* α $(incl\text{-}Set\ A\ B)$
⟨*proof*⟩

Identity

abbreviation (*input*) *id-Set* :: $V \Rightarrow V$
  where *id-Set* ≡ *id-Rel*

lemma (in $\mathcal{Z}$**)** *arr-Set-id-SetI*:
  assumes $A \in_\circ Vset\ \alpha$
  shows *arr-Set* α (*id-Set* A)
⟨*proof*⟩

lemma *arr-Set-comp-Set-id-Set-left*[*dg-Set-cs-simps*]:
  assumes *arr-Set* α F **and** $F(\!|ArrCod|\!) = A$
  shows *id-Set* $A \circ_{Set} F = F$
⟨*proof*⟩

lemma *arr-Set-comp-Set-id-Set-right*[*dg-Set-cs-simps*]:
  assumes *arr-Set* α F **and** $F(\!|ArrDom|\!) = A$
  shows $F \circ_{Set}$ *id-Set* $A = F$
⟨*proof*⟩

3.14.3 *Set* as a digraph

Definition and elementary properties

definition *dg-Set* :: $V \Rightarrow V$
  where *dg-Set* α =
    [
      Vset α,
      set $\{T.\ arr\text{-}Set\ \alpha\ T\}$,
      $(\lambda T \in_\circ set\ \{T.\ arr\text{-}Set\ \alpha\ T\}.\ T(\!|ArrDom|\!))$,
      $(\lambda T \in_\circ set\ \{T.\ arr\text{-}Set\ \alpha\ T\}.\ T(\!|ArrCod|\!))$
    ]$_\circ$

Components.

lemma *dg-Set-components*:
  shows *dg-Set* $\alpha(\!|Obj|\!)$ = *Vset* α
    and *dg-Set* $\alpha(\!|Arr|\!)$ = *set* $\{T.\ arr\text{-}Set\ \alpha\ T\}$
    and *dg-Set* $\alpha(\!|Dom|\!) = (\lambda T \in_\circ set\ \{T.\ arr\text{-}Set\ \alpha\ T\}.\ T(\!|ArrDom|\!))$
    and *dg-Set* $\alpha(\!|Cod|\!) = (\lambda T \in_\circ set\ \{T.\ arr\text{-}Set\ \alpha\ T\}.\ T(\!|ArrCod|\!))$
  ⟨*proof*⟩

Object

lemma *dg-Set-Obj-iff*: $x \in_\circ$ *dg-Set* $\alpha(\!|Obj|\!) \longleftrightarrow x \in_\circ Vset\ \alpha$
  ⟨*proof*⟩

Arrow

lemma *dg-Set-Arr-iff*[*dg-Set-cs-simps*]: $x \in_\circ$ *dg-Set* $\alpha(\!|Arr|\!) \longleftrightarrow$ *arr-Set* α x
  ⟨*proof*⟩

Domain

mk-VLambda *dg-Set-components*(3)
  |*vsv dg-Set-Dom-vsv*[*dg-Set-cs-intros*]|
  |*vdomain dg-Set-Dom-vdomain*[*dg-Set-cs-simps*]|
  |*app dg-Set-Dom-app*[*unfolded set-Collect-arr-Set*, *dg-Set-cs-simps*]|

lemma *dg-Set-Dom-vrange*: $\mathcal{R}_\circ$ (*dg-Set* $\alpha(\!|Dom|\!)$) $\subseteq_\circ$ *dg-Set* $\alpha(\!|Obj|\!)$
  ⟨*proof*⟩

Codomain

mk-VLambda *dg-Set-components*(4)
|*vsv dg-Set-Cod-vsv*[*dg-Set-cs-intros*]|
|*vdomain dg-Set-Cod-vdomain*[*dg-Set-cs-simps*]|
|*app dg-Set-Cod-app*[*unfolded set-Collect-arr-Set*, *dg-Set-cs-simps*]|

lemma *dg-Set-Cod-vrange*: $\mathcal{R}_\circ\ (dg\text{-}Set\ \alpha(\!|Cod|\!))\subseteq_\circ dg\text{-}Set\ \alpha(\!|Obj|\!)$
⟨*proof*⟩

Arrow with a domain and a codomain

Rules.

lemma *dg-Set-is-arrI*[*dg-Set-cs-intros*]:
assumes *arr-Set* $\alpha\ S$ **and** $S(\!|ArrDom|\!) = A$ **and** $S(\!|ArrCod|\!) = B$
shows $S : A \mapsto_{dg\text{-}Set\ \alpha} B$
⟨*proof*⟩

lemma *dg-Set-is-arrD*:
assumes $S : A \mapsto_{dg\text{-}Set\ \alpha} B$
shows *arr-Set* $\alpha\ S$
and [*dg-cs-simps*]: $S(\!|ArrDom|\!) = A$
and [*dg-cs-simps*]: $S(\!|ArrCod|\!) = B$
⟨*proof*⟩

lemma *dg-Set-is-arrE*:
assumes $S : A \mapsto_{dg\text{-}Set\ \alpha} B$
obtains *arr-Set* $\alpha\ S$ **and** $S(\!|ArrDom|\!) = A$ **and** $S(\!|ArrCod|\!) = B$
⟨*proof*⟩

lemma *dg-Set-ArrVal-vdomain*[*dg-Set-cs-simps*, *dg-cs-simps*]:
assumes $T : A \mapsto_{dg\text{-}Set\ \alpha} B$
shows $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
⟨*proof*⟩

Elementary properties.

lemma *dg-Set-ArrVal-app-vrange*[*dg-Set-cs-intros*]:
assumes $F : A \mapsto_{dg\text{-}Set\ \alpha} B$ **and** $a \in_\circ A$
shows $F(\!|ArrVal|\!)(\!|a|\!) \in_\circ B$
⟨*proof*⟩

lemma *dg-Set-is-arr-dg-Par-is-arr*:
assumes $T : A \mapsto_{dg\text{-}Set\ \alpha} B$
shows $T : A \mapsto_{dg\text{-}Par\ \alpha} B$
⟨*proof*⟩

lemma *dg-Set-Hom-vsubset-dg-Par-Hom*:
assumes $a \in_\circ dg\text{-}Set\ \alpha(\!|Obj|\!)$ $b \in_\circ dg\text{-}Set\ \alpha(\!|Obj|\!)$
shows $Hom\ (dg\text{-}Set\ \alpha)\ a\ b \subseteq_\circ Hom\ (dg\text{-}Par\ \alpha)\ a\ b$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *dg-Set-incl-Set-is-arr*:
assumes $A \in_\circ dg\text{-}Set\ \alpha(\!|Obj|\!)$ **and** $B \in_\circ dg\text{-}Set\ \alpha(\!|Obj|\!)$ **and** $A \subseteq_\circ B$
shows $incl\text{-}Set\ A\ B : A \mapsto_{dg\text{-}Set\ \alpha} B$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *dg-Set-incl-Set-is-arr'*[*dg-cs-intros*, *dg-Set-cs-intros*]:

assumes $A \in_{\circ} \textit{dg-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $B \in_{\circ} \textit{dg-Set}\ \alpha(\!|\textit{Obj}|\!)$
and $A \subseteq_{\circ} B$
and $A' = A$
and $B' = B$
and $\mathfrak{C}' = \textit{dg-Set}\ \alpha$
shows $\textit{incl-Set}\ A\ B : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*dg-Set-cs-intros*] = $\mathcal{Z}$.*dg-Set-incl-Set-is-arr'*

Set is a digraph

lemma (in $\mathcal{Z}$**)** *dg-Set-Hom-vifunion-in-Vset*:
assumes $X \in_{\circ} \textit{Vset}\ \alpha$ **and** $Y \in_{\circ} \textit{Vset}\ \alpha$
shows $(\bigcup_{\circ} A \in_{\circ} X.\ \bigcup_{\circ} B \in_{\circ} Y.\ \textit{Hom}\ (\textit{dg-Set}\ \alpha)\ A\ B) \in_{\circ} \textit{Vset}\ \alpha$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *digraph-dg-Set*: $\textit{digraph}\ \alpha\ (\textit{dg-Set}\ \alpha)$
⟨*proof*⟩

Set is a wide subdigraph of *Par*

lemma (in $\mathcal{Z}$**)** *wide-subdigraph-dg-Set-dg-Par*: $\textit{dg-Set}\ \alpha \subseteq_{DG.wide\alpha}\ \textit{dg-Par}\ \alpha$
⟨*proof*⟩

Chapter 4

Semicategories

4.1 Introduction

4.1.1 Background

Many concepts that are normally associated with category theory can be generalized to semicategories. It is the goal of this chapter to expose these generalized concepts and provide a foundation for the development of the notion of a category in [41].

4.1.2 Preliminaries

named-theorems *smc-op-simps*
named-theorems *smc-op-intros*

named-theorems *smc-cs-simps*
named-theorems *smc-cs-intros*

named-theorems *smc-arrow-cs-intros*

4.1.3 CS setup for foundations

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

4.2 Semicategory

4.2.1 Background

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

Slicing

Slicing is a term that is introduced in this work for the description of the process of the conversion of more specialized mathematical objects to their generalizations.

The terminology was adapted from the informal imperative object oriented programming, where the term slicing often refers to the process of copying an object of a subclass type to an object of a superclass type [5][1]. However, it is important to note that the term has other meanings in programming and computer science.

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

Components.

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

Regular definitions.

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

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

Composition and composable arrows

The definition of a set of *composable-arrs* is equivalent to the definition of *composable pairs* presented on page 10 in [39] (see theorem *dg-composable-arrs'* below). Nonetheless, the definition is meant to be used sparingly. Normally, the arrows are meant to be specified explicitly using the predicate *is-arr*.

definition *Comp* :: V
  where [*dg-field-simps*]: *Comp* = $4_{\mathbb{N}}$

abbreviation *Comp-app* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{A1}$› 55)
  where *Comp-app* $\mathfrak{C}$ a b ≡ $\mathfrak{C}(\!|Comp|\!)(\!|a, b|\!)_\bullet$

definition *composable-arrs* :: $V \Rightarrow V$
  where *composable-arrs* $\mathfrak{C}$ = *set*
    $\{[g, f]_\circ \mid g\ f.\ \exists a\ b\ c.\ g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b\}$

lemma *small-composable-arrs*[*simp*]:
  small $\{[g, f]_\circ \mid g\ f.\ \exists a\ b\ c.\ g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b\}$
⟨*proof*⟩

Rules.

lemma *composable-arrsI*[*smc-cs-intros*]:

[1] https://en.wikipedia.org/wiki/Object_slicing

assumes $gf = [g, f]_\circ$ **and** $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$
shows $gf \in_\circ$ *composable-arrs* $\mathfrak{C}$
⟨*proof*⟩

lemma *composable-arrsE*[*elim*!]:
assumes $gf \in_\circ$ *composable-arrs* $\mathfrak{C}$
obtains $g\ f\ a\ b\ c$ **where** $gf = [g, f]_\circ$ **and** $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma *small-composable-arrs'*[*simp*]:
small $\{[g, f]_\circ \mid g\ f.\ g \in_\circ \mathfrak{C}(\!|Arr|\!) \wedge f \in_\circ \mathfrak{C}(\!|Arr|\!) \wedge \mathfrak{C}(\!|Dom|\!)(\!|g|\!) = \mathfrak{C}(\!|Cod|\!)(\!|f|\!)\}$
⟨*proof*⟩

lemma *dg-composable-arrs'*:
set $\{[g, f]_\circ \mid g\ f.\ g \in_\circ \mathfrak{C}(\!|Arr|\!) \wedge f \in_\circ \mathfrak{C}(\!|Arr|\!) \wedge \mathfrak{C}(\!|Dom|\!)(\!|g|\!) = \mathfrak{C}(\!|Cod|\!)(\!|f|\!)\} =$
composable-arrs $\mathfrak{C}$
⟨*proof*⟩

4.2.2 Definition and elementary properties

The definition of a semicategory that is used in this work is similar to the definition that was used in [42]. It is also a natural generalization of the definition of a category that is presented in Chapter I-2 in [39]. The generalization is performed by omitting the identity and the axioms associated with it. The amendments to the definitions that are associated with size have already been explained in the previous chapter.

locale *semicategory* = $\mathcal{Z}\ \alpha$ + *vfsequence* $\mathfrak{C}$ + *Comp*: *vsv* ‹$\mathfrak{C}(\!|Comp|\!)$› **for** $\alpha\ \mathfrak{C}$ +
assumes *smc-length*[*smc-cs-simps*]: *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
and *smc-digraph*[*slicing-intros*]: *digraph* α (*smc-dg* $\mathfrak{C}$)
and *smc-Comp-vdomain*: $gf \in_\circ \mathcal{D}_\circ (\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
$(\exists g\ f\ b\ c\ a.\ gf = [g, f]_\circ \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
and *smc-Comp-is-arr*:
$[\![\ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
and *smc-Comp-assoc*[*smc-cs-simps*]:
$[\![\ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

lemmas [*smc-cs-simps*] =
semicategory.smc-length
semicategory.smc-Comp-assoc

lemma (**in** *semicategory*) *smc-Comp-is-arr'*[*smc-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*⟩

lemmas [*smc-cs-intros*] =
semicategory.smc-Comp-is-arr'
semicategory.smc-Comp-is-arr

lemmas [*slicing-intros*] = *semicategory.smc-digraph*

Rules.

lemma (**in** *semicategory*) *semicategory-axioms'*[*smc-cs-intros*]:
assumes $\alpha' = \alpha$
shows *semicategory* $\alpha'\ \mathfrak{C}$

⟨*proof*⟩

mk-ide rf *semicategory-def*[*unfolded semicategory-axioms-def*]
|*intro semicategoryI*|
|*dest semicategoryD*[*dest*]|
|*elim semicategoryE*[*elim*]|

lemma *semicategoryI′*:
assumes $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{C}$
and *vsv* $(\mathfrak{C}(\!|Comp|\!))$
and *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
and *vsv* $(\mathfrak{C}(\!|Dom|\!))$
and *vsv* $(\mathfrak{C}(\!|Cod|\!))$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
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 $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and $\bigwedge A\ B.\ [\![\ A \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ B \subseteq_\circ \mathfrak{C}(\!|Obj|\!);\ A \in_\circ Vset\ \alpha;\ B \in_\circ Vset\ \alpha\ ]\!] \Longrightarrow$
$(\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ \mathfrak{C}\ a\ b) \in_\circ Vset\ \alpha$
shows *semicategory* $\alpha\ \mathfrak{C}$
⟨*proof*⟩

lemma *semicategoryD′*:
assumes *semicategory* $\alpha\ \mathfrak{C}$
shows $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{C}$
and *vsv* $(\mathfrak{C}(\!|Comp|\!))$
and *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
and *vsv* $(\mathfrak{C}(\!|Dom|\!))$
and *vsv* $(\mathfrak{C}(\!|Cod|\!))$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
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 $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
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 *semicategoryE′*:
assumes *semicategory* $\alpha\ \mathfrak{C}$
obtains $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{C}$
and *vsv* $(\mathfrak{C}(\!|Comp|\!))$
and *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
and *vsv* $(\mathfrak{C}(\!|Dom|\!))$

and *vsv* $(\mathfrak{C}(\!|Cod|\!))$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
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 $\mathfrak{C}(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
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*⟩

While using the sublocale infrastructure in conjunction with the rewrite morphisms is plausible for achieving automation of slicing, this approach has certain limitations. For example, the rewrite morphisms cannot be added to a given interpretation that was achieved using the command **sublocale**[2]. Thus, instead of using a partial solution based on the command **sublocale**, the rewriting is performed manually for selected theorems. However, it is hoped that better automation will be provided in the future.

context *semicategory*
begin

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

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

lemmas-with [*unfolded slicing-simps*]:
smc-Dom-vdomain[*smc-cs-simps*] = *dg.dg-Dom-vdomain*
and *smc-Dom-vrange* = *dg.dg-Dom-vrange*
and *smc-Cod-vdomain*[*smc-cs-simps*] = *dg.dg-Cod-vdomain*
and *smc-Cod-vrange* = *dg.dg-Cod-vrange*
and *smc-Obj-vsubset-Vset* = *dg.dg-Obj-vsubset-Vset*
and *smc-Hom-vifunion-in-Vset*[*smc-cs-intros*] = *dg.dg-Hom-vifunion-in-Vset*
and *smc-Obj-if-Dom-vrange* = *dg.dg-Obj-if-Dom-vrange*
and *smc-Obj-if-Cod-vrange* = *dg.dg-Obj-if-Cod-vrange*
and *smc-is-arrD* = *dg.dg-is-arrD*
and *smc-is-arrE*[*elim*] = *dg.dg-is-arrE*
and *smc-in-ArrE*[*elim*] = *dg.dg-in-ArrE*
and *smc-Hom-in-Vset*[*smc-cs-intros*] = *dg.dg-Hom-in-Vset*
and *smc-Arr-vsubset-Vset* = *dg.dg-Arr-vsubset-Vset*
and *smc-Dom-vsubset-Vset* = *dg.dg-Dom-vsubset-Vset*
and *smc-Cod-vsubset-Vset* = *dg.dg-Cod-vsubset-Vset*
and *smc-Obj-in-Vset* = *dg.dg-Obj-in-Vset*
and *smc-in-Obj-in-Vset*[*smc-cs-intros*] = *dg.dg-in-Obj-in-Vset*
and *smc-Arr-in-Vset* = *dg.dg-Arr-in-Vset*
and *smc-in-Arr-in-Vset*[*smc-cs-intros*] = *dg.dg-in-Arr-in-Vset*
and *smc-Dom-in-Vset* = *dg.dg-Dom-in-Vset*
and *smc-Cod-in-Vset* = *dg.dg-Cod-in-Vset*
and *smc-digraph-if-ge-Limit* = *dg.dg-digraph-if-ge-Limit*
and *smc-Dom-app-in-Obj* = *dg.dg-Dom-app-in-Obj*
and *smc-Cod-app-in-Obj* = *dg.dg-Cod-app-in-Obj*
and *smc-Arr-vempty-if-Obj-vempty* = *dg.dg-Arr-vempty-if-Obj-vempty*
and *smc-Dom-vempty-if-Arr-vempty* = *dg.dg-Dom-vempty-if-Arr-vempty*

[2] https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2019-September/msg00074.html

and *smc-Cod-vempty-if-Arr-vempty* = *dg.dg-Cod-vempty-if-Arr-vempty*

end

lemmas [*smc-cs-intros*] =
semicategory.smc-is-arrD(1–3)
semicategory.smc-Hom-in-Vset

Elementary properties.

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

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

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

lemma (**in** *semicategory*) *smc-Comp-vdomainI*[*smc-cs-intros*]:
assumes $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$ **and** $gf = [g, f]_{\circ}$
shows $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\!|Comp|\!))$
⟨*proof*⟩

lemmas [*smc-cs-intros*] = *semicategory.smc-Comp-vdomainI*

lemma (**in** *semicategory*) *smc-Comp-vdomainE*[*elim*!]:
assumes $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\!|Comp|\!))$
obtains $g\ f\ a\ b\ c$ **where** $gf = [g, f]_{\circ}$ **and** $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-Comp-vdomain-is-composable-arrs*:
$\mathcal{D}_{\circ}(\mathfrak{C}(\!|Comp|\!)) = composable\text{-}arrs\ \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-Comp-vrange*: $\mathcal{R}_{\circ}(\mathfrak{C}(\!|Comp|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

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

Size.

lemma (**in** *semicategory*) *smc-Comp-vsubset-Vset*: $\mathfrak{C}(\!|Comp|\!) \subseteq_{\circ} Vset\ \alpha$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-semicategory-in-Vset-4*: $\mathfrak{C} \in_{\circ} Vset\ (\alpha + 4_{\mathbb{N}})$

⟨*proof*⟩

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

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

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

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

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

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

Further properties.

lemma (**in** *semicategory*) *smc-Comp-vempty-if-Arr-vempty*:
assumes $\mathfrak{C}(\!|Arr|\!) = 0$
shows $\mathfrak{C}(\!|Comp|\!) = 0$
⟨*proof*⟩

4.2.3 Opposite semicategory

Definition and elementary properties

See Chapter II-2 in [39].

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

Components.

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

lemma *op-smc-component-intros*[*smc-op-intros*]:
shows $a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow a \in_{\circ} op\text{-}smc\ \mathfrak{C}(\!|Obj|\!)$
and $f \in_{\circ} \mathfrak{C}(\!|Arr|\!) \Longrightarrow f \in_{\circ} op\text{-}smc\ \mathfrak{C}(\!|Arr|\!)$
⟨*proof*⟩

Slicing.

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

Regular definitions.

lemma *op-smc-Comp-vdomain*[*smc-op-simps*]:
$\mathcal{D}_{\circ}\ (op\text{-}smc\ \mathfrak{C}(\!|Comp|\!)) = (\mathcal{D}_{\circ}\ (\mathfrak{C}(\!|Comp|\!)))^{-1}_{\bullet}$
⟨*proof*⟩

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

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

lemma (**in** *semicategory*) *op-smc-Comp-vrange*[*smc-op-simps*]:
$\mathcal{R}_{\circ}\ (op\text{-}smc\ \mathfrak{C}(\!|Comp|\!)) = \mathcal{R}_{\circ}\ (\mathfrak{C}(\!|Comp|\!))$
⟨*proof*⟩

lemmas [*smc-op-simps*] = *semicategory.op-smc-Comp-vrange*

lemma (**in** *semicategory*) *op-smc-Comp*[*smc-op-simps*]:
assumes $f : b \mapsto_{\mathfrak{C}} c$ **and** $g : a \mapsto_{\mathfrak{C}} b$
shows $g \circ_{A\, op\text{-}smc\ \mathfrak{C}} f = f \circ_{A\, \mathfrak{C}} g$
⟨*proof*⟩

lemmas [*smc-op-simps*] = *semicategory.op-smc-Comp*

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

Further properties

lemma (**in** *semicategory*) *semicategory-op*[*smc-op-intros*]:
$semicategory\ \alpha\ (op\text{-}smc\ \mathfrak{C})$
⟨*proof*⟩

lemmas *semicategory-op*[*smc-op-intros*] = *semicategory.semicategory-op*

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

lemmas *smc-op-smc-op-smc*[*smc-op-simps*] = *semicategory.smc-op-smc-op-smc*

lemma *eq-op-smc-iff*[*smc-op-simps*]:
assumes $semicategory\ \alpha\ \mathfrak{A}$ **and** $semicategory\ \alpha\ \mathfrak{B}$
shows $op\text{-}smc\ \mathfrak{A} = op\text{-}smc\ \mathfrak{B} \longleftrightarrow \mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

4.2.4 Arrow with a domain and a codomain

lemma (**in** *semicategory*) *smc-assoc-helper*:
assumes $f : a \mapsto_{\mathfrak{C}} b$
and $g : b \mapsto_{\mathfrak{C}} c$

and $h : c \mapsto_{\mathfrak{C}} d$
and $h \circ_{A\mathfrak{C}} g = q$
shows $h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f) = q \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-pattern-rectangle-right*:
assumes $aa' : a \mapsto_{\mathfrak{C}} a'$
and $a'a'' : a' \mapsto_{\mathfrak{C}} a''$
and $a''b'' : a'' \mapsto_{\mathfrak{C}} b''$
and $ab : a \mapsto_{\mathfrak{C}} b$
and $bb' : b \mapsto_{\mathfrak{C}} b'$
and $b'b'' : b' \mapsto_{\mathfrak{C}} b''$
and $a'b' : a' \mapsto_{\mathfrak{C}} b'$
and $a'b' \circ_{A\mathfrak{C}} aa' = bb' \circ_{A\mathfrak{C}} ab$
and $b'b'' \circ_{A\mathfrak{C}} a'b' = a''b'' \circ_{A\mathfrak{C}} a'a''$
shows $a''b'' \circ_{A\mathfrak{C}} (a'a'' \circ_{A\mathfrak{C}} aa') = (b'b'' \circ_{A\mathfrak{C}} bb') \circ_{A\mathfrak{C}} ab$
⟨*proof*⟩

lemmas (**in** *semicategory*) *smc-pattern-rectangle-left* =
smc-pattern-rectangle-right[*symmetric*]

4.2.5 Monic arrow and epic arrow

See Chapter I-5 in [39].

definition *is-monic-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-monic-arr* $\mathfrak{C}\ b\ c\ m \longleftrightarrow$
$m : b \mapsto_{\mathfrak{C}} c \wedge$
(
$\forall f\ g\ a.$
$f : a \mapsto_{\mathfrak{C}} b \longrightarrow g : a \mapsto_{\mathfrak{C}} b \longrightarrow m \circ_{A\mathfrak{C}} f = m \circ_{A\mathfrak{C}} g \longrightarrow f = g$
)

syntax *-is-monic-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(⟨- : - ↦$_{mon1}$ -⟩ [51, 51, 51] 51)
syntax-consts *-is-monic-arr* ⇌ *is-monic-arr*
translations $m : b \mapsto_{mon\mathfrak{C}} c$ ⇌ *CONST is-monic-arr* $\mathfrak{C}\ b\ c\ m$

definition *is-epic-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-epic-arr* $\mathfrak{C}\ a\ b\ e \equiv e : b \mapsto_{mon\,op\text{-}smc\,\mathfrak{C}} a$

syntax *-is-epic-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(⟨- : - ↦$_{epi1}$ -⟩ [51, 51, 51] 51)
syntax-consts *-is-epic-arr* ⇌ *is-epic-arr*
translations $e : a \mapsto_{epi\mathfrak{C}} b$ ⇌ *CONST is-epic-arr* $\mathfrak{C}\ a\ b\ e$

Rules.

mk-ide rf *is-monic-arr-def*
|*intro is-monic-arrI*|
|*dest is-monic-arrD*[*dest*]|
|*elim is-monic-arrE*[*elim*!]|

lemmas [*smc-arrow-cs-intros*] = *is-monic-arrD*(1)

lemma (**in** *semicategory*) *is-epic-arrI*:
assumes $e : a \mapsto_{\mathfrak{C}} b$
and $\bigwedge f\ g\ c.\ [\![f : b \mapsto_{\mathfrak{C}} c;\ g : b \mapsto_{\mathfrak{C}} c;\ f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e]\!] \Longrightarrow$
$f = g$

shows $e : a \mapsto_{epi\mathfrak{C}} b$
⟨*proof*⟩

lemma *is-epic-arr-is-arr*[*smc-arrow-cs-intros*, *dest*]:
assumes $e : a \mapsto_{epi\mathfrak{C}} b$
shows $e : a \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *semicategory*) *is-epic-arrD*[*dest*]:
assumes $e : a \mapsto_{epi\mathfrak{C}} b$
shows $e : a \mapsto_{\mathfrak{C}} b$
and $\bigwedge f\ g\ c.\ [\![f : b \mapsto_{\mathfrak{C}} c;\ g : b \mapsto_{\mathfrak{C}} c;\ f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e]\!] \Longrightarrow f = g$
⟨*proof*⟩

lemma (**in** *semicategory*) *is-epic-arrE*[*elim*!]:
assumes $e : a \mapsto_{epi\mathfrak{C}} b$
obtains $e : a \mapsto_{\mathfrak{C}} b$
and $\bigwedge f\ g\ c.\ [\![f : b \mapsto_{\mathfrak{C}} c;\ g : b \mapsto_{\mathfrak{C}} c;\ f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e]\!] \Longrightarrow f = g$
⟨*proof*⟩

Elementary properties.

lemma (**in** *semicategory*) *op-smc-is-epic-arr*[*smc-op-simps*]:
$f : b \mapsto_{epi\, op\text{-}smc\ \mathfrak{C}} a \longleftrightarrow f : a \mapsto_{mon\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *semicategory*) *op-smc-is-monic-arr*[*smc-op-simps*]:
$f : b \mapsto_{mon\, op\text{-}smc\ \mathfrak{C}} a \longleftrightarrow f : a \mapsto_{epi\mathfrak{C}} b$
⟨*proof*⟩

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

lemmas [*smc-arrow-cs-intros*] = *semicategory.smc-Comp-is-monic-arr*

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

lemmas [*smc-arrow-cs-intros*] = *semicategory.smc-Comp-is-epic-arr*

lemma (**in** *semicategory*) *smc-Comp-is-monic-arr-is-monic-arr*:
assumes $g : b \mapsto_{\mathfrak{C}} c$ **and** $f : a \mapsto_{\mathfrak{C}} b$ **and** $g \circ_{A\mathfrak{C}} f : a \mapsto_{mon\mathfrak{C}} c$
shows $f : a \mapsto_{mon\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-Comp-is-epic-arr-is-epic-arr*:
assumes $g : a \mapsto_{\mathfrak{C}} b$ **and** $f : b \mapsto_{\mathfrak{C}} c$ **and** $f \circ_{A\mathfrak{C}} g : a \mapsto_{epi\mathfrak{C}} c$
shows $f : b \mapsto_{epi\mathfrak{C}} c$
⟨*proof*⟩

4.2.6 Idempotent arrow

See Chapter I-5 in [39].

definition *is-idem-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow bool$
  where *is-idem-arr* $\mathfrak{C}\ b\ f \longleftrightarrow f : b \mapsto_{\mathfrak{C}} b \wedge f \circ_{A\mathfrak{C}} f = f$

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

Rules.

mk-ide rf *is-idem-arr-def*
  |*intro is-idem-arrI*|
  |*dest is-idem-arrD*[*dest*]|
  |*elim is-idem-arrE*[*elim*!]|

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

Elementary properties.

lemma (**in** *semicategory*) *op-smc-is-idem-arr*[*smc-op-simps*]:
  $f : \mapsto_{ide\, op\text{-}smc\ \mathfrak{C}} b \longleftrightarrow f : \mapsto_{ide\mathfrak{C}} b$
  ⟨*proof*⟩

4.2.7 Terminal object and initial object

See Chapter I-5 in [39].

definition *obj-terminal* :: $V \Rightarrow V \Rightarrow bool$
  where *obj-terminal* $\mathfrak{C}\ t \longleftrightarrow$
    $t \in_{\circ} \mathfrak{C}(\!|Obj|\!) \wedge (\forall a.\ a \in_{\circ} \mathfrak{C}(\!|Obj|\!) \longrightarrow (\exists!f.\ f : a \mapsto_{\mathfrak{C}} t))$

definition *obj-initial* :: $V \Rightarrow V \Rightarrow bool$
  where *obj-initial* $\mathfrak{C} \equiv$ *obj-terminal* (*op-smc* $\mathfrak{C}$)

Rules.

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

lemma *obj-initialI*:
  assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\bigwedge b.\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \exists!f.\ f : a \mapsto_{\mathfrak{C}} b$
  shows *obj-initial* $\mathfrak{C}\ a$
  ⟨*proof*⟩

lemma *obj-initialD*[*dest*]:
  assumes *obj-initial* $\mathfrak{C}\ a$
  shows $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\bigwedge b.\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \exists!f.\ f : a \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

lemma *obj-initialE*[*elim*]:
  assumes *obj-initial* $\mathfrak{C}\ a$
  obtains $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\bigwedge b.\ b \in_{\circ} \mathfrak{C}(\!|Obj|\!) \Longrightarrow \exists!f.\ f : a \mapsto_{\mathfrak{C}} b$
  ⟨*proof*⟩

Elementary properties.

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

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

4.2.8 Null object

See Chapter I-5 in [39].

definition *obj-null* :: $V \Rightarrow V \Rightarrow$ *bool*
where *obj-null* $\mathfrak{C}$ $a \longleftrightarrow$ *obj-initial* $\mathfrak{C}$ $a \wedge$ *obj-terminal* $\mathfrak{C}$ a

Rules.

mk-ide rf *obj-null-def*
|*intro obj-nullI*|
|*dest obj-nullD*[*dest*]|
|*elim obj-nullE*[*elim*]|

Elementary properties.

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

4.2.9 Zero arrow

definition *is-zero-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-zero-arr* $\mathfrak{C}$ a b $h \longleftrightarrow$
$(\exists z\ g\ f.$ *obj-null* $\mathfrak{C}$ $z \wedge h = g \circ_{A\mathfrak{C}} f \wedge f : a \mapsto_{\mathfrak{C}} z \wedge g : z \mapsto_{\mathfrak{C}} b)$

syntax *-is-zero-arr* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
$(\langle$- : - $\mapsto_{01}$ -$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-zero-arr* ⇌ *is-zero-arr*
translations $h : a \mapsto_{0\mathfrak{C}} b \rightleftharpoons$ *CONST is-zero-arr* $\mathfrak{C}$ a b h

Rules.

lemma *is-zero-arrI*:
assumes *obj-null* $\mathfrak{C}$ z
and $h = g \circ_{A\mathfrak{C}} f$
and $f : a \mapsto_{\mathfrak{C}} z$
and $g : z \mapsto_{\mathfrak{C}} b$
shows $h : a \mapsto_{0\mathfrak{C}} b$
⟨*proof*⟩

lemma *is-zero-arrD*[*dest*]:
assumes $h : a \mapsto_{0\mathfrak{C}} b$
shows $\exists z\ g\ f.$ *obj-null* $\mathfrak{C}$ $z \wedge h = g \circ_{A\mathfrak{C}} f \wedge f : a \mapsto_{\mathfrak{C}} z \wedge g : z \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

lemma *is-zero-arrE*[*elim*]:
assumes $h : a \mapsto_{0\mathfrak{C}} b$
obtains $z\ g\ f$
where *obj-null* $\mathfrak{C}$ z
and $h = g \circ_{A\mathfrak{C}} f$
and $f : a \mapsto_{\mathfrak{C}} z$
and $g : z \mapsto_{\mathfrak{C}} b$
⟨*proof*⟩

Elementary properties.

lemma (in *semicategory*) *op-smc-is-zero-arr*[*smc-op-simps*]:

$f : b \mapsto_{0\,op\text{-}smc\ \mathfrak{C}} a \longleftrightarrow f : a \mapsto_{0\mathfrak{C}} b$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-is-zero-arr-Comp-right*:
assumes $h : b \mapsto_{0\mathfrak{C}} c$ **and** $h' : a \mapsto_{\mathfrak{C}} b$
shows $h \circ_{A\mathfrak{C}} h' : a \mapsto_{0\mathfrak{C}} c$
⟨*proof*⟩

lemmas [*smc-arrow-cs-intros*] = *semicategory.smc-is-zero-arr-Comp-right*

lemma (**in** *semicategory*) *smc-is-zero-arr-Comp-left*:
assumes $h' : b \mapsto_{\mathfrak{C}} c$ **and** $h : a \mapsto_{0\mathfrak{C}} b$
shows $h' \circ_{A\mathfrak{C}} h : a \mapsto_{0\mathfrak{C}} c$
⟨*proof*⟩

lemmas [*smc-arrow-cs-intros*] = *semicategory.smc-is-zero-arr-Comp-left*

4.3 Smallness for semicategories

4.3.1 Background

An explanation of the methodology chosen for the exposition of all matters related to the size of the semicategories and associated entities is given in the previous chapter.

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

4.3.2 Tiny semicategory

Definition and elementary properties

locale *tiny-semicategory* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *Comp*: *vsv* ‹$\mathfrak{C}(\!|Comp|\!)$› **for** α $\mathfrak{C}$ +
  assumes *tiny-smc-length*[*smc-cs-simps*]: *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
    and *tiny-smc-tiny-digraph*[*slicing-intros*]: *tiny-digraph* α (*smc-dg* $\mathfrak{C}$)
    and *tiny-smc-Comp-vdomain*: $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
      $(\exists g\ f\ b\ c\ a.\ gf = [g, f]_{\circ} \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
    and *tiny-smc-Comp-is-arr*[*smc-cs-intros*]:
      $[\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
    and *tiny-smc-assoc*[*smc-cs-simps*]:
      $[\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
        $(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

lemmas [*smc-cs-simps*] =
  tiny-semicategory.tiny-smc-length
  tiny-semicategory.tiny-smc-assoc

lemmas [*slicing-intros*] =
  tiny-semicategory.tiny-smc-Comp-is-arr

Rules.

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

mk-ide rf *tiny-semicategory-def*[*unfolded tiny-semicategory-axioms-def*]
  |*intro tiny-semicategoryI*|
  |*dest tiny-semicategoryD*[*dest*]|
  |*elim tiny-semicategoryE*[*elim*]|

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

lemma *tiny-semicategoryI''*:
  assumes $\mathcal{Z}$ α
    and *vfsequence* $\mathfrak{C}$
    and *vsv* ($\mathfrak{C}(\!|Comp|\!)$)
    and *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
    and *vsv* ($\mathfrak{C}(\!|Dom|\!)$)
    and *vsv* ($\mathfrak{C}(\!|Cod|\!)$)
    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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
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 $\mathfrak{C}(\!|Obj|\!) \in_\circ Vset\ \alpha$
and $\mathfrak{C}(\!|Arr|\!) \in_\circ Vset\ \alpha$
shows *tiny-semicategory* $\alpha\ \mathfrak{C}$
⟨*proof*⟩

Slicing.

context *tiny-semicategory*
begin

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

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

end

Elementary properties.

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

lemmas (**in** *tiny-semicategory*) *tiny-smc-semicategory* = *semicategory-axioms*

lemmas [*smc-small-cs-intros*] = *tiny-semicategory.tiny-smc-semicategory*

Size.

lemma (**in** *tiny-semicategory*) *tiny-smc-Comp-in-Vset*: $\mathfrak{C}(\!|Comp|\!) \in_\circ Vset\ \alpha$
⟨*proof*⟩

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

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

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

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

Opposite tiny semicategory

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

lemmas *tiny-semicategory-op*[*smc-op-intros*] =

tiny-semicategory.tiny-semicategory-op

4.3.3 Finite semicategory

Definition and elementary properties

A finite semicategory is a generalization of the concept of a finite category, as presented in nLab [3] [3].

locale *finite-semicategory* = $\mathcal{Z}$ α + *vfsequence* $\mathfrak{C}$ + *Comp*: *vsv* ‹$\mathfrak{C}(\!|Comp|\!)$› **for** α $\mathfrak{C}$ +
  assumes *fin-smc-length*[*smc-cs-simps*]: *vcard* $\mathfrak{C} = 5_{\mathbb{N}}$
    and *fin-smc-finite-digraph*[*slicing-intros*]: *finite-digraph* α (*smc-dg* $\mathfrak{C}$)
    and *fin-smc-Comp-vdomain*: $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\!|Comp|\!)) \longleftrightarrow$
      $(\exists g\ f\ b\ c\ a.\ gf = [g, f]_{\circ} \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$
    and *fin-smc-Comp-is-arr*[*smc-cs-intros*]:
      $[\![\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$
    and *fin-smc-assoc*[*smc-cs-simps*]:
      $[\![\ h : c \mapsto_{\mathfrak{C}} d;\ g : b \mapsto_{\mathfrak{C}} c;\ f : a \mapsto_{\mathfrak{C}} b\ ]\!] \Longrightarrow$
        $(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

lemmas [*smc-cs-simps*] =
  finite-semicategory.fin-smc-length
  finite-semicategory.fin-smc-assoc

lemmas [*slicing-intros*] =
  finite-semicategory.fin-smc-Comp-is-arr

Rules.

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

mk-ide rf *finite-semicategory-def*[*unfolded finite-semicategory-axioms-def*]
  |*intro finite-semicategoryI*|
  |*dest finite-semicategoryD*[*dest*]|
  |*elim finite-semicategoryE*[*elim*]|

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

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

Slicing.

context *finite-semicategory*
begin

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

lemmas-with [*unfolded slicing-simps*]:
  fin-smc-Obj-vfinite[*smc-small-cs-intros*] = *dg.fin-dg-Obj-vfinite*

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

and *fin-smc-Arr-vfinite*[*smc-small-cs-intros*] = *dg.fin-dg-Arr-vfinite*

end

Elementary properties.

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

lemmas (**in** *finite-semicategory*) *fin-smc-tiny-semicategory* =
tiny-semicategory-axioms

lemmas [*smc-small-cs-intros*] = *finite-semicategory.fin-smc-tiny-semicategory*

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

Size.

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

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

Opposite finite semicategory

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

lemmas *finite-semicategory-op*[*smc-op-intros*] =
finite-semicategory.finite-semicategory-op

4.4 Semifunctor

4.4.1 Background

named-theorems *smcf-cs-simps*
named-theorems *smcf-cs-intros*

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

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

Slicing

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

Components.

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

4.4.2 Definition and elementary properties

See Chapter I-3 in [39] and the description of the concept of a digraph homomorphism in the previous chapter.

locale *is-semifunctor* =
  $\mathcal{Z}$ α +
  vfsequence $\mathfrak{F}$ +
  HomDom: *semicategory* α $\mathfrak{A}$ +
  HomCod: *semicategory* α $\mathfrak{B}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *smcf-length*[*smc-cs-simps*]: *vcard* $\mathfrak{F}$ = $4_{\mathbf{N}}$
    and *smcf-is-dghm*[*slicing-intros*]:
      smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{DG\alpha}$ *smc-dg* $\mathfrak{B}$
    and *smcf-HomDom*[*smc-cs-simps*]: $\mathfrak{F}(\!|HomDom|\!)$ = $\mathfrak{A}$
    and *smcf-HomCod*[*smc-cs-simps*]: $\mathfrak{F}(\!|HomCod|\!)$ = $\mathfrak{B}$
    and *smcf-ArrMap-Comp*[*smc-cs-simps*]: $[\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
      $\mathfrak{F}(\!|ArrMap|\!)(\!|g \circ_{A\mathfrak{A}} f|\!) = \mathfrak{F}(\!|ArrMap|\!)(\!|g|\!) \circ_{A\mathfrak{B}} \mathfrak{F}(\!|ArrMap|\!)(\!|f|\!)$

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

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

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

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

abbreviation *smcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *smcfs* $\alpha\ \mathfrak{A}\ \mathfrak{B} \equiv$ *set* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}\}$

lemmas [*smc-cs-simps*] =
is-semifunctor.smcf-HomDom
is-semifunctor.smcf-HomCod
is-semifunctor.smcf-ArrMap-Comp

lemma *smcf-is-dghm'*[*slicing-intros*]:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{A}' =$ *smc-dg* $\mathfrak{A}$
and $\mathfrak{B}' =$ *smc-dg* $\mathfrak{B}$
shows *smcf-dghm* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{DG\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemma *cn-dghm-comp-is-dghm*:
assumes $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows *smcf-dghm* $\mathfrak{F}$: *op-dg* (*smc-dg* $\mathfrak{A}$) $\mapsto\mapsto_{DG\alpha}$ *smc-dg* $\mathfrak{B}$
⟨*proof*⟩

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

Rules.

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

mk-ide rf *is-semifunctor-def*[*unfolded is-semifunctor-axioms-def*]
|*intro is-semifunctorI*|
|*dest is-semifunctorD*[*dest*]|
|*elim is-semifunctorE*[*elim*]|

lemmas [*smc-cs-intros*] =
is-semifunctorD(3,4)

lemma *is-semifunctorI'*:
assumes $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{F}$
and *semicategory* $\alpha\ \mathfrak{A}$
and *semicategory* $\alpha\ \mathfrak{B}$
and *vcard* $\mathfrak{F} = 4_{\mathbb{N}}$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-semifunctorD′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{F}$
and *semicategory* $\alpha\ \mathfrak{A}$
and *semicategory* $\alpha\ \mathfrak{B}$
and *vcard* $\mathfrak{F} = 4_{\mathbb{N}}$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
⟨*proof*⟩

lemma *is-semifunctorE′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
obtains $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{F}$
and *semicategory* $\alpha\ \mathfrak{A}$
and *semicategory* $\alpha\ \mathfrak{B}$
and *vcard* $\mathfrak{F} = 4_{\mathbb{N}}$
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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{A}} c;\ f : a \mapsto_{\mathfrak{A}} b\ ]\!] \Longrightarrow$
$\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \circ_{A\mathfrak{A}} f \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
⟨*proof*⟩

Slicing.

context *is-semifunctor*
begin

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

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

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

end

lemmas [*smc-cs-simps*] =
is-semifunctor.smcf-ObjMap-vdomain
is-semifunctor.smcf-ArrMap-vdomain

lemmas [*smc-cs-intros*] =
is-semifunctor.smcf-ObjMap-app-in-HomCod-Obj
is-semifunctor.smcf-ArrMap-app-in-HomCod-Arr
is-semifunctor.smcf-ArrMap-is-arr'

Elementary properties.

lemma *cn-smcf-ArrMap-Comp*[*smc-cs-simps*]:
assumes *semicategory* α $\mathfrak{A}$
and $\mathfrak{F}$: *op-smc* $\mathfrak{A}$ $\mapsto\mapsto_{SMC\alpha}$ $\mathfrak{B}$
and $g : c \mapsto_{\mathfrak{A}} b$
and $f : b \mapsto_{\mathfrak{A}} a$
shows $\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \circ_{A\mathfrak{A}} g \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor g \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)$
⟨*proof*⟩

lemma *smcf-eqI*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
and $\mathfrak{G}(\lfloor ObjMap \rfloor) = \mathfrak{F}(\lfloor ObjMap \rfloor)$
and $\mathfrak{G}(\lfloor ArrMap \rfloor) = \mathfrak{F}(\lfloor ArrMap \rfloor)$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$
shows $\mathfrak{G} = \mathfrak{F}$
⟨*proof*⟩

lemma *smcf-dghm-eqI*:
assumes $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
and $\mathfrak{A} = \mathfrak{C}$
and $\mathfrak{B} = \mathfrak{D}$

and *smcf-dghm* $\mathfrak{G}$ = *smcf-dghm* $\mathfrak{F}$
shows $\mathfrak{G} = \mathfrak{F}$
⟨*proof*⟩

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

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

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

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

lemma (**in** *is-semifunctor*) *smcf-in-Vset-7*: $\mathfrak{F} \in_\circ Vset\ (\alpha + 7_{\mathbf{N}})$
⟨*proof*⟩

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

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

4.4.3 Opposite semifunctor

Definition and elementary properties

See Chapter II-2 in [39].

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

Components.

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

Slicing.

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

Further properties

lemma (**in** *is-semifunctor*) *is-semifunctor-op*:

op-smcf $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC\alpha}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

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

lemmas *is-semifunctor-op'*[*smc-op-intros*] = *is-semifunctor.is-semifunctor-op'*

lemma (**in** *is-semifunctor*) *smcf-op-smcf-op-smcf*[*smc-op-simps*]:
op-smcf (*op-smcf* $\mathfrak{F}$) = $\mathfrak{F}$
⟨*proof*⟩

lemmas *smcf-op-smcf-op-smcf*[*smc-op-simps*] = *is-semifunctor.smcf-op-smcf-op-smcf*

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

4.4.4 Composition of covariant semifunctors

Definition and elementary properties

abbreviation (*input*) *smcf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{SMCF}$› 55)
where *smcf-comp* ≡ *dghm-comp*

Slicing.

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

Object map

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

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

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

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

Arrow map

lemma *smcf-comp-ArrMap-vsv*[*smc-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows *vsv* $((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\lfloor ArrMap \rfloor))$
⟨*proof*⟩

lemma *smcf-comp-ArrMap-vdomain*[*smc-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{D}_\circ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\lfloor ArrMap \rfloor)) = \mathfrak{A}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *smcf-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\lfloor ArrMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *smcf-comp-ArrMap-app*[*smc-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and [*simp*]: $f \in_\circ \mathfrak{A}(\lfloor Arr \rfloor)$
shows $(\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \rfloor)$
⟨*proof*⟩

Further properties

lemma *smcf-comp-is-semifunctor*[*smc-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

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

lemma *op-smcf-smcf-comp*[*smc-op-simps*]:
op-smcf $(\mathfrak{G} \circ_{SMCF} \mathfrak{F}) =$ *op-smcf* $\mathfrak{G} \circ_{SMCF}$ *op-smcf* $\mathfrak{F}$
⟨*proof*⟩

4.4.5 Composition of contravariant semifunctors

Definition and elementary properties

See section 1.2 in [15].

definition *smcf-cn-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle_{SMCF\circ}\rangle$ 55)
where $\mathfrak{G}\ {}_{SMCF\circ}\ \mathfrak{F} =$
[
$\mathfrak{G}(\lfloor ObjMap \rfloor) \circ_\circ \mathfrak{F}(\lfloor ObjMap \rfloor)$,
$\mathfrak{G}(\lfloor ArrMap \rfloor) \circ_\circ \mathfrak{F}(\lfloor ArrMap \rfloor)$,
op-smc $(\mathfrak{F}(\lfloor HomDom \rfloor))$,
$\mathfrak{G}(\lfloor HomCod \rfloor)$
]$_\circ$

Components.

lemma *smcf-cn-comp-components*:

shows $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap\rfloor) = \mathfrak{G}(\lfloor ObjMap\rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ObjMap\rfloor)$
and $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap\rfloor) = \mathfrak{G}(\lfloor ArrMap\rfloor) \circ_{\circ} \mathfrak{F}(\lfloor ArrMap\rfloor)$
and [*smc-cn-cs-simps*]: $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor HomDom\rfloor) = op\text{-}smc\ (\mathfrak{F}(\lfloor HomDom\rfloor))$
and [*smc-cn-cs-simps*]: $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor HomCod\rfloor) = \mathfrak{G}(\lfloor HomCod\rfloor)$
⟨*proof*⟩

Slicing.

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

Object map: two contravariant semifunctors

lemma *smcf-cn-comp-ObjMap-vsv*[*smc-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $vsv\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap\rfloor))$
⟨*proof*⟩

lemma *smcf-cn-comp-ObjMap-vdomain*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap\rfloor)) = \mathfrak{A}(\lfloor Obj\rfloor)$
⟨*proof*⟩

lemma *smcf-cn-comp-ObjMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap\rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Obj\rfloor)$
⟨*proof*⟩

lemma *smcf-cn-comp-ObjMap-app*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\lfloor Obj\rfloor)$
shows $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap\rfloor)(\lfloor a\rfloor) = \mathfrak{G}(\lfloor ObjMap\rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap\rfloor)(\lfloor a\rfloor)\rfloor)$
⟨*proof*⟩

Arrow map: two contravariant semifunctors

lemma *smcf-cn-comp-ArrMap-vsv*[*smc-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $vsv\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap\rfloor))$
⟨*proof*⟩

lemma *smcf-cn-comp-ArrMap-vdomain*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap\rfloor)) = \mathfrak{A}(\lfloor Arr\rfloor)$
⟨*proof*⟩

lemma *smcf-cn-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap\rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Arr\rfloor)$
⟨*proof*⟩

lemma *smcf-cn-comp-ArrMap-app*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}\mapsto\mapsto_{\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\lfloor Arr\rfloor)$
shows $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap\rfloor)(\lfloor a\rfloor) = \mathfrak{G}(\lfloor ArrMap\rfloor)(\lfloor \mathfrak{F}(\lfloor ArrMap\rfloor)(\lfloor a\rfloor)\rfloor)$
⟨*proof*⟩

Object map: contravariant and covariant semifunctors

lemma *smcf-cn-cov-comp-ObjMap-vsv*[*smc-cn-cs-intros*]:

assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap \rfloor))$
⟨*proof*⟩

lemma *smcf-cn-cov-comp-ObjMap-vdomain*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

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

lemma *smcf-cn-cov-comp-ObjMap-app*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$ **and** $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
shows $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) = \mathfrak{G}(\lfloor ObjMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor) \rfloor)$
⟨*proof*⟩

Arrow map: contravariant and covariant semifunctors

lemma *smcf-cn-cov-comp-ArrMap-vsv*[*smc-cn-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows *vsv* $((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap \rfloor))$
⟨*proof*⟩

lemma *smcf-cn-cov-comp-ArrMap-vdomain*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows $\mathcal{D}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap \rfloor)) = \mathfrak{A}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *smcf-cn-cov-comp-ArrMap-vrange*:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap \rfloor)) \subseteq_{\circ} \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

lemma *smcf-cn-cov-comp-ArrMap-app*[*smc-cn-cs-simps*]:
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$ **and** $f \in_{\circ} \mathfrak{A}(\lfloor Arr \rfloor)$
shows $(\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \rfloor)$
⟨*proof*⟩

Opposite of the contravariant composition of semifunctors

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

Further properties

lemma *smcf-cn-comp-is-semifunctor*[*smc-cn-cs-intros*]:
— See section 1.2 in [15].
assumes *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{B}$
shows $\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{C}$
⟨*proof*⟩

lemma *smcf-cn-cov-comp-is-semifunctor*[*smc-cs-intros*]:
— See section 1.2 in [15].
assumes $\mathfrak{G} : \mathfrak{B}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A}\ {\mapsto\!\!\mapsto}_{SMC\alpha}\ \mathfrak{B}$
shows $\mathfrak{G}\ _{SMCF}\circ\ \mathfrak{F} : \mathfrak{A}\ _{SMC}{\mapsto\!\!\mapsto}_{\alpha}\ \mathfrak{C}$

⟨*proof*⟩

lemma *smcf-cov-cn-comp-is-semifunctor*[*smc-cn-cs-intros*]:
— See section 1.2 in [15].
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\!\!\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \ _{SMC}\!\!\mapsto\!\!\mapsto_{\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \ _{SMC}\!\!\mapsto\!\!\mapsto_{\alpha} \mathfrak{C}$
⟨*proof*⟩

4.4.6 Identity semifunctor

Definition and elementary properties

See Chapter I-3 in [39].

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

Slicing.

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

context *semicategory*
begin

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

lemmas-with [*unfolded slicing-simps*]:
smc-dghm-id-is-dghm = *dg.dg-dghm-id-is-dghm*

end

Object map

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

Arrow map

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

Opposite identity semifunctor

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

An identity semifunctor is a semifunctor

lemma (**in** *semicategory*) *smc-smcf-id-is-semifunctor*: *smcf-id* $\mathfrak{C} : \mathfrak{C} \mapsto\!\!\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-smcf-id-is-semifunctor′*:
assumes $\mathfrak{A}$ = $\mathfrak{C}$ **and** $\mathfrak{B}$ = $\mathfrak{C}$
shows *smcf-id* $\mathfrak{C} : \mathfrak{A} \mapsto\!\!\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemmas [*smc-cs-intros*] = *semicategory.smc-smcf-id-is-semifunctor′*

Further properties

lemma (**in** *is-semifunctor*) *smcf-smcf-comp-smcf-id-left*[*smc-cs-simps*]:

— See Chapter I-3 in [39]).
smcf-id $\mathfrak{B}$ $\circ_{SMCF}$ $\mathfrak{F}$ = $\mathfrak{F}$
⟨*proof*⟩

lemmas [*smc-cs-simps*] = *is-semifunctor.smcf-smcf-comp-smcf-id-left*

lemma (**in** *is-semifunctor*) *smcf-smcf-comp-smcf-id-right*[*smc-cs-simps*]:
— See Chapter I-3 in [39]).
$\mathfrak{F}$ $\circ_{SMCF}$ *smcf-id* $\mathfrak{A}$ = $\mathfrak{F}$
⟨*proof*⟩

lemmas [*smc-cs-simps*] = *is-semifunctor.smcf-smcf-comp-smcf-id-right*

4.4.7 Constant semifunctor

Definition and elementary properties

See Chapter III-3 in [39].

abbreviation (*input*) *smcf-const* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *smcf-const* ≡ *dghm-const*

Slicing.

lemma *smcf-dghm-smcf-const*[*slicing-commute*]:
dghm-const (*smc-dg* $\mathfrak{C}$) (*smc-dg* $\mathfrak{D}$) *a f* = *smcf-dghm* (*smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*)
⟨*proof*⟩

Object map

lemmas [*smc-cs-simps*] =
dghm-const-ObjMap-app

Arrow map

lemmas [*smc-cs-simps*] =
dghm-const-ArrMap-app

Opposite constant semifunctor

lemma *op-smcf-smcf-const*[*smc-op-simps*]:
op-smcf (*smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f*) = *smcf-const* (*op-smc* $\mathfrak{C}$) (*op-smc* $\mathfrak{D}$) *a f*
⟨*proof*⟩

A constant semifunctor is a semifunctor

lemma *smcf-const-is-semifunctor*:
assumes *semicategory* α $\mathfrak{C}$
and *semicategory* α $\mathfrak{D}$
and *f* : *a* $\mapsto_{\mathfrak{D}}$ *a*
and [*simp*]: *f* $\circ_{A\mathfrak{D}}$ *f* = *f*
shows *smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ *a f* : $\mathfrak{C}$ $\mapsto\mapsto_{SMC\alpha}$ $\mathfrak{D}$
⟨*proof*⟩

lemma *smcf-const-is-semifunctor'*[*smc-cs-intros*]:
assumes *semicategory* α $\mathfrak{C}$
and *semicategory* α $\mathfrak{D}$
and *f* : *a* $\mapsto_{\mathfrak{D}}$ *a*
and *f* $\circ_{A\mathfrak{D}}$ *f* = *f*
and $\mathfrak{A}$ = $\mathfrak{C}$

and $\mathfrak{B} = \mathfrak{D}$
shows *smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ a $f : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

Further properties

lemma (in *is-semifunctor*) *smcf-smcf-comp-smcf-const*[*smc-cs-simps*]:
assumes *semicategory* α $\mathfrak{C}$ **and** $f : a \mapsto_{\mathfrak{C}} a$ **and** $f \circ_{A\mathfrak{C}} f = f$
shows *smcf-const* $\mathfrak{B}$ $\mathfrak{C}$ a $f \circ_{SMCF} \mathfrak{F}$ = *smcf-const* $\mathfrak{A}$ $\mathfrak{C}$ a f
⟨*proof*⟩

lemmas [*smc-cs-simps*] = *is-semifunctor.smcf-smcf-comp-smcf-const*

4.4.8 Faithful semifunctor

Definition and elementary properties

See Chapter I-3 in [39].

locale *is-ft-semifunctor* = *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *ft-smcf-is-ft-dghm*:
smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A}$ $\mapsto\mapsto_{DG.faithful\alpha}$ *smc-dg* $\mathfrak{B}$

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

lemma (in *is-ft-semifunctor*) *ft-smcf-is-ft-dghm'*[*slicing-intros*]:
assumes $\mathfrak{A}'$ = *smc-dg* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *smc-dg* $\mathfrak{B}$
shows *smcf-dghm* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{DG.faithful\alpha} \mathfrak{B}'$
⟨*proof*⟩

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

Rules.

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

mk-ide rf *is-ft-semifunctor-def*[*unfolded is-ft-semifunctor-axioms-def*]
|*intro is-ft-semifunctorI*|
|*dest is-ft-semifunctorD*[*dest*]|
|*elim is-ft-semifunctorE*[*elim*]|

lemmas [*smcf-cs-intros*] = *is-ft-semifunctorD*(1)

lemma *is-ft-semifunctorI'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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 $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-ft-semifunctorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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-semifunctorE′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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-semifunctorI″*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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 $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}$
⟨*proof*⟩

Elementary properties.

context *is-ft-semifunctor*
begin

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

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

end

Opposite faithful semifunctor

lemma (**in** *is-ft-semifunctor*) *is-ft-semifunctor-op*:
op-smcf $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-ft-semifunctor*) *is-ft-semifunctor-op′*[*smc-op-intros*]:
assumes $\mathfrak{A}'$ = *op-smc* $\mathfrak{A}$ **and** $\mathfrak{B}'$ = *op-smc* $\mathfrak{B}$
shows *op-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-ft-semifunctor-op*[*smc-op-intros*] =
is-ft-semifunctor.is-ft-semifunctor-op′

The composition of faithful semifunctors is a faithful semifunctor

lemma *smcf-comp-is-ft-semifunctor*[*smcf-cs-intros*]:
— See Chapter I-3 in [39].
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.faithful\alpha} \mathfrak{C}$
⟨*proof*⟩

4.4.9 Full semifunctor

Definition and elementary properties

See Chapter I-3 in [39].

locale *is-fl-semifunctor* = *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *fl-smcf-is-fl-dghm*:
smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A} \mapsto\mapsto_{DG.full\alpha}$ *smc-dg* $\mathfrak{B}$

syntax *-is-fl-semifunctor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
($\langle$(- :/ - $\mapsto\mapsto_{SMC.full^1}$ -)$\rangle$ [51, 51, 51] 51)
syntax-consts *-is-fl-semifunctor* $\rightleftharpoons$ *is-fl-semifunctor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{B} \rightleftharpoons CONST$ *is-fl-semifunctor* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

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

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

Rules.

mk-ide rf *is-fl-semifunctor-def*[*unfolded is-fl-semifunctor-axioms-def*]
|*intro is-fl-semifunctorI*|
|*dest is-fl-semifunctorD*[*dest*]|
|*elim is-fl-semifunctorE*[*elim*]|

lemmas [*smcf-cs-intros*] = *is-fl-semifunctorD*(1)

lemma *is-fl-semifunctorI'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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 $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-fl-semifunctorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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-semifunctorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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-semifunctor*
begin

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

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

end

Opposite full semifunctor

lemma (in *is-fl-semifunctor*) *is-fl-semifunctor-op*:
op-smcf $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.full\alpha}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

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

lemmas *is-fl-semifunctor-op*[*smc-op-intros*] =
is-fl-semifunctor.is-fl-semifunctor-op

The composition of full semifunctors is a full semifunctor

lemma *smcf-comp-is-fl-semifunctor*[*smcf-cs-intros*]:
— See Chapter I-3 in [39].
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.full\alpha} \mathfrak{C}$
⟨*proof*⟩

4.4.10 Fully faithful semifunctor

Definition and elementary properties

See Chapter I-3 in [39]).

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

syntax *-is-ff-semifunctor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - ↦↦$_{SMC.ff^1}$ -)› [51, 51, 51] 51)
syntax-consts *-is-ff-semifunctor* ⇌ *is-ff-semifunctor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.ff\alpha} \mathfrak{B} \rightleftharpoons CONST\ is\text{-}ff\text{-}semifunctor\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}$

Rules.

mk-ide rf *is-ff-semifunctor-def*
|*intro is-ff-semifunctorI*|
|*dest is-ff-semifunctorD*[*dest*]|
|*elim is-ff-semifunctorE*[*elim*]|

lemmas [*smcf-cs-intros*] = *is-ff-semifunctorD*

Elementary properties.

lemma (in *is-ff-semifunctor*) *ff-smcf-is-ff-dghm*:
smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A} \mapsto\mapsto_{DG.ff\alpha}$ *smc-dg* $\mathfrak{B}$
⟨*proof*⟩

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

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

Opposite fully faithful semifunctor

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

lemma (in *is-ff-semifunctor*) *is-ff-semifunctor-op'*[*smc-op-intros*]:
assumes $\mathfrak{A}' = op\text{-}smc\ \mathfrak{A}$ **and** $\mathfrak{B}' = op\text{-}smc\ \mathfrak{B}$

shows *op-smcf* $\mathfrak{F} : \mathfrak{A}' \mapsto\mapsto_{SMC.ff\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-ff-semifunctor-op*[*smc-op-intros*] =
is-ff-semifunctor.is-ff-semifunctor-op'

The composition of fully faithful semifunctors is a fully faithful semifunctor

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

4.4.11 Isomorphism of semicategories

Definition and elementary properties

See Chapter I-3 in [39].

locale *is-iso-semifunctor* = *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
assumes *iso-smcf-is-iso-dghm*:
smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A} \mapsto\mapsto_{DG.iso\alpha}$ *smc-dg* $\mathfrak{B}$

syntax *-is-iso-semifunctor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - ↦↦$_{SMC.iso1}$ -)› [51, 51, 51] 51)
syntax-consts *-is-iso-semifunctor* ⇌ *is-iso-semifunctor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$ ⇌ *CONST is-iso-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

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

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

Rules.

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

mk-ide rf *is-iso-semifunctor-def*[*unfolded is-iso-semifunctor-axioms-def*]
|*intro is-iso-semifunctorI*|
|*dest is-iso-semifunctorD*[*dest*]|
|*elim is-iso-semifunctorE*[*elim*]|

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

lemma *is-iso-semifunctorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *v11* (𝔉(|*ObjMap*|))

and $v11$ ($\mathfrak{F}(\!|ArrMap|\!)$)
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ObjMap|\!)$) = $\mathfrak{B}(\!|Obj|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ArrMap|\!)$) = $\mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

lemma *is-iso-semifunctorE′*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $v11$ ($\mathfrak{F}(\!|ObjMap|\!)$)
and $v11$ ($\mathfrak{F}(\!|ArrMap|\!)$)
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ObjMap|\!)$) = $\mathfrak{B}(\!|Obj|\!)$
and $\mathcal{R}_\circ$ ($\mathfrak{F}(\!|ArrMap|\!)$) = $\mathfrak{B}(\!|Arr|\!)$
⟨*proof*⟩

Elementary properties.

context *is-iso-semifunctor*
begin

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

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

end

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

lemmas (**in** *is-iso-semifunctor*) *iso-smcf-is-ff-semifunctor* =
is-ff-semifunctor-axioms

lemmas [*smcf-cs-intros*] = *is-iso-semifunctor.iso-smcf-is-ff-semifunctor*

Opposite isomorphism of semicategories

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

lemmas *is-iso-semifunctor-op*[*smc-op-intros*] =

is-iso-semifunctor.is-iso-semifunctor-op

The composition of isomorphisms of semicategories is an isomorphism of semicategories

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

4.4.12 Inverse semifunctor

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

lemmas [*smc-cs-simps*] = *inv-dghm-components*(3,4)

Slicing.

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

context *is-iso-semifunctor*
begin

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

lemmas-with [*unfolded slicing-simps slicing-commute*]:
*inv-smcf-ObjMap-v*11 = *dghm.inv-dghm-ObjMap-v*11
and *inv-smcf-ObjMap-vdomain* = *dghm.inv-dghm-ObjMap-vdomain*
and *inv-smcf-ObjMap-app* = *dghm.inv-dghm-ObjMap-app*
and *inv-smcf-ObjMap-vrange* = *dghm.inv-dghm-ObjMap-vrange*
and *inv-smcf-ArrMap-v*11 = *dghm.inv-dghm-ArrMap-v*11
and *inv-smcf-ArrMap-vdomain* = *dghm.inv-dghm-ArrMap-vdomain*
and *inv-smcf-ArrMap-app* = *dghm.inv-dghm-ArrMap-app*
and *inv-smcf-ArrMap-vrange* = *dghm.inv-dghm-ArrMap-vrange*
and *iso-smcf-ObjMap-inv-smcf-ObjMap-app*[*smcf-cs-simps*] =
dghm.iso-dghm-ObjMap-inv-dghm-ObjMap-app
and *iso-smcf-ArrMap-inv-smcf-ArrMap-app*[*smcf-cs-simps*] =
dghm.iso-dghm-ArrMap-inv-dghm-ArrMap-app
and *iso-smcf-HomDom-is-arr-conv* = *dghm.iso-dghm-HomDom-is-arr-conv*
and *iso-smcf-HomCod-is-arr-conv* = *dghm.iso-dghm-HomCod-is-arr-conv*
and *iso-inv-smcf-ObjMap-smcf-ObjMap-app*[*smcf-cs-simps*] =
dghm.iso-inv-dghm-ObjMap-dghm-ObjMap-app
and *iso-inv-smcf-ArrMap-smcf-ArrMap-app*[*smcf-cs-simps*] =
dghm.iso-inv-dghm-ArrMap-dghm-ArrMap-app

end

lemmas [*smcf-cs-intros*] =
*is-iso-semifunctor.inv-smcf-ObjMap-v*11
*is-iso-semifunctor.inv-smcf-ArrMap-v*11

lemmas [*smcf-cs-simps*] =
is-iso-semifunctor.inv-smcf-ObjMap-vdomain
is-iso-semifunctor.inv-smcf-ObjMap-app
is-iso-semifunctor.inv-smcf-ObjMap-vrange

is-iso-semifunctor.inv-smcf-ArrMap-vdomain
is-iso-semifunctor.inv-smcf-ArrMap-app
is-iso-semifunctor.inv-smcf-ArrMap-vrange
is-iso-semifunctor.iso-smcf-ObjMap-inv-smcf-ObjMap-app
is-iso-semifunctor.iso-smcf-ArrMap-inv-smcf-ArrMap-app
is-iso-semifunctor.iso-inv-smcf-ObjMap-smcf-ObjMap-app
is-iso-semifunctor.iso-inv-smcf-ArrMap-smcf-ArrMap-app

4.4.13 An isomorphism of semicategories is an isomorphism in the category *SemiCAT*

lemma *is-iso-arr-is-iso-semifunctor*:
— See Chapter I-3 in [39].
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}$
and $\mathfrak{G} \circ_{SMCF} \mathfrak{F} = smcf\text{-}id\ \mathfrak{A}$
and $\mathfrak{F} \circ_{SMCF} \mathfrak{G} = smcf\text{-}id\ \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-iso-semifunctor-is-iso-arr*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
shows [*smcf-cs-intros*]: *inv-smcf* $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{A}$
and [*smcf-cs-simps*]: *inv-smcf* $\mathfrak{F} \circ_{SMCF} \mathfrak{F} = smcf\text{-}id\ \mathfrak{A}$
and [*smcf-cs-simps*]: $\mathfrak{F} \circ_{SMCF}$ *inv-smcf* $\mathfrak{F} = smcf\text{-}id\ \mathfrak{B}$
⟨*proof*⟩

An identity semifunctor is an isomorphism of semicategories

lemma (**in** *semicategory*) *smc-smcf-id-is-iso-semifunctor*:
smcf-id $\mathfrak{C} : \mathfrak{C} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-smcf-id-is-iso-semifunctor'*[*smcf-cs-intros*]:
assumes $\mathfrak{A}' = \mathfrak{C}$ **and** $\mathfrak{B}' = \mathfrak{C}$
shows *smcf-id* $\mathfrak{C} : \mathfrak{A}' \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*smcf-cs-intros*] = *semicategory.smc-smcf-id-is-iso-semifunctor'*

4.4.14 Isomorphic semicategories

Definition and elementary properties

See Chapter I-3 in [39]).

locale *iso-semicategory* = *L*: *semicategory* α $\mathfrak{A}$ + *R*: *semicategory* α $\mathfrak{B}$
for α $\mathfrak{A}$ $\mathfrak{B}$ +
assumes *iso-smc-is-iso-semifunctor*: $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$

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

Rules.

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

lemma *iso-semicategoryD*[*dest*]:

assumes $\mathfrak{A} \approx_{SMC\alpha} \mathfrak{B}$
shows $\exists \mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.iso\alpha} \mathfrak{B}$
⟨*proof*⟩

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

Elementary properties.

lemma (**in** *iso-semicategory*) *iso-smc-iso-digraph*: *smc-dg* $\mathfrak{A} \approx_{DG\alpha}$ *smc-dg* $\mathfrak{B}$
⟨*proof*⟩

A semicategory isomorphism is an equivalence relation

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

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

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

4.5 Smallness for semifunctors

4.5.1 Semifunctor with tiny maps

Definition and elementary properties

locale *is-tm-semifunctor* = *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ **for** α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  assumes *tm-smcf-is-tm-dghm*[*slicing-intros*]:
    smcf-dghm $\mathfrak{F}$: *smc-dg* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{DG.tm\alpha}$ *smc-dg* $\mathfrak{B}$

syntax *-is-tm-semifunctor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
  (‹(- :/ - $\mapsto\!\!\mapsto_{SMC.tm1}$ -)› [51, 51, 51] 51)
syntax-consts *-is-tm-semifunctor* $\rightleftharpoons$ *is-tm-semifunctor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\!\!\mapsto_{SMC.tm\alpha} \mathfrak{B} \rightleftharpoons$ *CONST is-tm-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

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

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

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

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

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

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

Rules.

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

mk-ide rf *is-tm-semifunctor-def*[*unfolded is-tm-semifunctor-axioms-def*]
  |*intro is-tm-semifunctorI*|
  |*dest is-tm-semifunctorD*[*dest*]|
  |*elim is-tm-semifunctorE*[*elim*]|

lemmas [*smc-small-cs-intros*] = *is-tm-semifunctorD*(1)

Slicing.

context *is-tm-semifunctor*
begin

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

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

tm-smcf-ObjMap-in-Vset[*smc-small-cs-intros*] = *dghm.tm-dghm-ObjMap-in-Vset*
and *tm-smcf-ArrMap-in-Vset*[*smc-small-cs-intros*] = *dghm.tm-dghm-ArrMap-in-Vset*

end

Elementary properties.

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

Further rules.

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

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

lemmas [*smc-small-cs-intros*] = *is-tm-semifunctorD'*(1)

lemma *is-tm-semifunctorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
obtains *semicategory* α $\mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
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-smcfs*[*simp*]: *small* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}\}$
⟨*proof*⟩

Opposite semifunctor with tiny maps

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

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

lemmas *is-tm-semifunctor-op*[*smc-op-intros*] = *is-tm-semifunctor.is-tm-semifunctor-op'*

Composition of semifunctors with tiny maps

lemma *smcf-comp-is-tm-semifunctor*[*smc-small-cs-intros*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$

⟨*proof*⟩

Finite semicategories and semifunctors with tiny maps

lemma (in *is-semifunctor*) *smcf-is-tm-semifunctor-if-HomDom-finite-semicategory*:
assumes *finite-semicategory* α $\mathfrak{A}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

Constant semifunctor with tiny maps

lemma *smcf-const-is-tm-semifunctor*:
assumes *tiny-semicategory* α $\mathfrak{C}$
and *semicategory* α $\mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} a$
and $f \circ_{A\mathfrak{D}} f = f$
shows *smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ a $f : \mathfrak{C} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{D}$
⟨*proof*⟩

lemma *smcf-const-is-tm-semifunctor′*:
assumes *tiny-semicategory* α $\mathfrak{C}$
and *semicategory* α $\mathfrak{D}$
and $f : a \mapsto_{\mathfrak{D}} a$
and $f \circ_{A\mathfrak{D}} f = f$
and $\mathfrak{C}' = \mathfrak{C}$
and $\mathfrak{D}' = \mathfrak{D}$
shows *smcf-const* $\mathfrak{C}$ $\mathfrak{D}$ a $f : \mathfrak{C}' \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{D}'$
⟨*proof*⟩

lemmas [*smc-small-cs-intros*] = *smcf-const-is-tm-semifunctor′*

4.5.2 Tiny semifunctor

Definition and elementary properties

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

syntax *-is-tiny-semifunctor* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $\mapsto\mapsto_{SMC.tiny1}$ -)› [51, 51, 51] 51)
syntax-consts *-is-tiny-semifunctor* ⇌ *is-tiny-semifunctor*
translations $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$ ⇌ *CONST is-tiny-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$

abbreviation (*input*) *is-cn-tiny-smcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
where *is-cn-tiny-smcf* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ ≡ $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha}$ $\mathfrak{B}$

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

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

abbreviation *tiny-smcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *tiny-smcfs* α $\mathfrak{A}$ $\mathfrak{B}$ ≡ *set* $\{\mathfrak{F}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

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

Rules.

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

mk-ide rf *is-tiny-semifunctor-def*[*unfolded is-tiny-semifunctor-axioms-def*]
|*intro is-tiny-semifunctorI*|
|*dest is-tiny-semifunctorD*[*dest*]|
|*elim is-tiny-semifunctorE*[*elim*]|

lemmas [*smc-small-cs-intros*] = *is-tiny-semifunctorD*(1)

Elementary properties.

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

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

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

Further rules.

lemma *is-tiny-semifunctorI'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *tiny-semicategory* α $\mathfrak{A}$
and *tiny-semicategory* α $\mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-semifunctorD'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *tiny-semicategory* α $\mathfrak{A}$
and *tiny-semicategory* α $\mathfrak{B}$
⟨*proof*⟩

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

lemma *is-tiny-semifunctorE'*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *tiny-semicategory* α $\mathfrak{A}$
and *tiny-semicategory* α $\mathfrak{B}$
⟨*proof*⟩

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

Size.

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

lemma *small-all-tiny-smcfs*[*simp*]: *small* $\{\mathfrak{F}.\ \exists \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

⟨*proof*⟩

lemma *tiny-smcfs-vsubset-Vset*[*simp*]:
set {𝔉. ∃𝔄 𝔅. 𝔉 : 𝔄 ↦↦$_{SMC.tiny\alpha}$ 𝔅} ⊆$_\circ$ *Vset* α
⟨*proof*⟩

lemma (**in** *is-semifunctor*) *smcf-is-tiny-semifunctor-if-ge-Limit*:
assumes 𝒵 β **and** α ∈$_\circ$ β
shows 𝔉 : 𝔄 ↦↦$_{SMC.tiny\beta}$ 𝔅
⟨*proof*⟩

Opposite tiny semifunctor

lemma (**in** *is-tiny-semifunctor*) *is-tiny-semifunctor-op*:
op-smcf 𝔉 : *op-smc* 𝔄 ↦↦$_{SMC.tiny\alpha}$ *op-smc* 𝔅
⟨*proof*⟩

lemma (**in** *is-tiny-semifunctor*) *is-tiny-semifunctor-op'*[*smc-op-intros*]:
assumes 𝔄' = *op-smc* 𝔄 **and** 𝔅' = *op-smc* 𝔅 **and** α' = α
shows *op-smcf* 𝔉 : 𝔄' ↦↦$_{SMC.tiny_{\alpha'}}$ 𝔅'
⟨*proof*⟩

lemmas *is-tiny-semifunctor-op*[*smc-op-intros*] =
is-tiny-semifunctor.is-tiny-semifunctor-op'

Composition of tiny semifunctors

lemma *smcf-comp-is-tiny-semifunctor*[*smc-small-cs-intros*]:
assumes 𝔊 : 𝔅 ↦↦$_{SMC.tiny\alpha}$ ℭ **and** 𝔉 : 𝔄 ↦↦$_{SMC.tiny\alpha}$ 𝔅
shows 𝔊 ∘$_{SMCF}$ 𝔉 : 𝔄 ↦↦$_{SMC.tiny\alpha}$ ℭ
⟨*proof*⟩

Tiny constant semifunctor

lemma *smcf-const-is-tiny-semifunctor*:
assumes *tiny-semicategory* α ℭ
and *tiny-semicategory* α 𝔇
and *f* : *a* ↦$_{\mathfrak{D}}$ *a*
and *f* ∘$_{A\mathfrak{D}}$ *f* = *f*
shows *smcf-const* ℭ 𝔇 *a f* : ℭ ↦↦$_{SMC.tiny\alpha}$ 𝔇
⟨*proof*⟩

lemma *smcf-const-is-tiny-semifunctor'*[*smc-small-cs-intros*]:
assumes *tiny-semicategory* α ℭ
and *tiny-semicategory* α 𝔇
and *f* : *a* ↦$_{\mathfrak{D}}$ *a*
and *f* ∘$_{A\mathfrak{D}}$ *f* = *f*
and ℭ' = ℭ
and 𝔇' = 𝔇
shows *smcf-const* ℭ 𝔇 *a f* : ℭ' ↦↦$_{SMC.tiny\alpha}$ 𝔇'
⟨*proof*⟩

4.6 Natural transformation of a semifunctor

4.6.1 Background

named-theorems *ntsmcf-cs-simps*
named-theorems *ntsmcf-cs-intros*

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

Slicing

definition *ntsmcf-tdghm* :: $V \Rightarrow V$
  where *ntsmcf-tdghm* $\mathfrak{N}$ =
    [
      $\mathfrak{N}(\lfloor NTMap \rfloor)$,
      smcf-dghm $(\mathfrak{N}(\lfloor NTDom \rfloor))$,
      smcf-dghm $(\mathfrak{N}(\lfloor NTCod \rfloor))$,
      smc-dg $(\mathfrak{N}(\lfloor NTDGDom \rfloor))$,
      smc-dg $(\mathfrak{N}(\lfloor NTDGCod \rfloor))$
    ]$_\circ$

Components.

lemma *ntsmcf-tdghm-components*:
  shows [*slicing-simps*]: *ntsmcf-tdghm* $\mathfrak{N}(\lfloor NTMap \rfloor)$ = $\mathfrak{N}(\lfloor NTMap \rfloor)$
    and [*slicing-commute*]: *ntsmcf-tdghm* $\mathfrak{N}(\lfloor NTDom \rfloor)$ = *smcf-dghm* $(\mathfrak{N}(\lfloor NTDom \rfloor))$
    and [*slicing-commute*]: *ntsmcf-tdghm* $\mathfrak{N}(\lfloor NTCod \rfloor)$ = *smcf-dghm* $(\mathfrak{N}(\lfloor NTCod \rfloor))$
    and [*slicing-commute*]: *ntsmcf-tdghm* $\mathfrak{N}(\lfloor NTDGDom \rfloor)$ = *smc-dg* $(\mathfrak{N}(\lfloor NTDGDom \rfloor))$
    and [*slicing-commute*]: *ntsmcf-tdghm* $\mathfrak{N}(\lfloor NTDGCod \rfloor)$ = *smc-dg* $(\mathfrak{N}(\lfloor NTDGCod \rfloor))$
  ⟨*proof*⟩

4.6.2 Definition and elementary properties

A natural transformation of semifunctors, as presented in this work, is a generalization of the concept of a natural transformation, as presented in Chapter I-4 in [39], to semicategories and semifunctors.

locale *is-ntsmcf* =
  $\mathcal{Z}$ α +
  vfsequence $\mathfrak{N}$ +
  NTDom: *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ +
  NTCod: *is-semifunctor* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{G}$
  for α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$ +
  assumes *ntsmcf-length*[*smc-cs-simps*]: *vcard* $\mathfrak{N}$ = $5_{\mathbb{N}}$
    and *ntsmcf-is-tdghm*[*slicing-intros*]: *ntsmcf-tdghm* $\mathfrak{N}$:
      smcf-dghm $\mathfrak{F}$ $\mapsto_{DGHM}$ *smcf-dghm* $\mathfrak{G}$: *smc-dg* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{DG\alpha}$ *smc-dg* $\mathfrak{B}$
    and *ntsmcf-NTDom*[*smc-cs-simps*]: $\mathfrak{N}(\lfloor NTDom \rfloor)$ = $\mathfrak{F}$
    and *ntsmcf-NTCod*[*smc-cs-simps*]: $\mathfrak{N}(\lfloor NTCod \rfloor)$ = $\mathfrak{G}$
    and *ntsmcf-NTDGDom*[*smc-cs-simps*]: $\mathfrak{N}(\lfloor NTDGDom \rfloor)$ = $\mathfrak{A}$
    and *ntsmcf-NTDGCod*[*smc-cs-simps*]: $\mathfrak{N}(\lfloor NTDGCod \rfloor)$ = $\mathfrak{B}$
    and *ntsmcf-Comp-commute*[*smc-cs-intros*]: $f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
      $\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$

syntax *-is-ntsmcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
  (‹(- :/ - $\mapsto_{SMCF}$ - :/ - $\mapsto\!\!\mapsto_{SMC^1}$ -)› [51, 51, 51, 51, 51] 51)
syntax-consts *-is-ntsmcf* ⇌ *is-ntsmcf*
translations $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{SMCF}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{B}$ ⇌
  CONST is-ntsmcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

abbreviation *all-ntsmcfs* :: $V \Rightarrow V$
  where *all-ntsmcfs* $\alpha \equiv$ *set* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}\ \mathfrak{A}\ \mathfrak{B}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}\}$

abbreviation *ntsmcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *ntsmcfs* $\alpha\ \mathfrak{A}\ \mathfrak{B} \equiv$ *set* $\{\mathfrak{N}.\ \exists \mathfrak{F}\ \mathfrak{G}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}\}$

abbreviation *these-ntsmcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
  where *these-ntsmcfs* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \equiv$ *set* $\{\mathfrak{N}.\ \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}\}$

lemmas [*smc-cs-simps*] =
  is-ntsmcf.ntsmcf-length
  is-ntsmcf.ntsmcf-NTDom
  is-ntsmcf.ntsmcf-NTCod
  is-ntsmcf.ntsmcf-NTDGDom
  is-ntsmcf.ntsmcf-NTDGCod
  is-ntsmcf.ntsmcf-Comp-commute

lemmas [*smc-cs-intros*] = *is-ntsmcf.ntsmcf-Comp-commute*

lemma (**in** *is-ntsmcf*) *ntsmcf-is-tdghm'*:
  assumes $\mathfrak{F}' =$ *smcf-dghm* $\mathfrak{F}$
    and $\mathfrak{G}' =$ *smcf-dghm* $\mathfrak{G}$
    and $\mathfrak{A}' =$ *smc-dg* $\mathfrak{A}$
    and $\mathfrak{B}' =$ *smc-dg* $\mathfrak{B}$
  shows *ntsmcf-tdghm* $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG\alpha} \mathfrak{B}'$
  ⟨*proof*⟩

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

Rules.

lemma (**in** *is-ntsmcf*) *is-ntsmcf-axioms'*[*smc-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 $\mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC_{\alpha'}} \mathfrak{B}'$
  ⟨*proof*⟩

mk-ide rf *is-ntsmcf-def*[*unfolded is-ntsmcf-axioms-def*]
  |*intro is-ntsmcfI*|
  |*dest is-ntsmcfD*[*dest*]|
  |*elim is-ntsmcfE*[*elim*]|

lemmas [*smc-cs-intros*] =
  is-ntsmcfD(3,4)

lemma *is-ntsmcfI'*:
  assumes $\mathcal{Z}\ \alpha$
    and *vfsequence* $\mathfrak{N}$
    and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
    and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
    and *vcard* $\mathfrak{N} = 5_{\mathbb{N}}$
    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 $\bigwedge a\ b\ f.\ f : a \mapsto_{\mathfrak{A}} b \Longrightarrow$
      $\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) = \mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$

shows $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-ntsmcfD'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{N}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *vcard* $\mathfrak{N} = 5_{\mathbb{N}}$
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-ntsmcfE'*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
obtains $\mathcal{Z}\ \alpha$
and *vfsequence* $\mathfrak{N}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and *vcard* $\mathfrak{N} = 5_{\mathbb{N}}$
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-ntsmcf*
begin

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

lemmas-with [*unfolded slicing-simps*]:
ntsmcf-NTMap-vsv = *tdghm.tdghm-NTMap-vsv*
and *ntsmcf-NTMap-vdomain*[*smc-cs-simps*] = *tdghm.tdghm-NTMap-vdomain*
and *ntsmcf-NTMap-is-arr* = *tdghm.tdghm-NTMap-is-arr*
and *ntsmcf-NTMap-is-arr'*[*smc-cs-intros*] = *tdghm.tdghm-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*]:

ntsmcf-NTMap-app-in-Arr[*smc-cs-intros*] = *tdghm.tdghm-NTMap-app-in-Arr*
and *ntsmcf-NTMap-vrange-vifunion* = *tdghm.tdghm-NTMap-vrange-vifunion*
and *ntsmcf-NTMap-vrange* = *tdghm.tdghm-NTMap-vrange*
and *ntsmcf-NTMap-vsubset-Vset* = *tdghm.tdghm-NTMap-vsubset-Vset*
and *ntsmcf-NTMap-in-Vset* = *tdghm.tdghm-NTMap-in-Vset*
and *ntsmcf-is-tdghm-if-ge-Limit* = *tdghm.tdghm-is-tdghm-if-ge-Limit*

end

lemmas [*smc-cs-intros*] = *is-ntsmcf.ntsmcf-NTMap-is-arr'*

lemma (in *is-ntsmcf*) *ntsmcf-Comp-commute'*:
assumes $f : a \mapsto_{\mathfrak{A}} b$ **and** $g : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
shows
$\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} (\mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} g) =$
$(\mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)) \circ_{A\mathfrak{B}} g$
⟨*proof*⟩

lemma (in *is-ntsmcf*) *ntsmcf-Comp-commute''*:
assumes $f : a \mapsto_{\mathfrak{A}} b$ **and** $g : c \mapsto_{\mathfrak{B}} \mathfrak{F}(\lfloor ObjMap \rfloor)(\lfloor a \rfloor)$
shows
$\mathfrak{G}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} (\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) \circ_{A\mathfrak{B}} g) =$
$(\mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor)) \circ_{A\mathfrak{B}} g$
⟨*proof*⟩

Elementary properties.

lemma *ntsmcf-eqI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
and $\mathfrak{N}(\lfloor NTMap \rfloor) = \mathfrak{N}'(\lfloor NTMap \rfloor)$
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 *ntsmcf-tdghm-eqI*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
and $\mathfrak{F} = \mathfrak{F}'$
and $\mathfrak{G} = \mathfrak{G}'$
and $\mathfrak{A} = \mathfrak{A}'$
and $\mathfrak{B} = \mathfrak{B}'$
and *ntsmcf-tdghm* $\mathfrak{N}$ = *ntsmcf-tdghm* $\mathfrak{N}'$
shows $\mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

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

Size.

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

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

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

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

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

Further elementary results.

lemma *these-ntsmcfs-iff*:
$\mathfrak{N} \in_{\circ}$ *these-ntsmcfs* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \longleftrightarrow \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

4.6.3 Opposite natural transformation of semifunctors

Definition and elementary properties

See section 1.5 in [15].

definition *op-ntsmcf* :: $V \Rightarrow V$
where *op-ntsmcf* $\mathfrak{N}$ =
[
$\mathfrak{N}(\lfloor NTMap \rfloor)$,
op-smcf $(\mathfrak{N}(\lfloor NTCod \rfloor))$,
op-smcf $(\mathfrak{N}(\lfloor NTDom \rfloor))$,
op-smc $(\mathfrak{N}(\lfloor NTDGDom \rfloor))$,
op-smc $(\mathfrak{N}(\lfloor NTDGCod \rfloor))$
]$_{\circ}$

Components.

lemma *op-ntsmcf-components*[*smc-op-simps*]:
shows *op-ntsmcf* $\mathfrak{N}(\lfloor NTMap \rfloor)$ = $\mathfrak{N}(\lfloor NTMap \rfloor)$
and *op-ntsmcf* $\mathfrak{N}(\lfloor NTDom \rfloor)$ = *op-smcf* $(\mathfrak{N}(\lfloor NTCod \rfloor))$
and *op-ntsmcf* $\mathfrak{N}(\lfloor NTCod \rfloor)$ = *op-smcf* $(\mathfrak{N}(\lfloor NTDom \rfloor))$
and *op-ntsmcf* $\mathfrak{N}(\lfloor NTDGDom \rfloor)$ = *op-smc* $(\mathfrak{N}(\lfloor NTDGDom \rfloor))$
and *op-ntsmcf* $\mathfrak{N}(\lfloor NTDGCod \rfloor)$ = *op-smc* $(\mathfrak{N}(\lfloor NTDGCod \rfloor))$
⟨*proof*⟩

Slicing.

lemma *op-tdghm-ntsmcf-tdghm*[*slicing-commute*]:
op-tdghm (*ntsmcf-tdghm* $\mathfrak{N}$) = *ntsmcf-tdghm* (*op-ntsmcf* $\mathfrak{N}$)
⟨*proof*⟩

Further properties

lemma (in *is-ntsmcf*) *is-ntsmcf-op*:
op-ntsmcf $\mathfrak{N}$: *op-smcf* $\mathfrak{G} \mapsto_{SMCF}$ *op-smcf* $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC\alpha}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

lemma (in *is-ntsmcf*) *is-ntsmcf-op'*[*smc-op-intros*]:
assumes $\mathfrak{G}'$ = *op-smcf* $\mathfrak{G}$

and $\mathfrak{F}' = op\text{-}smcf\ \mathfrak{F}$
and $\mathfrak{A}' = op\text{-}smc\ \mathfrak{A}$
and $\mathfrak{B}' = op\text{-}smc\ \mathfrak{B}$
shows $op\text{-}ntsmcf\ \mathfrak{N} : \mathfrak{G}' \mapsto_{SMCF} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*smc-op-intros*] = *is-ntsmcf.is-ntsmcf-op'*

lemma (in *is-ntsmcf*) *ntsmcf-op-ntsmcf-op-ntsmcf*[*smc-op-simps*]:
$op\text{-}ntsmcf\ (op\text{-}ntsmcf\ \mathfrak{N}) = \mathfrak{N}$
⟨*proof*⟩

lemmas *ntsmcf-op-ntsmcf-op-ntsmcf*[*smc-op-simps*] =
is-ntsmcf.ntsmcf-op-ntsmcf-op-ntsmcf

lemma *eq-op-ntsmcf-iff*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
shows $op\text{-}ntsmcf\ \mathfrak{N} = op\text{-}ntsmcf\ \mathfrak{N}' \longleftrightarrow \mathfrak{N} = \mathfrak{N}'$
⟨*proof*⟩

4.6.4 Vertical composition of natural transformations

Definition and elementary properties

See Chapter II-4 in [39].

definition $ntsmcf\text{-}vcomp :: V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\cdot_{NTSMCF}\rangle$ 55)
where *ntsmcf-vcomp* $\mathfrak{M}\ \mathfrak{N} =$
[
$(\lambda a \in_\circ \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!).\ (\mathfrak{M}(\!|NTMap|\!)(\!|a|\!)) \circ_{A\mathfrak{N}(\!|NTDGCod|\!)} (\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)))$,
$\mathfrak{N}(\!|NTDom|\!)$,
$\mathfrak{M}(\!|NTCod|\!)$,
$\mathfrak{N}(\!|NTDGDom|\!)$,
$\mathfrak{M}(\!|NTDGCod|\!)$
]$_\circ$

Components.

lemma *ntsmcf-vcomp-components*:
shows
$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTMap|\!) =$
$(\lambda a \in_\circ \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!).\ (\mathfrak{M}(\!|NTMap|\!)(\!|a|\!)) \circ_{A\mathfrak{N}(\!|NTDGCod|\!)} (\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)))$
and [*dg-shared-cs-simps*, *smc-cs-simps*]: $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTDom|\!) = \mathfrak{N}(\!|NTDom|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]: $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTCod|\!) = \mathfrak{M}(\!|NTCod|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTDGDom|\!) = \mathfrak{N}(\!|NTDGDom|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTDGCod|\!) = \mathfrak{M}(\!|NTDGCod|\!)$
⟨*proof*⟩

Natural transformation map

lemma *ntsmcf-vcomp-NTMap-vsv*[*dg-shared-cs-intros*, *smc-cs-intros*]:
vsv $((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\!|NTMap|\!))$
⟨*proof*⟩

lemma *ntsmcf-vcomp-NTMap-vdomain*[*smc-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$

shows $\mathcal{D}_{\circ}\ ((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)) = \mathfrak{A}(\lfloor Obj \rfloor)$
⟨*proof*⟩

lemma *ntsmcf-vcomp-NTMap-app*[*smc-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $a \in_{\circ} \mathfrak{A}(\lfloor Obj \rfloor)$
shows $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor a \rfloor) = \mathfrak{M}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
⟨*proof*⟩

lemma *ntsmcf-vcomp-NTMap-vrange*:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{R}_{\circ}\ ((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)) \subseteq_{\circ} \mathfrak{B}(\lfloor Arr \rfloor)$
⟨*proof*⟩

Further properties

lemma *ntsmcf-vcomp-composable-commute*[*smc-cs-simps*]:
— See Chapter II-4 in [39]).
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $f : a \mapsto_{\mathfrak{A}} b$
shows
$(\mathfrak{M}(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor b \rfloor)) \circ_{A\mathfrak{B}} \mathfrak{F}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) =$
$\mathfrak{H}(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{B}} (\mathfrak{M}(\lfloor NTMap \rfloor)(\lfloor a \rfloor) \circ_{A\mathfrak{B}} \mathfrak{N}(\lfloor NTMap \rfloor)(\lfloor a \rfloor))$
(**is** ‹(*?MC* $\circ_{A\mathfrak{B}}$ *?NC*) $\circ_{A\mathfrak{B}}$ *?R* = *?T* $\circ_{A\mathfrak{B}}$ (*?MD* $\circ_{A\mathfrak{B}}$ *?ND*)›)
⟨*proof*⟩

lemma *ntsmcf-vcomp-is-ntsmcf*[*smc-cs-intros*]:
— See Chapter II-4 in [39].
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *ntsmcf-vcomp-assoc*[*smc-cs-simps*]:
— See Chapter II-4 in [39].
assumes $\mathfrak{L} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $(\mathfrak{L} \cdot_{NTSMCF} \mathfrak{M}) \cdot_{NTSMCF} \mathfrak{N} = \mathfrak{L} \cdot_{NTSMCF} (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})$
⟨*proof*⟩

Opposite of the vertical composition of natural transformations of semifunctors

lemma *op-ntsmcf-ntsmcf-vcomp*[*smc-op-simps*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows *op-ntsmcf* $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})$ = *op-ntsmcf* $\mathfrak{N} \cdot_{NTSMCF}$ *op-ntsmcf* $\mathfrak{M}$
⟨*proof*⟩

4.6.5 Horizontal composition of natural transformations

Definition and elementary properties

See Chapter II-5 in [39].

definition *ntsmcf-hcomp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** ‹$\circ_{NTSMCF}$› 55)

where *ntsmcf-hcomp* $\mathfrak{M}$ $\mathfrak{N}$ =
[
(
$\lambda a \in_{\circ} \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!)$.
(
$\mathfrak{M}(\!|NTCod|\!)(\!|ArrMap|\!)(\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)) \circ_{A} \mathfrak{M}(\!|NTDGCod|\!)$
$\mathfrak{M}(\!|NTMap|\!)(\mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|a|\!))$
)
),
$(\mathfrak{M}(\!|NTDom|\!) \circ_{SMCF} \mathfrak{N}(\!|NTDom|\!))$,
$(\mathfrak{M}(\!|NTCod|\!) \circ_{SMCF} \mathfrak{N}(\!|NTCod|\!))$,
$(\mathfrak{N}(\!|NTDGDom|\!))$,
$(\mathfrak{M}(\!|NTDGCod|\!))$
]$_{\circ}$

Components.

lemma *ntsmcf-hcomp-components*:
shows
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTMap|\!)$ =
(
$\lambda a \in_{\circ} \mathfrak{N}(\!|NTDGDom|\!)(\!|Obj|\!)$.
(
$\mathfrak{M}(\!|NTCod|\!)(\!|ArrMap|\!)(\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)) \circ_{A} \mathfrak{M}(\!|NTDGCod|\!)$
$\mathfrak{M}(\!|NTMap|\!)(\mathfrak{N}(\!|NTDom|\!)(\!|ObjMap|\!)(\!|a|\!))$
)
)
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTDom|\!) = \mathfrak{M}(\!|NTDom|\!) \circ_{SMCF} \mathfrak{N}(\!|NTDom|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTCod|\!) = \mathfrak{M}(\!|NTCod|\!) \circ_{SMCF} \mathfrak{N}(\!|NTCod|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTDGDom|\!) = \mathfrak{N}(\!|NTDGDom|\!)$
and [*dg-shared-cs-simps*, *smc-cs-simps*]:
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTDGCod|\!) = \mathfrak{M}(\!|NTDGCod|\!)$
⟨*proof*⟩

Natural transformation map

lemma *ntsmcf-hcomp-NTMap-vsv*[*smc-cs-intros*]: *vsv* $((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTMap|\!))$
⟨*proof*⟩

lemma *ntsmcf-hcomp-NTMap-vdomain*[*smc-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{D}_{\circ}$ $((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTMap|\!)) = \mathfrak{A}(\!|Obj|\!)$
⟨*proof*⟩

lemma *ntsmcf-hcomp-NTMap-app*[*smc-cs-simps*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $a \in_{\circ} \mathfrak{A}(\!|Obj|\!)$
shows $(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTMap|\!)(\!|a|\!) =$
$\mathfrak{G}'(\!|ArrMap|\!)(\mathfrak{N}(\!|NTMap|\!)(\!|a|\!)) \circ_{A\mathfrak{C}} \mathfrak{M}(\!|NTMap|\!)(\mathfrak{F}(\!|ObjMap|\!)(\!|a|\!))$
⟨*proof*⟩

lemma *ntsmcf-hcomp-NTMap-vrange*:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{R}_{\circ}$ $((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\!|NTMap|\!)) \subseteq_{\circ} \mathfrak{C}(\!|Arr|\!)$

⟨*proof*⟩

Further properties

lemma *ntsmcf-hcomp-composable-commute*:
— See Chapter II-5 in [39].
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $f : a \mapsto_{\mathfrak{A}} b$
shows
$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor b \rfloor) \circ_{A\mathfrak{C}} (\mathfrak{F}' \circ_{SMCF} \mathfrak{F})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) =$
$(\mathfrak{G}' \circ_{SMCF} \mathfrak{G})(\lfloor ArrMap \rfloor)(\lfloor f \rfloor) \circ_{A\mathfrak{C}} (\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)(\lfloor a \rfloor)$
⟨*proof*⟩

lemma *ntsmcf-hcomp-is-ntsmcf*:
— See Chapter II-5 in [39].
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{M} \circ_{NTSMCF} \mathfrak{N} : \mathfrak{F}' \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF} \mathfrak{G}' \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntsmcf-hcomp-is-ntsmcf'*[*smc-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{S} = \mathfrak{F}' \circ_{SMCF} \mathfrak{F}$
and $\mathfrak{S}' = \mathfrak{G}' \circ_{SMCF} \mathfrak{G}$
shows $\mathfrak{M} \circ_{NTSMCF} \mathfrak{N} : \mathfrak{S} \mapsto_{SMCF} \mathfrak{S}' : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntsmcf-hcomp-assoc*[*smc-cs-simps*]:
— See Chapter II-5 in [39].
assumes $\mathfrak{L} : \mathfrak{F}'' \mapsto_{SMCF} \mathfrak{G}'' : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
and $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $(\mathfrak{L} \circ_{NTSMCF} \mathfrak{M}) \circ_{NTSMCF} \mathfrak{N} = \mathfrak{L} \circ_{NTSMCF} (\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})$
⟨*proof*⟩

Opposite of the horizontal composition of the natural transformation of semifunctors

lemma *op-ntsmcf-ntsmcf-hcomp*[*smc-op-simps*]:
assumes $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows *op-ntsmcf* $(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N}) =$ *op-ntsmcf* $\mathfrak{M} \circ_{NTSMCF}$ *op-ntsmcf* $\mathfrak{N}$
⟨*proof*⟩

4.6.6 Interchange law

lemma *ntsmcf-comp-interchange-law*:
— See Chapter II-5 in [39].
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{M}' : \mathfrak{G}' \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
shows
$((\mathfrak{M}' \cdot_{NTSMCF} \mathfrak{N}') \circ_{NTSMCF} (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})) =$
$(\mathfrak{M}' \circ_{NTSMCF} \mathfrak{M}) \cdot_{NTSMCF} (\mathfrak{N}' \circ_{NTSMCF} \mathfrak{N})$
⟨*proof*⟩

4.6.7 Composition of a natural transformation of semifunctors and a semifunctor

Definition and elementary properties

abbreviation (*input*) *ntsmcf-smcf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle \circ_{NTSMCF\text{-}SMCF} \rangle$ 55)
where *ntsmcf-smcf-comp* ≡ *tdghm-dghm-comp*

Slicing.

lemma *ntsmcf-tdghm-ntsmcf-smcf-comp*[*slicing-commute*]:
ntsmcf-tdghm $\mathfrak{N} \circ_{TDGHM\text{-}DGHM}$ *smcf-dghm* $\mathfrak{H}$ = *ntsmcf-tdghm* $(\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H})$
⟨*proof*⟩

Natural transformation map

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

lemmas [*smc-cs-simps*] =
is-semifunctor.ntsmcf-smcf-comp-NTMap-vdomain
is-semifunctor.ntsmcf-smcf-comp-NTMap-app

lemma *ntsmcf-smcf-comp-NTMap-vrange*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ\ ((\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

Opposite of the composition of a natural transformation of semifunctors and a semifunctor

lemma *op-ntsmcf-ntsmcf-smcf-comp*[*smc-op-simps*]:
op-ntsmcf $(\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H})$ = *op-ntsmcf* $\mathfrak{N} \circ_{NTSMCF\text{-}SMCF}$ *op-smcf* $\mathfrak{H}$
⟨*proof*⟩

Composition of a natural transformation of semifunctors and a semifunctors is a natural transformation of semifunctors

lemma *ntsmcf-smcf-comp-is-ntsmcf*[*intro*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H} : \mathfrak{F} \circ_{SMCF} \mathfrak{H} \mapsto_{SMCF} \mathfrak{G} \circ_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntsmcf-smcf-comp-is-semifunctor'*[*smc-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{SMCF} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{SMCF} \mathfrak{H}$
shows $\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma *ntsmcf-smcf-comp-ntsmcf-smcf-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
and $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $(\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{G}) \circ_{NTSMCF\text{-}SMCF} \mathfrak{F} = \mathfrak{N} \circ_{NTSMCF\text{-}SMCF} (\mathfrak{G} \circ_{SMCF} \mathfrak{F})$

⟨*proof*⟩

lemma (**in** *is-ntsmcf*) *ntsmcf-ntsmcf-smcf-comp-smcf-id*[*smc-cs-simps*]:
$\mathfrak{N} \circ_{NTSMCF-SMCF} \textit{smcf-id}\ \mathfrak{A} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*smc-cs-simps*] = *is-ntsmcf.ntsmcf-ntsmcf-smcf-comp-smcf-id*

lemma *ntsmcf-vcomp-ntsmcf-smcf-comp*[*smc-cs-simps*]:
assumes $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
shows
$(\mathfrak{M} \circ_{NTSMCF-SMCF} \mathfrak{K}) \cdot_{NTSMCF} (\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{K}) =$
$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N}) \circ_{NTSMCF-SMCF} \mathfrak{K}$
⟨*proof*⟩

4.6.8 Composition of a semifunctor and a natural transformation of semifunctors

Definition and elementary properties

abbreviation (*input*) *smcf-ntsmcf-comp* :: $V \Rightarrow V \Rightarrow V$ (**infixl** $\langle\circ_{SMCF-NTSMCF}\rangle$ 55)
where *smcf-ntsmcf-comp* ≡ *dghm-tdghm-comp*

Slicing.

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

Natural transformation map

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

lemmas [*smc-cs-simps*] =
is-ntsmcf.smcf-ntsmcf-comp-NTMap-vdomain
is-ntsmcf.smcf-ntsmcf-comp-NTMap-app

lemma *smcf-ntsmcf-comp-NTMap-vrange*:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathcal{R}_\circ\ ((\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N})(\lfloor NTMap \rfloor)) \subseteq_\circ \mathfrak{C}(\lfloor Arr \rfloor)$
⟨*proof*⟩

Opposite of the composition of a semifunctor and a natural transformation of semifunctors

lemma *op-ntsmcf-smcf-ntsmcf-comp*[*smc-op-simps*]:
op-ntsmcf $(\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N})$ = *op-smcf* $\mathfrak{H} \circ_{SMCF-NTSMCF}$ *op-ntsmcf* $\mathfrak{N}$
⟨*proof*⟩

Composition of a semifunctor and a natural transformation of semifunctors is a natural transformation of semifunctors

lemma *smcf-ntsmcf-comp-is-ntsmcf*[*intro*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$

shows $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{H} \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF} \mathfrak{H} \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *smcf-ntsmcf-comp-is-semifunctor'*[*smc-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{H} \circ_{SMCF} \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{H} \circ_{SMCF} \mathfrak{G}$
shows $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma *smcf-comp-smcf-ntsmcf-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{G} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
shows $(\mathfrak{G} \circ_{SMCF} \mathfrak{F}) \circ_{SMCF-NTSMCF} \mathfrak{N} = \mathfrak{G} \circ_{SMCF-NTSMCF} (\mathfrak{F} \circ_{SMCF-NTSMCF} \mathfrak{N})$
⟨*proof*⟩

lemma (**in** *is-ntsmcf*) *ntsmcf-smcf-ntsmcf-comp-smcf-id*[*smc-cs-simps*]:
smcf-id $\mathfrak{B} \circ_{SMCF-NTSMCF} \mathfrak{N} = \mathfrak{N}$
⟨*proof*⟩

lemmas [*smc-cs-simps*] = *is-ntsmcf.ntsmcf-smcf-ntsmcf-comp-smcf-id*

lemma *smcf-ntsmcf-comp-ntsmcf-smcf-comp-assoc*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
and $\mathfrak{K} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $(\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N}) \circ_{NTSMCF-SMCF} \mathfrak{K} = \mathfrak{H} \circ_{SMCF-NTSMCF} (\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{K})$
⟨*proof*⟩

lemma *smcf-ntsmcf-comp-ntsmcf-vcomp*:
assumes $\mathfrak{K} : \mathfrak{B} \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
and $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows
$\mathfrak{K} \circ_{SMCF-NTSMCF} (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N}) =$
$(\mathfrak{K} \circ_{SMCF-NTSMCF} \mathfrak{M}) \cdot_{NTSMCF} (\mathfrak{K} \circ_{SMCF-NTSMCF} \mathfrak{N})$
⟨*proof*⟩

4.7 Smallness for natural transformations of semifunctors

4.7.1 Natural transformation of semifunctors with tiny maps

Definition and elementary properties

locale *is-tm-ntsmcf* = *is-ntsmcf* 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-ntsmcf-is-tm-tdghm*[*slicing-intros*]: *ntsmcf-tdghm* $\mathfrak{N}$:
smcf-dghm $\mathfrak{F}$ $\mapsto_{DGHM.tm}$ *smcf-dghm* $\mathfrak{G}$: *smc-dg* $\mathfrak{A}$ $\mapsto\!\!\mapsto_{DG.tm\alpha}$ *smc-dg* $\mathfrak{B}$

syntax *-is-tm-ntsmcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow$ *bool*
(‹(- :/ - $\mapsto_{SMCF.tm}$ - :/ - $\mapsto\!\!\mapsto_{SMC.tm1}$ -)› [51, 51, 51, 51, 51] 51)
syntax-consts *-is-tm-ntsmcf* $\rightleftharpoons$ *is-tm-ntsmcf*
translations $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}$ $\rightleftharpoons$
CONST is-tm-ntsmcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

abbreviation *all-tm-ntsmcfs* :: $V \Rightarrow V$
where *all-tm-ntsmcfs* $\alpha \equiv$
set {$\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{A}$ $\mathfrak{B}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}$}

abbreviation *tm-ntsmcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
where *tm-ntsmcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\equiv$
set {$\mathfrak{N}$. $\exists \mathfrak{F}$ $\mathfrak{G}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}$}

abbreviation *these-tm-ntsmcfs* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$
where *these-tm-ntsmcfs* α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\equiv$
set {$\mathfrak{N}$. $\mathfrak{N}$: $\mathfrak{F}$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}$: $\mathfrak{A}$ $\mapsto\!\!\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}$}

lemma (**in** *is-tm-ntsmcf*) *tm-ntsmcf-is-tm-tdghm'*:
assumes α' = α
and $\mathfrak{F}'$ = *smcf-dghm* $\mathfrak{F}$
and $\mathfrak{G}'$ = *smcf-dghm* $\mathfrak{G}$
and $\mathfrak{A}'$ = *smc-dg* $\mathfrak{A}$
and $\mathfrak{B}'$ = *smc-dg* $\mathfrak{B}$
shows *ntsmcf-tdghm* $\mathfrak{N}$:
$\mathfrak{F}'$ $\mapsto_{DGHM.tm}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{DG.tm_{\alpha'}}$ $\mathfrak{B}'$
⟨*proof*⟩

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

Rules.

lemma (**in** *is-tm-ntsmcf*) *is-tm-ntsmcf-axioms'*[*smc-small-cs-intros*]:
assumes α' = α **and** $\mathfrak{A}'$ = $\mathfrak{A}$ **and** $\mathfrak{B}'$ = $\mathfrak{B}$ **and** $\mathfrak{F}'$ = $\mathfrak{F}$ **and** $\mathfrak{G}'$ = $\mathfrak{G}$
shows $\mathfrak{N}$: $\mathfrak{F}'$ $\mapsto_{SMCF.tm}$ $\mathfrak{G}'$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{SMC.tm\alpha}$ $\mathfrak{B}'$
⟨*proof*⟩

mk-ide rf *is-tm-ntsmcf-def*[*unfolded is-tm-ntsmcf-axioms-def*]
|*intro is-tm-ntsmcfI*|
|*dest is-tm-ntsmcfD*[*dest*]|
|*elim is-tm-ntsmcfE*[*elim*]|

lemmas [*smc-small-cs-intros*] = *is-tm-ntsmcfD*(1)

Slicing.

context *is-tm-ntsmcf*
begin

interpretation *tdghm*: *is-tm-tdghm*
α ‹*smc-dg* $\mathfrak{A}$› ‹*smc-dg* $\mathfrak{B}$› ‹*smcf-dghm* $\mathfrak{F}$› ‹*smcf-dghm* $\mathfrak{G}$› ‹*ntsmcf-tdghm* $\mathfrak{N}$›

⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
tm-ntsmcf-NTMap-in-Vset = *tdghm.tm-tdghm-NTMap-in-Vset*

end

Elementary properties.

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

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

Further rules.

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

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

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

Size.

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

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

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

Further elementary results.

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

Opposite natural transformation of semifunctors with tiny maps

lemma (**in** *is-tm-ntsmcf*) *is-tm-ntsmcf-op*: *op-ntsmcf* $\mathfrak{N}$:
op-smcf $\mathfrak{G} \mapsto\mapsto_{SMCF.tm}$ *op-smcf* $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

lemma (**in** *is-tm-ntsmcf*) *is-tm-ntsmcf-op'*[*smc-op-intros*]:

assumes $\mathfrak{G}' = op\text{-}smcf\ \mathfrak{G}$
and $\mathfrak{F}' = op\text{-}smcf\ \mathfrak{F}$
and $\mathfrak{A}' = op\text{-}smc\ \mathfrak{A}$
and $\mathfrak{B}' = op\text{-}smc\ \mathfrak{B}$
shows $op\text{-}ntsmcf\ \mathfrak{N} : \mathfrak{G}' \mapsto_{SMCF.tm} \mathfrak{F}' : \mathfrak{A}' \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas *is-tm-ntsmcf-op*[*smc-op-intros*] = *is-tm-ntsmcf.is-tm-ntsmcf-op'*

Vertical composition of natural transformations of semifunctors with tiny maps

lemma *ntsmcf-vcomp-is-tm-ntsmcf*[*smc-small-cs-intros*]:
assumes $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF.tm} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
shows $\mathfrak{M} \bullet_{NTSMCF} \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
⟨*proof*⟩

Composition of a natural transformation of semifunctors and a semifunctor

lemma *ntsmcf-smcf-comp-is-tm-ntsmcf*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
shows $\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H} : \mathfrak{F} \circ_{SMCF} \mathfrak{H} \mapsto_{SMCF.tm} \mathfrak{G} \circ_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *ntsmcf-smcf-comp-is-tm-ntsmcf'*[*smc-small-cs-intros*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
and $\mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{F} \circ_{SMCF} \mathfrak{H}$
and $\mathfrak{G}' = \mathfrak{G} \circ_{SMCF} \mathfrak{H}$
shows $\mathfrak{N} \circ_{NTSMCF\text{-}SMCF} \mathfrak{H} : \mathfrak{F}' \mapsto_{SMCF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

Composition of a semifunctor and a natural transformation of semifunctors

lemma *smcf-ntsmcf-comp-is-tm-ntsmcf*:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$ **and** $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
shows $\mathfrak{H} \circ_{SMCF\text{-}NTSMCF} \mathfrak{N} : \mathfrak{H} \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{H} \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *smcf-ntsmcf-comp-is-tm-ntsmcf'*[*smc-small-cs-intros*]:
assumes $\mathfrak{H} : \mathfrak{B} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
and $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}$
and $\mathfrak{F}' = \mathfrak{H} \circ_{SMCF} \mathfrak{F}$
and $\mathfrak{G}' = \mathfrak{H} \circ_{SMCF} \mathfrak{G}$
shows $\mathfrak{H} \circ_{SMCF\text{-}NTSMCF} \mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{C}$
⟨*proof*⟩

4.7.2 Tiny natural transformation of semifunctors

Definition and elementary properties

locale *is-tiny-ntsmcf* = *is-ntsmcf* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$ **for** $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G}\ \mathfrak{N}$ +
assumes *tiny-ntsmcf-is-tdghm*[*slicing-intros*]: *ntsmcf-tdghm* $\mathfrak{N}$:
smcf-dghm $\mathfrak{F} \mapsto_{DGHM.tiny}$ *smcf-dghm* $\mathfrak{G}$: *smc-dg* $\mathfrak{A} \mapsto\mapsto_{DG.tiny\alpha}$ *smc-dg* $\mathfrak{B}$

syntax *-is-tiny-ntsmcf* :: $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$
(‹(- :/ - ↦$_{SMCF.tiny}$ - :/ - ↦↦$_{SMC.tiny1}$ -)› [51, 51, 51, 51, 51] 51)
syntax-consts *-is-tiny-ntsmcf* ⇌ *is-tiny-ntsmcf*
translations $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$ ⇌

CONST is-tiny-ntsmcf α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{F}$ $\mathfrak{G}$ $\mathfrak{N}$

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

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

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

lemma (in *is-tiny-ntsmcf*) *tiny-ntsmcf-is-tdghm′*:
assumes $\alpha' = \alpha$
and $\mathfrak{F}' =$ *smcf-dghm* $\mathfrak{F}$
and $\mathfrak{G}' =$ *smcf-dghm* $\mathfrak{G}$
and $\mathfrak{A}' =$ *smc-dg* $\mathfrak{A}$
and $\mathfrak{B}' =$ *smc-dg* $\mathfrak{B}$
shows *ntsmcf-tdghm* $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tiny} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG.tiny_{\alpha'}} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*slicing-intros*] = *is-tiny-ntsmcf.tiny-ntsmcf-is-tdghm′*

Rules.

lemma (in *is-tiny-ntsmcf*) *is-tiny-ntsmcf-axioms′*[*smc-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 $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

mk-ide rf *is-tiny-ntsmcf-def*[*unfolded is-tiny-ntsmcf-axioms-def*]
|*intro is-tiny-ntsmcfI*|
|*dest is-tiny-ntsmcfD*[*dest*]|
|*elim is-tiny-ntsmcfE*[*elim*]|

Elementary properties.

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

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

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

Further rules.

lemma *is-tiny-ntsmcfI′*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *is-tiny-ntsmcfD′*:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
shows $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$

and $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
and $\mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

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

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

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

Size.

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

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

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

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

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

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

Further elementary results.

lemma *these-tiny-ntsmcfs-iff*:
$\mathfrak{N} \in_{\circ}$ *these-tiny-ntsmcfs* $\alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \longleftrightarrow$
$\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha} \mathfrak{B}$
⟨*proof*⟩

Opposite natural transformation of tiny semifunctors

lemma (**in** *is-tiny-ntsmcf*) *is-tm-ntsmcf-op*: *op-ntsmcf* $\mathfrak{N}$:
op-smcf $\mathfrak{G} \mapsto_{SMCF.tiny}$ *op-smcf* $\mathfrak{F}$: *op-smc* $\mathfrak{A} \mapsto\mapsto_{SMC.tiny\alpha}$ *op-smc* $\mathfrak{B}$

⟨*proof*⟩

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

lemmas *is-tiny-ntsmcf-op*[*smc-op-intros*] = *is-tiny-ntsmcf.is-tiny-ntsmcf-op'*

Vertical composition of tiny natural transformations of semifunctors

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

4.8 Product semicategory

4.8.1 Background

The concept of a product semicategory, as presented in this work, is a generalization of the concept of a product category, as presented in Chapter II-3 in [39].

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

4.8.2 Product semicategory: definition and elementary properties

definition *smc-prod* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V$
  where *smc-prod* I $\mathfrak{A}$ =
    [
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$,
      $(\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!))(\!|f(\!|i|\!)|\!)))$,
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!))(\!|f(\!|i|\!)|\!)))$,
      $(\lambda gf \in_{\circ} composable\text{-}arrs\ (dg\text{-}prod\ I\ \mathfrak{A}).\ (\lambda i \in_{\circ} I.\ gf(\!|0|\!)(\!|i|\!) \circ_{A\mathfrak{A}\ i}\ gf(\!|1_{\mathbb{N}}|\!)(\!|i|\!)))$
    ]$_{\circ}$

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

Components.

lemma *smc-prod-components*:
  shows $(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Obj|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$
    and $(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Arr|\!) = (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$
    and $(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Dom|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Dom|\!))(\!|f(\!|i|\!)|\!)))$
    and $(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Cod|\!) =$
      $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ (\lambda i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Cod|\!))(\!|f(\!|i|\!)|\!)))$
    and $(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Comp|\!) =$
      $(\lambda gf \in_{\circ} composable\text{-}arrs\ (dg\text{-}prod\ I\ \mathfrak{A}).\ (\lambda i \in_{\circ} I.\ gf(\!|0|\!)(\!|i|\!) \circ_{A\mathfrak{A}\ i}\ gf(\!|1_{\mathbb{N}}|\!)(\!|i|\!)))$
  $\langle$*proof*$\rangle$

Slicing.

lemma *smc-dg-smc-prod*[*slicing-commute*]:
  dg-prod I $(\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i)) = smc\text{-}dg\ (smc\text{-}prod\ I\ \mathfrak{A})$
  $\langle$*proof*$\rangle$

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

lemmas-with
  [**where** $\mathfrak{A}$=$\langle\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i)\rangle$, *unfolded slicing-simps slicing-commute*]:
  smc-prod-ObjI = *dg-prod-ObjI*
  and *smc-prod-ObjD* = *dg-prod-ObjD*
  and *smc-prod-ObjE* = *dg-prod-ObjE*
  and *smc-prod-Obj-cong* = *dg-prod-Obj-cong*
  and *smc-prod-ArrI* = *dg-prod-ArrI*
  and *smc-prod-ArrD* = *dg-prod-ArrD*
  and *smc-prod-ArrE* = *dg-prod-ArrE*

and *smc-prod-Arr-cong* = *dg-prod-Arr-cong*
and *smc-prod-Dom-vsv*[*smc-cs-intros*] = *dg-prod-Dom-vsv*
and *smc-prod-Dom-vdomain*[*smc-cs-simps*] = *dg-prod-Dom-vdomain*
and *smc-prod-Dom-app* = *dg-prod-Dom-app*
and *smc-prod-Dom-app-component-app*[*smc-cs-simps*] =
dg-prod-Dom-app-component-app
and *smc-prod-Cod-vsv*[*smc-cs-intros*] = *dg-prod-Cod-vsv*
and *smc-prod-Cod-app* = *dg-prod-Cod-app*
and *smc-prod-Cod-vdomain*[*smc-cs-simps*] = *dg-prod-Cod-vdomain*
and *smc-prod-Cod-app-component-app*[*smc-cs-simps*] =
dg-prod-Cod-app-component-app
and *smc-prod-vunion-Obj-in-Obj* = *dg-prod-vunion-Obj-in-Obj*
and *smc-prod-vdiff-vunion-Obj-in-Obj* = *dg-prod-vdiff-vunion-Obj-in-Obj*
and *smc-prod-vunion-Arr-in-Arr* = *dg-prod-vunion-Arr-in-Arr*
and *smc-prod-vdiff-vunion-Arr-in-Arr* = *dg-prod-vdiff-vunion-Arr-in-Arr*

end

lemma *smc-prod-dg-prod-is-arr*:
$g : b \mapsto_{\prod_{DG} i \in_\circ I.\ \mathfrak{A}\ i}\ c \longleftrightarrow g : b \mapsto_{\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i}\ c$
⟨*proof*⟩

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

4.8.3 Local assumptions for a product semicategory

locale *psemicategory-base* = $\mathcal{Z}\ \alpha$ **for** $\alpha\ I\ \mathfrak{A}$ +
assumes *psmc-semicategories*[*smc-prod-cs-intros*]:
$i \in_\circ I \Longrightarrow$ *semicategory* $\alpha\ (\mathfrak{A}\ i)$
and *psmc-index-in-Vset*[*smc-cs-intros*]: $I \in_\circ$ *Vset* α

Rules.

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

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

lemma *psemicategory-base-pdigraph-baseI*:
assumes *pdigraph-base* $\alpha\ I\ (\lambda i.\ \textit{smc-dg}\ (\mathfrak{A}\ i))$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow$ *semicategory* $\alpha\ (\mathfrak{A}\ i)$
shows *psemicategory-base* $\alpha\ I\ \mathfrak{A}$
⟨*proof*⟩

Product semicategory is a product digraph.

context *psemicategory-base*
begin

lemma *psmc-pdigraph-base*: *pdigraph-base* $\alpha\ I\ (\lambda i.\ \textit{smc-dg}\ (\mathfrak{A}\ i))$
⟨*proof*⟩

interpretation *pdg*: *pdigraph-base* $\alpha\ I\ ‹\lambda i.\ \textit{smc-dg}\ (\mathfrak{A}\ i)›$

⟨*proof*⟩

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

end

lemmas [*smc-cs-intros*] = *psemicategory-base.smc-prod-is-arrD*(7)

Elementary properties.

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

Composition

lemma *smc-prod-Comp*:
$(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Comp|\!) =$
(
$\lambda gf \in_{\circ} composable\text{-}arrs\ (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i).$
$(\lambda i \in_{\circ} I.\ gf(\!|0|\!)(\!|i|\!) \circ_{A \mathfrak{A}\ i}\ gf(\!|1_{\mathbb{N}}|\!)(\!|i|\!))$
)
⟨*proof*⟩

lemma *smc-prod-Comp-vdomain*[*smc-cs-simps*]:
$\mathcal{D}_{\circ}\ ((\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Comp|\!)) = composable\text{-}arrs\ (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

lemma *smc-prod-Comp-app*:
assumes $g : b \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ c$ **and** $f : a \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ b$
shows $g \circ_{A (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)}\ f = (\lambda i \in_{\circ} I.\ g(\!|i|\!) \circ_{A \mathfrak{A}\ i}\ f(\!|i|\!))$
⟨*proof*⟩

lemma *smc-prod-Comp-app-component*[*smc-cs-simps*]:
assumes $g : b \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ c$
and $f : a \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ b$
and $i \in_{\circ} I$
shows $(g \circ_{A (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)}\ f)(\!|i|\!) = g(\!|i|\!) \circ_{A \mathfrak{A}\ i}\ f(\!|i|\!)$
⟨*proof*⟩

lemma (**in** *psemicategory-base*) *smc-prod-Comp-vrange*:
$\mathcal{R}_{\circ}\ ((\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Comp|\!)) \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *smc-prod-Comp-app-vdomain*[*smc-cs-simps*]:
assumes $g : b \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ c$ **and** $f : a \mapsto_{\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i}\ b$
shows $\mathcal{D}_{\circ}\ (g \circ_{A (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)}\ f) = I$
⟨*proof*⟩

A product α-semicategory is a tiny β-semicategory

lemma (in *psemicategory-base*) *psmc-tiny-semicategory-smc-prod*:
assumes $\mathcal{Z}\ \beta$ **and** $\alpha \in_\circ \beta$
shows *tiny-semicategory* β $(\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

4.8.4 Further local assumptions for product semicategories

Definition and elementary properties

locale *psemicategory* = *psemicategory-base* $\alpha\ I\ \mathfrak{A}$ **for** $\alpha\ I\ \mathfrak{A}$ +
assumes *psmc-Obj-vsubset-Vset*:
$J \subseteq_\circ I \Longrightarrow (\prod_{SMC} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!) \subseteq_\circ Vset\ \alpha$
and *psmc-Hom-vifunion-in-Vset*:
$[\![$
$J \subseteq_\circ I$;
$A \subseteq_\circ (\prod_{SMC} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$B \subseteq_\circ (\prod_{SMC} i \in_\circ J.\ \mathfrak{A}\ i)(\!|Obj|\!)$;
$A \in_\circ Vset\ \alpha$;
$B \in_\circ Vset\ \alpha$
$]\!] \Longrightarrow (\bigcup_\circ a \in_\circ A.\ \bigcup_\circ b \in_\circ B.\ Hom\ (\prod_{SMC} i \in_\circ J.\ \mathfrak{A}\ i)\ a\ b) \in_\circ Vset\ \alpha$

Rules.

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

mk-ide rf *psemicategory-def*[*unfolded psemicategory-axioms-def*]
|*intro psemicategoryI*|
|*dest psemicategoryD*[*dest*]|
|*elim psemicategoryE*[*elim*]|

lemmas [*smc-prod-cs-intros*] = *psemicategoryD*(1)

lemma *psemicategory-pdigraphI*:
assumes *pdigraph* $\alpha\ I\ (\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i))$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow semicategory\ \alpha\ (\mathfrak{A}\ i)$
shows *psemicategory* $\alpha\ I\ \mathfrak{A}$
⟨*proof*⟩

Product semicategory is a product digraph.

context *psemicategory*
begin

lemma *psmc-pdigraph*: *pdigraph* $\alpha\ I\ (\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i))$
⟨*proof*⟩

interpretation *pdg*: *pdigraph* $\alpha\ I\ \langle\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i)\rangle$ ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
psmc-Obj-vsubset-Vset' = *pdg.pdg-Obj-vsubset-Vset'*
and *psmc-Hom-vifunion-in-Vset'* = *pdg.pdg-Hom-vifunion-in-Vset'*
and *psmc-smc-prod-vunion-is-arr* = *pdg.pdg-dg-prod-vunion-is-arr*
and *psmc-smc-prod-vdiff-vunion-is-arr* = *pdg.pdg-dg-prod-vdiff-vunion-is-arr*

end

Elementary properties.

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

A product α-semicategory is an α-semicategory

lemma (in *psemicategory*) *psmc-semicategory-smc-prod*:
semicategory α $(\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

4.8.5 Local assumptions for a finite product semicategory

Definition and elementary properties

locale *finite-psemicategory* = *psemicategory-base* α I $\mathfrak{A}$ **for** α I $\mathfrak{A}$ +
assumes *fin-psmc-index-vfinite*: *vfinite* I

Rules.

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

mk-ide rf *finite-psemicategory-def*[*unfolded finite-psemicategory-axioms-def*]
|*intro finite-psemicategoryI*|
|*dest finite-psemicategoryD*[*dest*]|
|*elim finite-psemicategoryE*[*elim*]|

lemmas [*smc-prod-cs-intros*] = *finite-psemicategoryD*(1)

lemma *finite-psemicategory-finite-pdigraphI*:
assumes *finite-pdigraph* α I $(\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i))$
and $\bigwedge i.\ i \in_\circ I \Longrightarrow$ *semicategory* α $(\mathfrak{A}\ i)$
shows *finite-psemicategory* α I $\mathfrak{A}$
⟨*proof*⟩

Local assumptions for a finite product semicategory and local assumptions for an arbitrary product semicategory

sublocale *finite-psemicategory* ⊆ *psemicategory* α I $\mathfrak{A}$
⟨*proof*⟩

4.8.6 Binary union and complement

lemma (in *psemicategory*) *psmc-smc-prod-vunion-Comp*:
assumes *vdisjnt* J K
and $J \subseteq_\circ I$
and $K \subseteq_\circ I$
and $g : b \mapsto_{(\prod_{SMC} j \in_\circ J.\ \mathfrak{A}\ j)} c$
and $g' : b' \mapsto_{(\prod_{SMC} k \in_\circ K.\ \mathfrak{A}\ k)} c'$
and $f : a \mapsto_{(\prod_{SMC} j \in_\circ J.\ \mathfrak{A}\ j)} b$
and $f' : a' \mapsto_{(\prod_{SMC} k \in_\circ K.\ \mathfrak{A}\ k)} b'$
shows $(g \circ_{A(\prod_{SMC} j \in_\circ J.\ \mathfrak{A}\ j)} f) \cup_\circ (g' \circ_{A(\prod_{SMC} j \in_\circ K.\ \mathfrak{A}\ j)} f') =$
$g \cup_\circ g' \circ_{A(\prod_{SMC} j \in_\circ J \cup_\circ K.\ \mathfrak{A}\ j)} f \cup_\circ f'$
⟨*proof*⟩

lemma (in *psemicategory***)** *psmc-smc-prod-vdiff-vunion-Comp*:
assumes $J \subseteq_\circ I$
and $g : b \mapsto_{(\prod_{SMC} j \in_\circ I -_\circ J.\ \mathfrak{A}\ j)} c$
and $g' : b' \mapsto_{(\prod_{SMC} k \in_\circ J.\ \mathfrak{A}\ k)} c'$
and $f : a \mapsto_{(\prod_{SMC} j \in_\circ I -_\circ J.\ \mathfrak{A}\ j)} b$
and $f' : a' \mapsto_{(\prod_{SMC} k \in_\circ J.\ \mathfrak{A}\ k)} b'$
shows $(g \circ_{A(\prod_{SMC} j \in_\circ I -_\circ J.\ \mathfrak{A}\ j)} f) \cup_\circ (g' \circ_{A(\prod_{SMC} j \in_\circ J.\ \mathfrak{A}\ j)} f') =$
$g \cup_\circ g' \circ_{A(\prod_{SMC} j \in_\circ I.\ \mathfrak{A}\ j)} f \cup_\circ f'$
⟨*proof*⟩

4.8.7 Projection

Definition and elementary properties

See Chapter II-3 in [39].

definition *smcf-proj* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$ ($\langle \pi_{SMC} \rangle$)
where $\pi_{SMC}\ I\ \mathfrak{A}\ i =$
$[$
$(\lambda a \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ a(\!|i|\!))$,
$(\lambda f \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ f(\!|i|\!))$,
$(\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i)$,
$\mathfrak{A}\ i$
$]_\circ$

Components.

lemma *smcf-proj-components*:
shows $(\pi_{SMC}\ I\ \mathfrak{A}\ i)(\!|ObjMap|\!) = (\lambda a \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Obj|\!)).\ a(\!|i|\!))$
and $(\pi_{SMC}\ I\ \mathfrak{A}\ i)(\!|ArrMap|\!) = (\lambda f \in_\circ (\prod_\circ i \in_\circ I.\ \mathfrak{A}\ i(\!|Arr|\!)).\ f(\!|i|\!))$
and $(\pi_{SMC}\ I\ \mathfrak{A}\ i)(\!|HomDom|\!) = (\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i)$
and $(\pi_{SMC}\ I\ \mathfrak{A}\ i)(\!|HomCod|\!) = \mathfrak{A}\ i$
⟨*proof*⟩

Slicing

lemma *smcf-dghm-smcf-proj*[*slicing-commute*]:
$\pi_{DG}\ I\ (\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i))\ i = smcf\text{-}dghm\ (\pi_{SMC}\ I\ \mathfrak{A}\ i)$
⟨*proof*⟩

context *psemicategory*
begin

interpretation *pdg*: *pdigraph* $\alpha\ I\ \langle \lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i) \rangle$ ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
smcf-proj-is-dghm = *pdg.pdg-dghm-proj-is-dghm*

end

Projection semifunctor is a semifunctor

lemma (in *psemicategory***)** *psmc-smcf-proj-is-semifunctor*:
assumes $i \in_\circ I$
shows $\pi_{SMC}\ I\ \mathfrak{A}\ i : (\prod_{SMC} i \in_\circ I.\ \mathfrak{A}\ i) \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
⟨*proof*⟩

lemma (in *psemicategory***)** *psmc-smcf-proj-is-semifunctor'*:

assumes $i \in_{\circ} I$ **and** $\mathfrak{C} = (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$ **and** $\mathfrak{D} = \mathfrak{A}\ i$
shows $\pi_{SMC}\ I\ \mathfrak{A}\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
⟨*proof*⟩

lemmas [*smc-cs-intros*] = *psemicategory.psmc-smcf-proj-is-semifunctor'*

4.8.8 Semicategory product universal property semifunctor

Definition and elementary properties

The following semifunctor is used in the proof of the universal property of the product semicategory later in this work.

definition *smcf-up* :: $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$
where *smcf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi$ =
[
$(\lambda a \in_{\circ} \mathfrak{C}(\!|Obj|\!).\ (\lambda i \in_{\circ} I.\ \varphi\ i(\!|ObjMap|\!)(\!|a|\!)))$,
$(\lambda f \in_{\circ} \mathfrak{C}(\!|Arr|\!).\ (\lambda i \in_{\circ} I.\ \varphi\ i(\!|ArrMap|\!)(\!|f|\!)))$,
$\mathfrak{C}$,
$(\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$
]$_{\circ}$

Components.

lemma *smcf-up-components*:
shows *smcf-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 *smcf-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 *smcf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomDom|\!) = \mathfrak{C}$
and *smcf-up* $I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|HomCod|\!) = (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Slicing.

lemma *smcf-dghm-smcf-up*[*slicing-commute*]:
dghm-up I $(\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i))$ $(smc\text{-}dg\ \mathfrak{C})$ $(\lambda i.\ smcf\text{-}dghm\ (\varphi\ i))$ =
smcf-dghm $(smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi)$
⟨*proof*⟩

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

lemmas-with
[
where $\mathfrak{A}$=‹$\lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i)$› **and** φ=‹$\lambda i.\ smcf\text{-}dghm\ (\varphi\ i)$› **and** $\mathfrak{C}$=‹$smc\text{-}dg\ \mathfrak{C}$›,
unfolded slicing-simps slicing-commute
]:
smcf-up-ObjMap-vdomain[*smc-cs-simps*] = *dghm-up-ObjMap-vdomain*
and *smcf-up-ObjMap-app* = *dghm-up-ObjMap-app*
and *smcf-up-ObjMap-app-vdomain*[*smc-cs-simps*] = *dghm-up-ObjMap-app-vdomain*
and *smcf-up-ObjMap-app-component*[*smc-cs-simps*] = *dghm-up-ObjMap-app-component*
and *smcf-up-ArrMap-vdomain*[*smc-cs-simps*] = *dghm-up-ArrMap-vdomain*
and *smcf-up-ArrMap-app* = *dghm-up-ArrMap-app*
and *smcf-up-ArrMap-app-vdomain*[*smc-cs-simps*] = *dghm-up-ArrMap-app-vdomain*
and *smcf-up-ArrMap-app-component*[*smc-cs-simps*] = *dghm-up-ArrMap-app-component*

lemma *smcf-up-ObjMap-vrange*:
assumes $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_{\circ}\ (smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)) \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Obj|\!)$

⟨*proof*⟩

lemma *smcf-up-ObjMap-app-vrange*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Obj|\!)$ **and** $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_{\circ}\ (smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ObjMap|\!)(\!|a|\!)) \subseteq_{\circ} (\bigcup_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Obj|\!))$
⟨*proof*⟩

lemma *smcf-up-ArrMap-vrange*:
assumes $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_{\circ}\ (smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!)) \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)(\!|Arr|\!)$
⟨*proof*⟩

lemma *smcf-up-ArrMap-app-vrange*:
assumes $a \in_{\circ} \mathfrak{C}(\!|Arr|\!)$ **and** $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
shows $\mathcal{R}_{\circ}\ (smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi(\!|ArrMap|\!)(\!|a|\!)) \subseteq_{\circ} (\bigcup_{\circ} i \in_{\circ} I.\ \mathfrak{A}\ i(\!|Arr|\!))$
⟨*proof*⟩

end

context *psemicategory*
begin

interpretation *pdg*: *pdigraph* $\alpha\ I\ \langle \lambda i.\ smc\text{-}dg\ (\mathfrak{A}\ i)\rangle$ ⟨*proof*⟩

lemmas-with [*unfolded slicing-simps slicing-commute*]:
psmc-dghm-comp-dghm-proj-dghm-up = *pdg.pdg-dghm-comp-dghm-proj-dghm-up*
and *psmc-dghm-up-eq-dghm-proj* = *pdg.pdg-dghm-up-eq-dghm-proj*

end

Semicategory product universal property semifunctor is a semifunctor

lemma (**in** *psemicategory*) *psmc-smcf-up-is-semifunctor*:
assumes *semicategory* $\alpha\ \mathfrak{C}$ **and** $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
shows $smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$
⟨*proof*⟩

Further properties

lemma (**in** *psemicategory*) *psmc-Comp-smcf-proj-smcf-up*:
assumes *semicategory* $\alpha\ \mathfrak{C}$
and $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{A}\ i$
and $i \in_{\circ} I$
shows $\varphi\ i = \pi_{SMC}\ I\ \mathfrak{A}\ i \circ_{SMCF} smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi$
⟨*proof*⟩

lemma (**in** *psemicategory*) *psmc-smcf-up-eq-smcf-proj*:
assumes $\mathfrak{F} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} (\prod_{SMC} i \in_{\circ} I.\ \mathfrak{A}\ i)$
and $\bigwedge i.\ i \in_{\circ} I \Longrightarrow \varphi\ i = \pi_{SMC}\ I\ \mathfrak{A}\ i \circ_{SMCF} \mathfrak{F}$
shows $smcf\text{-}up\ I\ \mathfrak{A}\ \mathfrak{C}\ \varphi = \mathfrak{F}$
⟨*proof*⟩

4.8.9 Singleton semicategory

Slicing

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

lemmas-with [**where** $\mathfrak{C}$=‹*smc-dg* $\mathfrak{C}$›, *unfolded slicing-simps slicing-commute*]:
smc-singleton-ObjI = *dg-singleton-ObjI*
and *smc-singleton-ObjE* = *dg-singleton-ObjE*
and *smc-singleton-ArrI* = *dg-singleton-ArrI*
and *smc-singleton-ArrE* = *dg-singleton-ArrE*

end

context *semicategory*
begin

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

lemmas-with [*unfolded slicing-simps slicing-commute*]:
smc-finite-pdigraph-smc-singleton = *dg.dg-finite-pdigraph-dg-singleton*
and *smc-singleton-is-arrI* = *dg.dg-singleton-is-arrI*
and *smc-singleton-is-arrD* = *dg.dg-singleton-is-arrD*
and *smc-singleton-is-arrE* = *dg.dg-singleton-is-arrE*

end

Singleton semicategory is a semicategory

lemma (**in** *semicategory*) *smc-finite-psemicategory-smc-singleton*:
assumes $j \in_\circ$ *Vset* α
shows *finite-psemicategory* α (*set* $\{j\}$) ($\lambda i.$ $\mathfrak{C}$)
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-semicategory-smc-singleton*:
assumes $j \in_\circ$ *Vset* α
shows *semicategory* α ($\prod_{SMC} i \in_\circ$ *set* $\{j\}$. $\mathfrak{C}$)
⟨*proof*⟩

4.8.10 Singleton semifunctor

Definition and elementary properties

definition *smcf-singleton* :: $V \Rightarrow V \Rightarrow V$
where *smcf-singleton* j $\mathfrak{C}$ =
[
($\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!)$. *set* $\{\langle j, a\rangle\}$),
($\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!)$. *set* $\{\langle j, f\rangle\}$),
$\mathfrak{C}$,
($\prod_{SMC} i \in_\circ$ *set* $\{j\}$. $\mathfrak{C}$)
]$_\circ$

Components.

lemma *smcf-singleton-components*:
shows *smcf-singleton* j $\mathfrak{C}(\!|ObjMap|\!)$ = ($\lambda a \in_\circ \mathfrak{C}(\!|Obj|\!)$. *set* $\{\langle j, a\rangle\}$)
and *smcf-singleton* j $\mathfrak{C}(\!|ArrMap|\!)$ = ($\lambda f \in_\circ \mathfrak{C}(\!|Arr|\!)$. *set* $\{\langle j, f\rangle\}$)
and *smcf-singleton* j $\mathfrak{C}(\!|HomDom|\!)$ = $\mathfrak{C}$
and *smcf-singleton* j $\mathfrak{C}(\!|HomCod|\!)$ = ($\prod_{SMC} i \in_\circ$ *set* $\{j\}$. $\mathfrak{C}$)
⟨*proof*⟩

Slicing.

lemma *smcf-dghm-smcf-singleton*[*slicing-commute*]:
dghm-singleton j (*smc-dg* $\mathfrak{C}$)= *smcf-dghm* (*smcf-singleton* j $\mathfrak{C}$)

⟨*proof*⟩

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

lemmas-with [**where** $\mathfrak{C}$=‹*smc-dg* $\mathfrak{C}$›, *unfolded slicing-simps slicing-commute*]:
smcf-singleton-ObjMap-vsv[*smc-cs-intros*] = *dghm-singleton-ObjMap-vsv*
and *smcf-singleton-ObjMap-vdomain*[*smc-cs-simps*] =
dghm-singleton-ObjMap-vdomain
and *smcf-singleton-ObjMap-vrange* = *dghm-singleton-ObjMap-vrange*
and *smcf-singleton-ObjMap-app*[*smc-prod-cs-simps*] = *dghm-singleton-ObjMap-app*
and *smcf-singleton-ArrMap-vsv*[*smc-cs-intros*] = *dghm-singleton-ArrMap-vsv*
and *smcf-singleton-ArrMap-vdomain*[*smc-cs-simps*] =
dghm-singleton-ArrMap-vdomain
and *smcf-singleton-ArrMap-vrange* = *dghm-singleton-ArrMap-vrange*
and *smcf-singleton-ArrMap-app*[*smc-prod-cs-simps*] = *dghm-singleton-ArrMap-app*

end

context *semicategory*
begin

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

lemmas-with [*unfolded slicing-simps slicing-commute*]:
smc-smcf-singleton-is-dghm = *dg.dg-dghm-singleton-is-dghm*

end

Singleton semifunctor is an isomorphism of semicategories

lemma (**in** *semicategory*) *smc-smcf-singleton-is-iso-semifunctor*:
assumes $j \in_\circ$ *Vset* α
shows *smcf-singleton* j $\mathfrak{C}$: $\mathfrak{C}$ $\mapsto\!\!\mapsto_{SMC.iso\alpha}$ ($\prod_{SMC} i \in_\circ$ *set* $\{j\}$. $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*smc-cs-intros*] = *semicategory.smc-smcf-singleton-is-iso-semifunctor*

4.8.11 Product of two semicategories

Definition and elementary properties.

See Chapter II-3 in [39].

definition *smc-prod-2* :: $V \Rightarrow V \Rightarrow V$ (**infixr** ‹$\times_{SMC}$› 80)
where $\mathfrak{A} \times_{SMC} \mathfrak{B} \equiv$ *smc-prod* ($2_{\mathbb{N}}$) ($\lambda i.$ (i = 0 ? $\mathfrak{A}$: $\mathfrak{B}$))

Slicing.

lemma *smc-dg-smc-prod-2*[*slicing-commute*]:
smc-dg $\mathfrak{A}$ $\times_{DG}$ *smc-dg* $\mathfrak{B}$ = *smc-dg* ($\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩

interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩

lemmas-with
[
where $\mathfrak{A}$=‹*smc-dg* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*smc-dg* $\mathfrak{B}$›,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*smc-digraph* $\mathfrak{B}$.*smc-digraph*
]:
smc-prod-2-ObjI = *dg-prod-2-ObjI*
and *smc-prod-2-ObjI'*[*smc-prod-cs-intros*] = *dg-prod-2-ObjI'*
and *smc-prod-2-ObjE* = *dg-prod-2-ObjE*
and *smc-prod-2-ArrI* = *dg-prod-2-ArrI*
and *smc-prod-2-ArrI'*[*smc-prod-cs-intros*] = *dg-prod-2-ArrI'*
and *smc-prod-2-ArrE* = *dg-prod-2-ArrE*
and *smc-prod-2-is-arrI* = *dg-prod-2-is-arrI*
and *smc-prod-2-is-arrI'*[*smc-prod-cs-intros*] = *dg-prod-2-is-arrI'*
and *smc-prod-2-is-arrE* = *dg-prod-2-is-arrE*
and *smc-prod-2-Dom-vsv* = *dg-prod-2-Dom-vsv*
and *smc-prod-2-Dom-vdomain*[*smc-cs-simps*] = *dg-prod-2-Dom-vdomain*
and *smc-prod-2-Dom-app*[*smc-prod-cs-simps*] = *dg-prod-2-Dom-app*
and *smc-prod-2-Dom-vrange* = *dg-prod-2-Dom-vrange*
and *smc-prod-2-Cod-vsv* = *dg-prod-2-Cod-vsv*
and *smc-prod-2-Cod-vdomain*[*smc-cs-simps*] = *dg-prod-2-Cod-vdomain*
and *smc-prod-2-Cod-app*[*smc-prod-cs-simps*] = *dg-prod-2-Cod-app*
and *smc-prod-2-Cod-vrange* = *dg-prod-2-Cod-vrange*
and *smc-prod-2-op-smc-smc-Obj*[*smc-op-simps*] = *dg-prod-2-op-dg-dg-Obj*
and *smc-prod-2-smc-op-smc-Obj*[*smc-op-simps*] = *dg-prod-2-dg-op-dg-Obj*
and *smc-prod-2-op-smc-smc-Arr*[*smc-op-simps*] = *dg-prod-2-op-dg-dg-Arr*
and *smc-prod-2-smc-op-smc-Arr*[*smc-op-simps*] = *dg-prod-2-dg-op-dg-Arr*

end

Product of two semicategories is a semicategory

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *finite-psemicategory-smc-prod-2*:
finite-psemicategory α ($2_{\mathbb{N}}$) (*if*2 $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

interpretation *finite-psemicategory* α ‹$2_{\mathbb{N}}$› ‹*if*2 $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemma *semicategory-smc-prod-2*[*smc-cs-intros*]: *semicategory* α ($\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$)
⟨*proof*⟩

end

Composition

context

fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩

interpretation *finite-psemicategory* α ‹$2_{\mathbb{N}}$› ‹*if*2 $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemma *smc-prod-2-Comp-app*[*smc-prod-cs-simps*]:
assumes $[g,\ g']_{\circ} : [b,\ b']_{\circ} \mapsto_{\mathfrak{A} \times_{SMC} \mathfrak{B}} [c,\ c']_{\circ}$
and $[f,\ f']_{\circ} : [a,\ a']_{\circ} \mapsto_{\mathfrak{A} \times_{SMC} \mathfrak{B}} [b,\ b']_{\circ}$
shows $[g,\ g']_{\circ} \circ_{A\mathfrak{A} \times_{SMC} \mathfrak{B}} [f,\ f']_{\circ} = [g \circ_{A\mathfrak{A}} f,\ g' \circ_{A\mathfrak{B}} f']_{\circ}$
⟨*proof*⟩

end

Opposite product semicategory

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩

lemma *op-smc-smc-prod-2*[*smc-op-simps*]:
op-smc ($\mathfrak{A} \times_{SMC} \mathfrak{B}$) = *op-smc* $\mathfrak{A} \times_{SMC}$ *op-smc* $\mathfrak{B}$
⟨*proof*⟩

end

4.8.12 Projections for the product of two semicategories

Definition and elementary properties

See Chapter II-3 in [39].

definition *smcf-proj-fst* :: $V \Rightarrow V \Rightarrow V$ (‹$\pi_{SMC.1}$›)
where $\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$ = *smcf-proj* ($2_{\mathbb{N}}$) ($\lambda i.$ (i = 0 *?* $\mathfrak{A}$: $\mathfrak{B}$)) 0
definition *smcf-proj-snd* :: $V \Rightarrow V \Rightarrow V$ (‹$\pi_{SMC.2}$›)
where $\pi_{SMC.2}$ $\mathfrak{A}$ $\mathfrak{B}$ = *smcf-proj* ($2_{\mathbb{N}}$) ($\lambda i.$ (i = 0 *?* $\mathfrak{A}$: $\mathfrak{B}$)) ($1_{\mathbb{N}}$)

Slicing

lemma *smcf-dghm-smcf-proj-fst*[*slicing-commute*]:
$\pi_{DG.1}$ (*smc-dg* $\mathfrak{A}$) (*smc-dg* $\mathfrak{B}$) = *smcf-dghm* ($\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

lemma *smcf-dghm-smcf-proj-snd*[*slicing-commute*]:
$\pi_{DG.2}$ (*smc-dg* $\mathfrak{A}$) (*smc-dg* $\mathfrak{B}$) = *smcf-dghm* ($\pi_{SMC.2}$ $\mathfrak{A}$ $\mathfrak{B}$)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩

interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩

lemmas-with
[
where $\mathfrak{A}$=‹*smc-dg* $\mathfrak{A}$› **and** $\mathfrak{B}$=‹*smc-dg* $\mathfrak{B}$›,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*smc-digraph* $\mathfrak{B}$.*smc-digraph*
]:
smcf-proj-fst-ObjMap-app[*smc-cs-simps*] = *dghm-proj-fst-ObjMap-app*
and *smcf-proj-snd-ObjMap-app*[*smc-cs-simps*] = *dghm-proj-snd-ObjMap-app*
and *smcf-proj-fst-ArrMap-app*[*smc-cs-simps*] = *dghm-proj-fst-ArrMap-app*
and *smcf-proj-snd-ArrMap-app*[*smc-cs-simps*] = *dghm-proj-snd-ArrMap-app*

end

Domain and codomain of a projection of a product of two semicategories

lemma *smcf-proj-fst-HomDom*: $\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$(|*HomDom*|) = $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *smcf-proj-fst-HomCod*: $\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$(|*HomCod*|) = $\mathfrak{A}$
⟨*proof*⟩

lemma *smcf-proj-snd-HomDom*: $\pi_{SMC.2}$ $\mathfrak{A}$ $\mathfrak{B}$(|*HomDom*|) = $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *smcf-proj-snd-HomCod*: $\pi_{SMC.2}$ $\mathfrak{A}$ $\mathfrak{B}$(|*HomCod*|) = $\mathfrak{B}$
⟨*proof*⟩

Projection of a product of two semicategories is a semifunctor

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ **and** $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation *finite-psemicategory* α ‹$2_{\mathbb{N}}$› ‹*if*2 $\mathfrak{A}$ $\mathfrak{B}$›
⟨*proof*⟩

lemma *smcf-proj-fst-is-semifunctor*:
assumes $i \in_{\circ} I$
shows $\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$: $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *smcf-proj-fst-is-semifunctor'*[*smc-cs-intros*]:
assumes $i \in_{\circ} I$ **and** $\mathfrak{C}$ = $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$ **and** $\mathfrak{D}$ = $\mathfrak{A}$
shows $\pi_{SMC.1}$ $\mathfrak{A}$ $\mathfrak{B}$: $\mathfrak{C}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{D}$
⟨*proof*⟩

lemma *smcf-proj-snd-is-semifunctor*:
assumes $i \in_{\circ} I$
shows $\pi_{SMC.2}$ $\mathfrak{A}$ $\mathfrak{B}$: $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$ $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{B}$
⟨*proof*⟩

lemma *smcf-proj-snd-is-semifunctor'*[*smc-cs-intros*]:
assumes $i \in_{\circ} I$ **and** $\mathfrak{C}$ = $\mathfrak{A}$ $\times_{SMC}$ $\mathfrak{B}$ **and** $\mathfrak{D}$ = $\mathfrak{B}$

shows $\pi_{SMC.2}\ \mathfrak{A}\ \mathfrak{B} : \mathfrak{C} \mapsto\mapsto_{SMC\alpha} \mathfrak{D}$
⟨*proof*⟩

end

4.8.13 Product of three semicategories

Definition and elementary properties.

definition *smc-prod-3* :: $V \Rightarrow V \Rightarrow V \Rightarrow V$
($\langle$(- $\times_{SMC3}$ - $\times_{SMC3}$ -)$\rangle$ [81, 81, 81] 80)
where $\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C}$ = $(\prod_{SMC} i \in_{\circ} 3_{\mathbb{N}}.\ if3\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{C}\ i)$

Slicing.

lemma *smc-dg-smc-prod-3*[*slicing-commute*]:
smc-dg $\mathfrak{A}$ $\times_{DG3}$ *smc-dg* $\mathfrak{B}$ $\times_{DG3}$ *smc-dg* $\mathfrak{C}$ = *smc-dg* ($\mathfrak{A}$ $\times_{SMC3}$ $\mathfrak{B}$ $\times_{SMC3}$ $\mathfrak{C}$)
⟨*proof*⟩

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$
and $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
and $\mathfrak{C}$: *semicategory* α $\mathfrak{C}$
begin

interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *semicategory* α $\mathfrak{C}$ ⟨*proof*⟩

lemmas-with
[
where $\mathfrak{A}$=$\langle$*smc-dg* $\mathfrak{A}$$\rangle$ **and** $\mathfrak{B}$=$\langle$*smc-dg* $\mathfrak{B}$$\rangle$ **and** $\mathfrak{C}$=$\langle$*smc-dg* $\mathfrak{C}$$\rangle$,
unfolded slicing-simps slicing-commute,
OF $\mathfrak{A}$.*smc-digraph* $\mathfrak{B}$.*smc-digraph* $\mathfrak{C}$.*smc-digraph*
]:
smc-prod-3-ObjI = *dg-prod-3-ObjI*
and *smc-prod-3-ObjI'*[*smc-prod-cs-intros*] = *dg-prod-3-ObjI'*
and *smc-prod-3-ObjE* = *dg-prod-3-ObjE*
and *smc-prod-3-ArrI* = *dg-prod-3-ArrI*
and *smc-prod-3-ArrI'*[*smc-prod-cs-intros*] = *dg-prod-3-ArrI'*
and *smc-prod-3-ArrE* = *dg-prod-3-ArrE*
and *smc-prod-3-is-arrI* = *dg-prod-3-is-arrI*
and *smc-prod-3-is-arrI'*[*smc-prod-cs-intros*] = *dg-prod-3-is-arrI'*
and *smc-prod-3-is-arrE* = *dg-prod-3-is-arrE*
and *smc-prod-3-Dom-vsv* = *dg-prod-3-Dom-vsv*
and *smc-prod-3-Dom-vdomain*[*smc-cs-simps*] = *dg-prod-3-Dom-vdomain*
and *smc-prod-3-Dom-app*[*smc-prod-cs-simps*] = *dg-prod-3-Dom-app*
and *smc-prod-3-Dom-vrange* = *dg-prod-3-Dom-vrange*
and *smc-prod-3-Cod-vsv* = *dg-prod-3-Cod-vsv*
and *smc-prod-3-Cod-vdomain*[*smc-cs-simps*] = *dg-prod-3-Cod-vdomain*
and *smc-prod-3-Cod-app*[*smc-prod-cs-simps*] = *dg-prod-3-Cod-app*
and *smc-prod-3-Cod-vrange* = *dg-prod-3-Cod-vrange*

end

Product of three semicategories is a semicategory

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$

assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$
and $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
and $\mathfrak{C}$: *semicategory* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩
interpretation $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$ ⟨*proof*⟩
interpretation $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$ ⟨*proof*⟩
interpretation $\mathfrak{C}$: *semicategory* α $\mathfrak{C}$ ⟨*proof*⟩

lemma *finite-psemicategory-smc-prod-3*:
finite-psemicategory α $(3_{\mathbb{N}})$ (*if*3 $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$)
⟨*proof*⟩

interpretation *finite-psemicategory* α ‹$3_{\mathbb{N}}$› ‹*if*3 $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemma *semicategory-smc-prod-3*[*smc-cs-intros*]:
semicategory α $(\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C})$
⟨*proof*⟩

end

Composition

context
fixes α $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$
assumes $\mathfrak{A}$: *semicategory* α $\mathfrak{A}$
and $\mathfrak{B}$: *semicategory* α $\mathfrak{B}$
and $\mathfrak{C}$: *semicategory* α $\mathfrak{C}$
begin

interpretation $\mathcal{Z}$ α ⟨*proof*⟩

interpretation *finite-psemicategory* α ‹$3_{\mathbb{N}}$› ‹*if*3 $\mathfrak{A}$ $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemma *smc-prod-3-Comp-app*[*smc-prod-cs-simps*]:
assumes $[g,\ g',\ g'']_{\circ} : [b,\ b',\ b'']_{\circ} \mapsto_{\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C}} [c,\ c',\ c'']_{\circ}$
and $[f,\ f',\ f'']_{\circ} : [a,\ a',\ a'']_{\circ} \mapsto_{\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C}} [b,\ b',\ b'']_{\circ}$
shows
$[g,\ g',\ g'']_{\circ} \circ_{A\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C}} [f,\ f',\ f'']_{\circ} =$
$[g \circ_{A\mathfrak{A}} f,\ g' \circ_{A\mathfrak{B}} f',\ g'' \circ_{A\mathfrak{C}} f'']_{\circ}$
⟨*proof*⟩

end

4.9 Subsemicategory

4.9.1 Background

named-theorems *smc-sub-cs-intros*
named-theorems *smc-sub-bw-cs-intros*
named-theorems *smc-sub-fw-cs-intros*
named-theorems *smc-sub-bw-cs-simps*

4.9.2 Simple subsemicategory

Definition and elementary properties

See Chapter I-3 in [39].

locale *subsemicategory* =
  sdg: *semicategory* α $\mathfrak{B}$ + *dg*: *semicategory* α $\mathfrak{C}$ **for** α $\mathfrak{B}$ $\mathfrak{C}$ +
  assumes *subsmc-subdigraph*[*slicing-intros*]: *smc-dg* $\mathfrak{B}$ $\subseteq_{DG\alpha}$ *smc-dg* $\mathfrak{C}$
    and *subsmc-Comp*[*smc-sub-fw-cs-intros*]:
      $[\![\ g : b \mapsto_{\mathfrak{B}} c;\ f : a \mapsto_{\mathfrak{B}} b\ ]\!] \Longrightarrow g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$

abbreviation *is-subsemicategory* ($\langle$(-/ $\subseteq_{SMC^1}$ -)$\rangle$ [51, 51] 50)
  where $\mathfrak{B} \subseteq_{SMC\alpha} \mathfrak{C} \equiv$ *subsemicategory* α $\mathfrak{B}$ $\mathfrak{C}$

lemmas [*smc-sub-fw-cs-intros*] = *subsemicategory.subsmc-Comp*

Rules.

lemma (**in** *subsemicategory*) *subsemicategory-axioms'*[*smc-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
  shows $\mathfrak{B}' \subseteq_{SMC_{\alpha'}} \mathfrak{C}$
  ⟨*proof*⟩

lemma (**in** *subsemicategory*) *subsemicategory-axioms''*[*smc-cs-intros*]:
  assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
  shows $\mathfrak{B} \subseteq_{SMC_{\alpha'}} \mathfrak{C}'$
  ⟨*proof*⟩

mk-ide rf *subsemicategory-def*[*unfolded subsemicategory-axioms-def*]
  |*intro subsemicategoryI*|
  |*dest subsemicategoryD*[*dest*]|
  |*elim subsemicategoryE*[*elim*!]|

lemmas [*smc-sub-cs-intros*] = *subsemicategoryD*(1,2)

lemma *subsemicategoryI'*:
  assumes *semicategory* α $\mathfrak{B}$
    and *semicategory* 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 $\bigwedge b\ c\ g\ a\ f.\ [\![\ g : b \mapsto_{\mathfrak{B}} c;\ f : a \mapsto_{\mathfrak{B}} b\ ]\!] \Longrightarrow$
      $g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$
  shows $\mathfrak{B} \subseteq_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

Subsemicategory is a subdigraph.

context *subsemicategory*
begin

interpretation *subdg*: *subdigraph* α ‹*smc-dg* $\mathfrak{B}$› ‹*smc-dg* $\mathfrak{C}$›

⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
subsmc-Obj-vsubset = *subdg.subdg-Obj-vsubset*
and *subsmc-is-arr-vsubset* = *subdg.subdg-is-arr-vsubset*
and *subsmc-subdigraph-op-dg-op-dg* = *subdg.subdg-subdigraph-op-dg-op-dg*
and *subsmc-objD* = *subdg.subdg-objD*
and *subsmc-arrD* = *subdg.subdg-arrD*
and *subsmc-dom-simp* = *subdg.subdg-dom-simp*
and *subsmc-cod-simp* = *subdg.subdg-cod-simp*
and *subsmc-is-arrD* = *subdg.subdg-is-arrD*
and *subsmc-dghm-inc-op-dg-is-dghm* = *subdg.subdg-dghm-inc-op-dg-is-dghm*
and *subsmc-op-dg-dghm-inc* = *subdg.subdg-op-dg-dghm-inc*
and *subsmc-inc-is-ft-dghm-axioms* = *subdg.inc.is-ft-dghm-axioms*

end

lemmas *subsmc-subdigraph-op-dg-op-dg*[*intro*] =
subsemicategory.subsmc-subdigraph-op-dg-op-dg

lemmas [*smc-sub-fw-cs-intros*] =
subsemicategory.subsmc-Obj-vsubset
subsemicategory.subsmc-is-arr-vsubset
subsemicategory.subsmc-objD
subsemicategory.subsmc-arrD
subsemicategory.subsmc-is-arrD

lemmas [*smc-sub-bw-cs-simps*] =
subsemicategory.subsmc-dom-simp
subsemicategory.subsmc-cod-simp

The opposite subsemicategory.

lemma (**in** *subsemicategory*) *subsmc-subsemicategory-op-smc*:
op-smc $\mathfrak{B}$ $\subseteq_{SMC\alpha}$ *op-smc* $\mathfrak{C}$
⟨*proof*⟩

lemmas *subsmc-subsemicategory-op-smc*[*intro*, *smc-op-intros*] =
subsemicategory.subsmc-subsemicategory-op-smc

Further rules.

lemma (**in** *subsemicategory*) *subsmc-Comp-simp*:
assumes $g : b \mapsto_{\mathfrak{B}} c$ **and** $f : a \mapsto_{\mathfrak{B}} b$
shows $g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$
⟨*proof*⟩

lemmas [*smc-sub-bw-cs-simps*] = *subsemicategory.subsmc-Comp-simp*

lemma (**in** *subsemicategory*) *subsmc-is-idem-arrD*:
assumes $f : \mapsto_{ide\mathfrak{B}} b$
shows $f : \mapsto_{ide\mathfrak{C}} b$
⟨*proof*⟩

lemmas [*smc-sub-fw-cs-intros*] = *subsemicategory.subsmc-is-idem-arrD*

Subsemicategory relation is a partial order

lemma *subsmc-refl*:
assumes *semicategory* α $\mathfrak{A}$

shows $\mathfrak{A} \subseteq_{SMC\alpha} \mathfrak{A}$
⟨*proof*⟩

lemma *subsmc-trans*[*trans*]:
assumes $\mathfrak{A} \subseteq_{SMC\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{SMC\alpha} \mathfrak{C}$
shows $\mathfrak{A} \subseteq_{SMC\alpha} \mathfrak{C}$
⟨*proof*⟩

lemma *subsmc-antisym*:
assumes $\mathfrak{A} \subseteq_{SMC\alpha} \mathfrak{B}$ **and** $\mathfrak{B} \subseteq_{SMC\alpha} \mathfrak{A}$
shows $\mathfrak{A} = \mathfrak{B}$
⟨*proof*⟩

4.9.3 Inclusion semifunctor

Definition and elementary properties

See Chapter I-3 in [39].

abbreviation (*input*) *smcf-inc* :: $V \Rightarrow V \Rightarrow V$
where *smcf-inc* ≡ *dghm-inc*

Slicing.

lemma *dghm-smcf-inc*[*slicing-commute*]:
dghm-inc (*smc-dg* $\mathfrak{B}$) (*smc-dg* $\mathfrak{C}$) = *smcf-dghm* (*smcf-inc* $\mathfrak{B}$ $\mathfrak{C}$)
⟨*proof*⟩

Elementary properties.

lemmas [*smc-cs-simps*] =
dghm-inc-ObjMap-app
dghm-inc-ArrMap-app

Canonical inclusion semifunctor associated with a subsemicategory

sublocale *subsemicategory* ⊆ *inc*: *is-ft-semifunctor* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*smcf-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

lemmas (**in** *subsemicategory*) *subsmc-smcf-inc-is-ft-semifunctor* =
inc.is-ft-semifunctor-axioms

Inclusion semifunctor for the opposite semicategories

lemma (**in** *subsemicategory*)
subsemicategory-smcf-inc-op-smc-is-semifunctor[*smc-sub-cs-intros*]:
smcf-inc (*op-smc* $\mathfrak{B}$) (*op-smc* $\mathfrak{C}$) : *op-smc* $\mathfrak{B}$ $\mapsto\!\!\mapsto_{SMC.faithful\alpha}$ *op-smc* $\mathfrak{C}$
⟨*proof*⟩

lemmas [*smc-sub-cs-intros*] =
subsemicategory.subsemicategory-smcf-inc-op-smc-is-semifunctor

lemma (**in** *subsemicategory*) *subdg-op-smc-smcf-inc*[*smc-op-simps*]:
op-smcf (*smcf-inc* $\mathfrak{B}$ $\mathfrak{C}$) = *smcf-inc* (*op-smc* $\mathfrak{B}$) (*op-smc* $\mathfrak{C}$)
⟨*proof*⟩

lemmas [*smc-op-simps*] = *subsemicategory.subdg-op-smc-smcf-inc*

4.9.4 Full subsemicategory

See Chapter I-3 in [39].

locale *fl-subsemicategory* = *subsemicategory* +
assumes *fl-subsemicategory-fl-subdigraph*: *smc-dg* $\mathfrak{B}$ $\subseteq_{DG.full\alpha}$ *smc-dg* $\mathfrak{C}$

abbreviation *is-fl-subsemicategory* (‹(-/ $\subseteq_{SMC.full_1}$ -)› [51, 51] 50)
where $\mathfrak{B}$ $\subseteq_{SMC.full\alpha}$ $\mathfrak{C}$ ≡ *fl-subsemicategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

lemma (in *fl-subsemicategory*) *fl-subsemicategory-axioms'*[*smc-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{B}' \subseteq_{SMC.full_{\alpha'}} \mathfrak{C}$
⟨*proof*⟩

lemma (in *fl-subsemicategory*) *fl-subsemicategory-axioms''*[*smc-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{B} \subseteq_{SMC.full_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *fl-subsemicategory-def*[*unfolded fl-subsemicategory-axioms-def*]
|*intro fl-subsemicategoryI*|
|*dest fl-subsemicategoryD*[*dest*]|
|*elim fl-subsemicategoryE*[*elim*!]|

lemmas [*smc-sub-cs-intros*] = *fl-subsemicategoryD*(1)

Full subsemicategory.

sublocale *fl-subsemicategory* ⊆ *inc*: *is-fl-semifunctor* α $\mathfrak{B}$ $\mathfrak{C}$ ‹*smcf-inc* $\mathfrak{B}$ $\mathfrak{C}$›
⟨*proof*⟩

4.9.5 Wide subsemicategory

Definition and elementary properties

See [3][4]).

locale *wide-subsemicategory* = *subsemicategory* +
assumes *wide-subsmc-wide-subdigraph*: *smc-dg* $\mathfrak{B}$ $\subseteq_{DG.wide\alpha}$ *smc-dg* $\mathfrak{C}$

abbreviation *is-wide-subsemicategory* (‹(-/ $\subseteq_{SMC.wide_1}$ -)› [51, 51] 50)
where $\mathfrak{B}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{C}$ ≡ *wide-subsemicategory* α $\mathfrak{B}$ $\mathfrak{C}$

Rules.

lemma (in *wide-subsemicategory*) *wide-subsemicategory-axioms'*[*smc-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{B}' = \mathfrak{B}$
shows $\mathfrak{B}' \subseteq_{SMC.wide_{\alpha'}} \mathfrak{C}$
⟨*proof*⟩

lemma (in *wide-subsemicategory*) *wide-subsemicategory-axioms''*[*smc-cs-intros*]:
assumes $\alpha' = \alpha$ **and** $\mathfrak{C}' = \mathfrak{C}$
shows $\mathfrak{B} \subseteq_{SMC.wide_{\alpha'}} \mathfrak{C}'$
⟨*proof*⟩

mk-ide rf *wide-subsemicategory-def*[*unfolded wide-subsemicategory-axioms-def*]
|*intro wide-subsemicategoryI*|
|*dest wide-subsemicategoryD*[*dest*]|
|*elim wide-subsemicategoryE*[*elim*!]|

[4] https://ncatlab.org/nlab/show/wide+subcategory

lemmas [*smc-sub-cs-intros*] = *wide-subsemicategoryD*(1)

Wide subsemicategory is wide subdigraph.

context *wide-subsemicategory*
begin

interpretation *wide-subdg*: *wide-subdigraph* α ‹*smc-dg* $\mathfrak{B}$› ‹*smc-dg* $\mathfrak{C}$›
⟨*proof*⟩

lemmas-with [*unfolded slicing-simps*]:
wide-subsmc-Obj[*dg-sub-bw-cs-intros*] = *wide-subdg.wide-subdg-Obj*
and *wide-subsmc-obj-eq*[*dg-sub-bw-cs-simps*] = *wide-subdg.wide-subdg-obj-eq*

end

lemmas [*dg-sub-bw-cs-intros*] = *wide-subsemicategory.wide-subsmc-Obj*
lemmas [*dg-sub-bw-cs-simps*] = *wide-subsemicategory.wide-subsmc-obj-eq*

The wide subsemicategory relation is a partial order

lemma *wide-subsmc-refl*:
assumes *semicategory* α $\mathfrak{A}$
shows $\mathfrak{A}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma *wide-subsmc-trans*[*trans*]:
assumes $\mathfrak{A}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{C}$
shows $\mathfrak{A}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma *wide-subsmc-antisym*:
assumes $\mathfrak{A}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{B}$ **and** $\mathfrak{B}$ $\subseteq_{SMC.wide\alpha}$ $\mathfrak{A}$
shows $\mathfrak{A}$ = $\mathfrak{B}$
⟨*proof*⟩

4.10 Simple semicategories

4.10.1 Background

The section presents a variety of simple semicategories, such as the empty semicategory 0 and a semicategory with one object and one arrow 1. All of the entities presented in this section are generalizations of certain simple categories, whose definitions can be found in [39].

4.10.2 Empty semicategory 0

Definition and elementary properties

See Chapter I-2 in [39].

definition *smc*-0 :: *V*
  where *smc*-0 = [0, 0, 0, 0, 0]$_\circ$

Components.

lemma *smc*-0-*components*:
  shows *smc*-0(|*Obj*|) = 0
    and *smc*-0(|*Arr*|) = 0
    and *smc*-0(|*Dom*|) = 0
    and *smc*-0(|*Cod*|) = 0
    and *smc*-0(|*Comp*|) = 0
  ⟨*proof*⟩

Slicing.

lemma *smc-dg-smc*-0: *smc-dg smc*-0 = *dg*-0
  ⟨*proof*⟩

lemmas-with (**in** $\mathcal{Z}$) [*folded smc-dg-smc*-0, *unfolded slicing-simps*]:
  smc-0-*is-arr-iff* = *dg*-0-*is-arr-iff*

0 is a semicategory

lemma (**in** $\mathcal{Z}$) *semicategory-smc*-0[*smc-cs-intros*]: *semicategory* α *smc*-0
⟨*proof*⟩

lemmas [*smc-cs-intros*] = $\mathcal{Z}$.*semicategory-smc*-0

Opposite of the semicategory 0

lemma *op-smc-smc*-0[*smc-op-simps*]: *op-smc* (*smc*-0) = *smc*-0
⟨*proof*⟩

A semicategory without objects is empty

lemma (**in** *semicategory*) *smc-smc*-0-*if-Obj*-0:
  assumes $\mathfrak{C}$(|*Obj*|) = 0
  shows $\mathfrak{C}$ = *smc*-0
  ⟨*proof*⟩

4.10.3 Empty semifunctor

An empty semifunctor is defined as a semifunctor between an empty semicategory and an arbitrary semicategory.

Definition and elementary properties

definition *smcf*-0 :: $V \Rightarrow V$
  where *smcf*-0 $\mathfrak{A}$ = $[0, 0, smc\text{-}0, \mathfrak{A}]_{\circ}$

Components.

lemma *smcf*-0-*components*:
  shows *smcf*-0 $\mathfrak{A}(\lfloor ObjMap \rfloor)$ = 0
    and *smcf*-0 $\mathfrak{A}(\lfloor ArrMap \rfloor)$ = 0
    and *smcf*-0 $\mathfrak{A}(\lfloor HomDom \rfloor)$ = *smc*-0
    and *smcf*-0 $\mathfrak{A}(\lfloor HomCod \rfloor)$ = $\mathfrak{A}$
  ⟨*proof*⟩

Slicing.

lemma *smcf-dghm-smcf*-0: *smcf-dghm* (*smcf*-0 $\mathfrak{A}$) = *dghm*-0 (*smc-dg* $\mathfrak{A}$)
  ⟨*proof*⟩

Opposite empty semicategory homomorphism.

lemma *op-smcf-smcf*-0: *op-smcf* (*smcf*-0 $\mathfrak{C}$) = *smcf*-0 (*op-smc* $\mathfrak{C}$)
  ⟨*proof*⟩

Object map

lemma *smcf*-0-*ObjMap-vsv*[*smc-cs-intros*]: *vsv* (*smcf*-0 $\mathfrak{C}(\lfloor ObjMap \rfloor)$)
  ⟨*proof*⟩

Arrow map

lemma *smcf*-0-*ArrMap-vsv*[*smc-cs-intros*]: *vsv* (*smcf*-0 $\mathfrak{C}(\lfloor ArrMap \rfloor)$)
  ⟨*proof*⟩

Empty semifunctor is a faithful semifunctor

lemma (**in** $\mathcal{Z}$) *smcf*-0-*is-ft-semifunctor*:
  assumes *semicategory* α $\mathfrak{A}$
  shows *smcf*-0 $\mathfrak{A}$: *smc*-0 $\mapsto\!\!\mapsto_{SMC.faithful\alpha}$ $\mathfrak{A}$
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smcf*-0-*is-ft-semifunctor'*[*smcf-cs-intros*]:
  assumes *semicategory* α $\mathfrak{A}$
    and $\mathfrak{B}'$ = $\mathfrak{A}$
    and $\mathfrak{A}'$ = *smc*-0
  shows *smcf*-0 $\mathfrak{A}$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{SMC.faithful\alpha}$ $\mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*smcf-cs-intros*] = $\mathcal{Z}$.*smcf*-0-*is-ft-semifunctor'*

lemma (**in** $\mathcal{Z}$) *smcf*-0-*is-semifunctor*:
  assumes *semicategory* α $\mathfrak{A}$
  shows *smcf*-0 $\mathfrak{A}$: *smc*-0 $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{A}$
  ⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smcf*-0-*is-semifunctor'*[*smc-cs-intros*]:
  assumes *semicategory* α $\mathfrak{A}$
    and $\mathfrak{B}'$ = $\mathfrak{A}$
    and $\mathfrak{A}'$ = *smc*-0
  shows *smcf*-0 $\mathfrak{A}$: $\mathfrak{A}'$ $\mapsto\!\!\mapsto_{SMC\alpha}$ $\mathfrak{B}'$
  ⟨*proof*⟩

lemmas [*smc-cs-intros*] = $\mathcal{Z}$.*smcf*-0-*is-semifunctor*′

Further properties

lemma *is-semifunctor-is-smcf*-0-*if-smc*-0:
assumes $\mathfrak{F}$: *smc*-0 $\mapsto\mapsto_{SMC\alpha}$ $\mathfrak{C}$
shows $\mathfrak{F}$ = *smcf*-0 $\mathfrak{C}$
⟨*proof*⟩

4.10.4 Empty natural transformation of semifunctors

Definition and elementary properties

definition *ntsmcf*-0 :: $V \Rightarrow V$
where *ntsmcf*-0 $\mathfrak{C}$ = [0, *smcf*-0 $\mathfrak{C}$, *smcf*-0 $\mathfrak{C}$, *smc*-0, $\mathfrak{C}$]$_\circ$

Components.

lemma *ntsmcf*-0-*components*:
shows *ntsmcf*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$ = 0
and [*smc-cs-simps*]: *ntsmcf*-0 $\mathfrak{C}(\lfloor NTDom \rfloor)$ = *smcf*-0 $\mathfrak{C}$
and [*smc-cs-simps*]: *ntsmcf*-0 $\mathfrak{C}(\lfloor NTCod \rfloor)$ = *smcf*-0 $\mathfrak{C}$
and [*smc-cs-simps*]: *ntsmcf*-0 $\mathfrak{C}(\lfloor NTDGDom \rfloor)$ = *smc*-0
and [*smc-cs-simps*]: *ntsmcf*-0 $\mathfrak{C}(\lfloor NTDGCod \rfloor)$ = $\mathfrak{C}$
⟨*proof*⟩

Slicing.

lemma *ntsmcf-tdghm-ntsmcf*-0: *ntsmcf-tdghm* (*ntsmcf*-0 $\mathfrak{A}$) = *tdghm*-0 (*smc-dg* $\mathfrak{A}$)
⟨*proof*⟩

Duality.

lemma *op-ntsmcf-ntsmcf*-0: *op-ntsmcf* (*ntsmcf*-0 $\mathfrak{C}$) = *ntsmcf*-0 (*op-smc* $\mathfrak{C}$)
⟨*proof*⟩

Natural transformation map

lemma *ntsmcf*-0-*NTMap-vsv*[*smc-cs-intros*]: *vsv* (*ntsmcf*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$)
⟨*proof*⟩

lemma *ntsmcf*-0-*NTMap-vdomain*[*smc-cs-simps*]: $\mathcal{D}_\circ$ (*ntsmcf*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$) = 0
⟨*proof*⟩

lemma *ntsmcf*-0-*NTMap-vrange*[*smc-cs-simps*]: $\mathcal{R}_\circ$ (*ntsmcf*-0 $\mathfrak{C}(\lfloor NTMap \rfloor)$) = 0
⟨*proof*⟩

Empty natural transformation of semifunctors is a natural transformation of semifunctors

lemma (**in** *semicategory*) *smc-ntsmcf*-0-*is-ntsmcfI*:
ntsmcf-0 $\mathfrak{C}$: *smcf*-0 $\mathfrak{C}$ $\mapsto_{SMCF}$ *smcf*-0 $\mathfrak{C}$: *smc*-0 $\mapsto\mapsto_{SMC\alpha}$ $\mathfrak{C}$
⟨*proof*⟩

lemma (**in** *semicategory*) *smc-ntsmcf*-0-*is-ntsmcfI*′[*smc-cs-intros*]:
assumes $\mathfrak{F}'$ = *smcf*-0 $\mathfrak{C}$
and $\mathfrak{G}'$ = *smcf*-0 $\mathfrak{C}$
and $\mathfrak{A}'$ = *smc*-0
and $\mathfrak{B}'$ = $\mathfrak{C}$
and $\mathfrak{F}'$ = $\mathfrak{F}$

and $\mathfrak{G}' = \mathfrak{G}$
shows *ntsmcf*-0 $\mathfrak{C} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC\alpha} \mathfrak{B}'$
⟨*proof*⟩

lemmas [*smc-cs-intros*] = *semicategory.smc-ntsmcf*-0-*is-ntsmcfI'*

lemma *is-ntsmcf-is-ntsmcf*-0-*if-smc*-0:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
shows $\mathfrak{N} = ntsmcf\text{-}0\ \mathfrak{C}$ **and** $\mathfrak{F} = smcf\text{-}0\ \mathfrak{C}$ **and** $\mathfrak{G} = smcf\text{-}0\ \mathfrak{C}$
⟨*proof*⟩

Further properties

lemma *ntsmcf-vcomp-ntsmcf-ntsmcf*-0[*smc-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
shows $\mathfrak{N} \cdot_{NTSMCF} ntsmcf\text{-}0\ \mathfrak{C} = ntsmcf\text{-}0\ \mathfrak{C}$
⟨*proof*⟩

lemma *ntsmcf-vcomp-ntsmcf*-0-*ntsmcf*[*smc-cs-simps*]:
assumes $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto\mapsto_{SMC\alpha} \mathfrak{C}$
shows $ntsmcf\text{-}0\ \mathfrak{C} \cdot_{NTSMCF} \mathfrak{N} = ntsmcf\text{-}0\ \mathfrak{C}$
⟨*proof*⟩

4.10.5 10: semicategory with one object and no arrows

Definition and elementary properties

definition *smc*-10 :: $V \Rightarrow V$
where $smc\text{-}10\ \mathfrak{a} = [set\ \{\mathfrak{a}\},\ 0,\ 0,\ 0,\ 0]_\circ$

Components.

lemma *smc*-10-*components*:
shows $smc\text{-}10\ \mathfrak{a}(\lfloor Obj \rfloor) = set\ \{\mathfrak{a}\}$
and $smc\text{-}10\ \mathfrak{a}(\lfloor Arr \rfloor) = 0$
and $smc\text{-}10\ \mathfrak{a}(\lfloor Dom \rfloor) = 0$
and $smc\text{-}10\ \mathfrak{a}(\lfloor Cod \rfloor) = 0$
and $smc\text{-}10\ \mathfrak{a}(\lfloor Comp \rfloor) = 0$
⟨*proof*⟩

Slicing.

lemma *smc-dg-smc*-10: $smc\text{-}dg\ (smc\text{-}10\ \mathfrak{a}) = (dg\text{-}10\ \mathfrak{a})$
⟨*proof*⟩

lemmas-with (**in** $\mathcal{Z}$) [*folded smc-dg-smc*-10, *unfolded slicing-simps*]:
smc-10-*is-arr-iff* = *dg*-10-*is-arr-iff*

10 is a semicategory

lemma (**in** $\mathcal{Z}$) *semicategory-smc*-10:
assumes $\mathfrak{a} \in_\circ Vset\ \alpha$
shows $semicategory\ \alpha\ (smc\text{-}10\ \mathfrak{a})$
⟨*proof*⟩

Arrow with a domain and a codomain

lemma *smc*-10-*is-arr-iff*: $\mathfrak{F} : \mathfrak{A} \mapsto_{smc\text{-}10\ \mathfrak{a}} \mathfrak{B} \longleftrightarrow False$
⟨*proof*⟩

4.10.6 1: semicategory with one object and one arrow

Definition and elementary properties

definition *smc*-1 :: $V \Rightarrow V \Rightarrow V$
  where *smc*-1 $\mathfrak{a}$ $\mathfrak{f}$ =
    $[set\ \{\mathfrak{a}\}, set\ \{\mathfrak{f}\}, set\ \{\langle \mathfrak{f}, \mathfrak{a}\rangle\}, set\ \{\langle \mathfrak{f}, \mathfrak{a}\rangle\}, set\ \{\langle [\mathfrak{f}, \mathfrak{f}]_\circ, \mathfrak{f}\rangle\}]_\circ$

Components.

lemma *smc*-1-*components*:
  shows *smc*-1 $\mathfrak{a}$ $\mathfrak{f}(\vert Obj \vert) = set\ \{\mathfrak{a}\}$
    and *smc*-1 $\mathfrak{a}$ $\mathfrak{f}(\vert Arr \vert) = set\ \{\mathfrak{f}\}$
    and *smc*-1 $\mathfrak{a}$ $\mathfrak{f}(\vert Dom \vert) = set\ \{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$
    and *smc*-1 $\mathfrak{a}$ $\mathfrak{f}(\vert Cod \vert) = set\ \{\langle \mathfrak{f}, \mathfrak{a}\rangle\}$
    and *smc*-1 $\mathfrak{a}$ $\mathfrak{f}(\vert Comp \vert) = set\ \{\langle [\mathfrak{f}, \mathfrak{f}]_\circ, \mathfrak{f}\rangle\}$
  ⟨*proof*⟩

Slicing.

lemma *dg-smc*-1: *smc-dg* (*smc*-1 $\mathfrak{a}$ $\mathfrak{f}$) = *dg*-1 $\mathfrak{a}$ $\mathfrak{f}$
  ⟨*proof*⟩

lemmas-with [*folded dg-smc*-1, *unfolded slicing-simps*]:
  smc-1-*is-arrI* = *dg*-1-*is-arrI*
  and *smc*-1-*is-arrD* = *dg*-1-*is-arrD*
  and *smc*-1-*is-arrE* = *dg*-1-*is-arrE*
  and *smc*-1-*is-arr-iff* = *dg*-1-*is-arr-iff*

Composition

lemma *smc*-1-*Comp-app*[*simp*]: $\mathfrak{f} \circ_{A\ smc\text{-}1\ \mathfrak{a}\ \mathfrak{f}} \mathfrak{f} = \mathfrak{f}$
  ⟨*proof*⟩

1 is a semicategory

lemma (**in** $\mathcal{Z}$) *semicategory-smc*-1:
  assumes $\mathfrak{a} \in_\circ Vset\ \alpha$ **and** $\mathfrak{f} \in_\circ Vset\ \alpha$
  shows *semicategory* α (*smc*-1 $\mathfrak{a}$ $\mathfrak{f}$)
⟨*proof*⟩

4.11 *Rel* as a semicategory

4.11.1 Background

The methodology chosen for the exposition of *Rel* as a semicategory is analogous to the one used in the previous chapter for the exposition of *Rel* as a digraph. The general references for this section are Chapter I-7 in [39] and nLab [3][5].

named-theorems *smc-Rel-cs-simps*
named-theorems *smc-Rel-cs-intros*

lemmas (**in** *arr-Rel*) [*smc-Rel-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (**in** *arr-Rel*) [*smc-cs-intros*, *smc-Rel-cs-intros*] =
arr-Rel-axioms′

lemmas [*smc-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-v*11
*arr-Rel-comp-Rel-converse-Rel-right-if-v*11

lemmas [*smc-Rel-cs-intros*] =
dg-Rel-shared-cs-intros
arr-Rel-comp-Rel
arr-Rel.arr-Rel-converse-Rel

4.11.2 *Rel* as a semicategory

Definition and elementary properties

definition *smc-Rel* :: $V \Rightarrow V$
where *smc-Rel* α =
[
Vset α,
set $\{T.\ \textit{arr-Rel}\ \alpha\ T\}$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$,
$(\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$,
$(\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Rel}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
]$_{\circ}$

Components.

lemma *smc-Rel-components*:
shows *smc-Rel* $\alpha(\!|\textit{Obj}|\!)$ = *Vset* α
and *smc-Rel* $\alpha(\!|\textit{Arr}|\!)$ = *set* $\{T.\ \textit{arr-Rel}\ \alpha\ T\}$
and *smc-Rel* $\alpha(\!|\textit{Dom}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$
and *smc-Rel* $\alpha(\!|\textit{Cod}|\!) = (\lambda T \in_{\circ} \textit{set}\ \{T.\ \textit{arr-Rel}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$
and *smc-Rel* $\alpha(\!|\textit{Comp}|\!) = (\lambda ST \in_{\circ} \textit{composable-arrs}\ (\textit{dg-Rel}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

Slicing.

[5] https://ncatlab.org/nlab/show/Rel

lemma *smc-dg-smc-Rel*: *smc-dg* (*smc-Rel* α) = *dg-Rel* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-smc-Rel*, *unfolded slicing-simps*]:
smc-Rel-Obj-iff = *dg-Rel-Obj-iff*
and *smc-Rel-Arr-iff*[*smc-Rel-cs-simps*] = *dg-Rel-Arr-iff*
and *smc-Rel-Dom-vsv*[*smc-Rel-cs-intros*] = *dg-Rel-Dom-vsv*
and *smc-Rel-Dom-vdomain*[*smc-Rel-cs-simps*] = *dg-Rel-Dom-vdomain*
and *smc-Rel-Dom-app*[*smc-Rel-cs-simps*] = *dg-Rel-Dom-app*
and *smc-Rel-Dom-vrange* = *dg-Rel-Dom-vrange*
and *smc-Rel-Cod-vsv*[*smc-Rel-cs-intros*] = *dg-Rel-Cod-vsv*
and *smc-Rel-Cod-vdomain*[*smc-Rel-cs-simps*] = *dg-Rel-Cod-vdomain*
and *smc-Rel-Cod-app*[*smc-Rel-cs-simps*] = *dg-Rel-Cod-app*
and *smc-Rel-Cod-vrange* = *dg-Rel-Cod-vrange*
and *smc-Rel-is-arrI*[*smc-Rel-cs-intros*] = *dg-Rel-is-arrI*
and *smc-Rel-is-arrD* = *dg-Rel-is-arrD*
and *smc-Rel-is-arrE* = *dg-Rel-is-arrE*
and *smc-Rel-is-arr-ArrValE* = *dg-Rel-is-arr-ArrValE*

lemmas [*smc-cs-simps*] = *smc-Rel-is-arrD*(2,3)

lemmas-with (in $\mathcal{Z}$**)** [*folded smc-dg-smc-Rel*, *unfolded slicing-simps*]:
smc-Rel-Hom-vifunion-in-Vset = *dg-Rel-Hom-vifunion-in-Vset*
and *smc-Rel-incl-Rel-is-arr* = *dg-Rel-incl-Rel-is-arr*
and *smc-Rel-incl-Rel-is-arr′*[*smc-Rel-cs-intros*] = *dg-Rel-incl-Rel-is-arr′*

lemmas [*smc-Rel-cs-intros*] = $\mathcal{Z}$.*smc-Rel-incl-Rel-is-arr′*

Composable arrows

lemma *smc-Rel-composable-arrs-dg-Rel*:
composable-arrs (*dg-Rel* α) = *composable-arrs* (*smc-Rel* α)
⟨*proof*⟩

lemma *smc-Rel-Comp*:
smc-Rel $\alpha(\!|Comp|\!)$ = $(\lambda ST \in_{\circ} composable\text{-}arrs\ (smc\text{-}Rel\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

Composition

lemma *smc-Rel-Comp-app*[*smc-Rel-cs-simps*]:
assumes $S : b \mapsto_{smc\text{-}Rel\ \alpha} c$ **and** $T : a \mapsto_{smc\text{-}Rel\ \alpha} b$
shows $S \circ_{A\, smc\text{-}Rel\ \alpha} T = S \circ_{Rel} T$
⟨*proof*⟩

lemma *smc-Rel-Comp-vdomain*: $\mathcal{D}_{\circ}$ (*smc-Rel* $\alpha(\!|Comp|\!)$) = *composable-arrs* (*smc-Rel* α)
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Rel-Comp-vrange*: $\mathcal{R}_{\circ}$ (*smc-Rel* $\alpha(\!|Comp|\!)$) $\subseteq_{\circ}$ *set* $\{T.\ arr\text{-}Rel\ \alpha\ T\}$
⟨*proof*⟩

Rel is a semicategory

lemma (in $\mathcal{Z}$**)** *semicategory-smc-Rel*: *semicategory* α (*smc-Rel* α)
⟨*proof*⟩

4.11.3 Canonical dagger for *Rel*

Definition and elementary properties

definition *smcf-dag-Rel* :: $V \Rightarrow V$ ($\langle\dagger_{SMC.Rel}\rangle$)
where $\dagger_{SMC.Rel}\ \alpha =$
[
vid-on (*smc-Rel* $\alpha(\!|Obj|\!)$),
VLambda (*smc-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*,
op-smc (*smc-Rel* α),
smc-Rel α
]$_\circ$

Components.

lemma *smcf-dag-Rel-components*:
shows $\dagger_{SMC.Rel}\ \alpha(\!|ObjMap|\!) =$ *vid-on* (*smc-Rel* $\alpha(\!|Obj|\!)$)
and $\dagger_{SMC.Rel}\ \alpha(\!|ArrMap|\!) =$ *VLambda* (*smc-Rel* $\alpha(\!|Arr|\!)$) *converse-Rel*
and $\dagger_{SMC.Rel}\ \alpha(\!|HomDom|\!) =$ *op-smc* (*smc-Rel* α)
and $\dagger_{SMC.Rel}\ \alpha(\!|HomCod|\!) =$ *smc-Rel* α
⟨*proof*⟩

Slicing.

lemma *smcf-dghm-smcf-dag-Rel*: *smcf-dghm* ($\dagger_{SMC.Rel}\ \alpha$) = $\dagger_{DG.Rel}\ \alpha$
⟨*proof*⟩

lemmas-with [
folded smc-dg-smc-Rel smcf-dghm-smcf-dag-Rel, *unfolded slicing-simps*
]:
smcf-dag-Rel-ObjMap-vsv[*smc-Rel-cs-intros*] = *dghm-dag-Rel-ObjMap-vsv*
and *smcf-dag-Rel-ObjMap-vdomain*[*smc-Rel-cs-simps*] =
dghm-dag-Rel-ObjMap-vdomain
and *smcf-dag-Rel-ObjMap-app*[*smc-Rel-cs-simps*] = *dghm-dag-Rel-ObjMap-app*
and *smcf-dag-Rel-ObjMap-vrange*[*smc-Rel-cs-simps*] = *dghm-dag-Rel-ObjMap-vrange*
and *smcf-dag-Rel-ArrMap-vsv*[*smc-Rel-cs-intros*] = *dghm-dag-Rel-ArrMap-vsv*
and *smcf-dag-Rel-ArrMap-vdomain*[*smc-Rel-cs-simps*] =
dghm-dag-Rel-ArrMap-vdomain
and *smcf-dag-Rel-ArrMap-app*[*smc-Rel-cs-simps*] = *dghm-dag-Rel-ArrMap-app*
and *smcf-dag-Rel-ArrMap-app-vdomain*[*smc-cs-simps*] =
dghm-dag-Rel-ArrMap-app-vdomain
and *smcf-dag-Rel-ArrMap-app-vrange*[*smc-cs-simps*] =
dghm-dag-Rel-ArrMap-app-vrange
and *smcf-dag-Rel-ArrMap-vrange*[*smc-Rel-cs-simps*] = *dghm-dag-Rel-ArrMap-vrange*
and *smcf-dag-Rel-ArrMap-app-iff*[*smc-cs-simps*] = *dghm-dag-Rel-ArrMap-app-iff*
and *smcf-dag-Rel-app-is-arr* = *dghm-dag-Rel-ArrMap-app-is-arr*

Canonical dagger is a contravariant isomorphism of *Rel*

lemma (in $\mathcal{Z}$**)** *smcf-dag-Rel-is-iso-semifunctor*:
$\dagger_{SMC.Rel}\ \alpha$: *op-smc* (*smc-Rel* α) $\mapsto\mapsto_{SMC.iso\alpha}$ *smc-Rel* α
⟨*proof*⟩

Further properties of the canonical dagger

lemma (in $\mathcal{Z}$**)** *smcf-cn-comp-smcf-dag-Rel-smcf-dag-Rel*:
$\dagger_{SMC.Rel}\ \alpha\ _{SMCF\circ}\ \dagger_{SMC.Rel}\ \alpha$ = *smcf-id* (*smc-Rel* α)
⟨*proof*⟩

lemma *smcf-dag-Rel-ArrMap-smc-Rel-Comp*:
assumes $S : b \mapsto_{smc\text{-}Rel\ \alpha} c$ **and** $T : a \mapsto_{smc\text{-}Rel\ \alpha} b$

shows $\dagger_{SMC.Rel}\ \alpha(\!|ArrMap|\!)(\!|S \circ_{A\,smc\text{-}Rel\ \alpha}\ T|\!) =$
$\dagger_{SMC.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!) \circ_{A\,smc\text{-}Rel\ \alpha}\ \dagger_{SMC.Rel}\ \alpha(\!|ArrMap|\!)(\!|S|\!)$
⟨*proof*⟩

4.11.4 Monic arrow and epic arrow

The conditions for an arrow of *Rel* to be either monic or epic are outlined in nLab [3][6].

context
begin

private lemma *smc-Rel-is-monic-arr-vsubset*:
assumes $T : A \mapsto_{smc\text{-}Rel\ \alpha} B$
and $R : A' \mapsto_{smc\text{-}Rel\ \alpha} A$
and $S : A' \mapsto_{smc\text{-}Rel\ \alpha} A$
and $T \circ_{A\,smc\text{-}Rel\ \alpha} R = T \circ_{A\,smc\text{-}Rel\ \alpha} S$
and $\bigwedge y\ z\ X.$
$[\![\ y \subseteq_\circ A;\ z \subseteq_\circ A;\ T(\!|ArrVal|\!)\ `_\circ\ y = X;\ T(\!|ArrVal|\!)\ `_\circ\ z = X\ ]\!] \Longrightarrow y = z$
shows $R(\!|ArrVal|\!) \subseteq_\circ S(\!|ArrVal|\!)$
⟨*proof*⟩

lemma *smc-Rel-is-monic-arrI*:
assumes $T : A \mapsto_{smc\text{-}Rel\ \alpha} B$
and $\bigwedge y\ z\ X.\ [\![\ y \subseteq_\circ A;\ z \subseteq_\circ A;\ T(\!|ArrVal|\!)\ `_\circ\ y = X;\ T(\!|ArrVal|\!)\ `_\circ\ z = X\ ]\!] \Longrightarrow$
$y = z$
shows $T : A \mapsto_{mon\,smc\text{-}Rel\ \alpha} B$
⟨*proof*⟩

end

lemma *smc-Rel-is-monic-arrD*[*dest*]:
assumes $T : A \mapsto_{mon\,smc\text{-}Rel\ \alpha} B$
and $y \subseteq_\circ A$
and $z \subseteq_\circ A$
and $T(\!|ArrVal|\!)\ `_\circ\ y = X$
and $T(\!|ArrVal|\!)\ `_\circ\ z = X$
shows $y = z$
⟨*proof*⟩

lemma *smc-Rel-is-monic-arr*:
$T : A \mapsto_{mon\,smc\text{-}Rel\ \alpha} B \longleftrightarrow$
$T : A \mapsto_{smc\text{-}Rel\ \alpha} B \wedge$
(
$\forall\ y\ z\ X.$
$y \subseteq_\circ A \longrightarrow$
$z \subseteq_\circ A \longrightarrow$
$(T(\!|ArrVal|\!))\ `_\circ\ y = X \longrightarrow$
$(T(\!|ArrVal|\!))\ `_\circ\ z = X \longrightarrow$
$y = z$
)
⟨*proof*⟩

lemma *smc-Rel-is-monic-arr-is-epic-arr*:
assumes $T : A \mapsto_{smc\text{-}Rel\ \alpha} B$
and $(\dagger_{SMC.Rel}\ \alpha)(\!|ArrMap|\!)(\!|T|\!) : B \mapsto_{mon\,smc\text{-}Rel\ \alpha} A$
shows $T : A \mapsto_{epi\,smc\text{-}Rel\ \alpha} B$
⟨*proof*⟩

[6] https://ncatlab.org/nlab/show/Rel

lemma *smc-Rel-is-epic-arr-is-monic-arr*:
assumes $T : A \mapsto_{epi smc\text{-}Rel\ \alpha} B$
shows $\dagger_{SMC.Rel}\ \alpha(\!|ArrMap|\!)(\!|T|\!) : B \mapsto_{mon smc\text{-}Rel\ \alpha} A$
⟨*proof*⟩

lemma *smc-Rel-is-epic-arrI*:
assumes $T : A \mapsto_{smc\text{-}Rel\ \alpha} B$
and $\bigwedge y\ z\ X.\ [\![\ y \subseteq_\circ B;\ z \subseteq_\circ B;\ T(\!|ArrVal|\!) -`_\circ y = X;\ T(\!|ArrVal|\!) -`_\circ z = X\ ]\!] \Longrightarrow y = z$
shows $T : A \mapsto_{epi smc\text{-}Rel\ \alpha} B$
⟨*proof*⟩

lemma *smc-Rel-is-epic-arrD*[*dest*]:
assumes $T : A \mapsto_{epi smc\text{-}Rel\ \alpha} B$
and $y \subseteq_\circ B$
and $z \subseteq_\circ B$
and $T(\!|ArrVal|\!) -`_\circ y = X$
and $T(\!|ArrVal|\!) -`_\circ z = X$
shows $y = z$
⟨*proof*⟩

lemma *smc-Rel-is-epic-arr*:
$T : A \mapsto_{epi smc\text{-}Rel\ \alpha} B \longleftrightarrow$
$T : A \mapsto_{smc\text{-}Rel\ \alpha} B \wedge$
(
$\forall\ y\ z\ X.$
$y \subseteq_\circ B \longrightarrow$
$z \subseteq_\circ B \longrightarrow$
$T(\!|ArrVal|\!) -`_\circ y = X \longrightarrow$
$T(\!|ArrVal|\!) -`_\circ z = X \longrightarrow$
$y = z$
)
⟨*proof*⟩

4.11.5 Terminal object, initial object and null object

An object in the semicategory *Rel* is terminal/initial/null if and only if it is the empty set (see nLab [3])[7].

lemma (in $\mathcal{Z}$**)** *smc-Rel-obj-terminal*: *obj-terminal* (*smc-Rel* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Rel-obj-initial*: *obj-initial* (*smc-Rel* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Rel-obj-terminal-obj-initial*:
obj-initial (*smc-Rel* α) $A \longleftrightarrow$ *obj-terminal* (*smc-Rel* α) A
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Rel-obj-null*: *obj-null* (*smc-Rel* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

[7] https://ncatlab.org/nlab/show/database+of+categories

4.11.6 Zero arrow

A zero arrow for *Rel* is any admissible V-arrow, such that its value is the empty set. A reference for this result is not given, but the result is not expected to be original.

lemma (in $\mathcal{Z}$**)** *smc-Rel-is-zero-arr*:
assumes $A \in_{\circ}$ *smc-Rel* $\alpha(\!|Obj|\!)$ **and** $B \in_{\circ}$ *smc-Rel* $\alpha(\!|Obj|\!)$
shows $T : A \mapsto_{0 \, smc\text{-}Rel\ \alpha} B \longleftrightarrow T = [0, A, B]_{\circ}$
⟨*proof*⟩

4.12 *Par* as a semicategory

4.12.1 Background

The methodology chosen for the exposition of *Par* as a semicategory is analogous to the one used in the previous chapter for the exposition of *Par* as a digraph.

named-theorems *smc-Par-cs-simps*
named-theorems *smc-Par-cs-intros*

lemmas (in *arr-Par*) [*smc-Par-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (in *arr-Par*) [*smc-cs-intros*, *smc-Par-cs-intros*] =
arr-Par-axioms′

lemmas [*smc-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 [*smc-Par-cs-intros*] =
dg-Rel-shared-cs-intros
arr-Par-comp-Par

4.12.2 *Par* as a semicategory

Definition and elementary properties

definition *smc-Par* :: $V \Rightarrow V$
where *smc-Par* α =
[
Vset α,
set $\{T.$ *arr-Par* α $T\}$,
$(\lambda T \in_\circ set\ \{T.$ *arr-Par* α $T\}.$ $T(\!|ArrDom|\!))$,
$(\lambda T \in_\circ set\ \{T.$ *arr-Par* α $T\}.$ $T(\!|ArrCod|\!))$,
$(\lambda ST \in_\circ composable\text{-}arrs\ (dg\text{-}Par\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
]$_\circ$

Components.

lemma *smc-Par-components*:
shows *smc-Par* $\alpha(\!|Obj|\!)$ = *Vset* α
and *smc-Par* $\alpha(\!|Arr|\!)$ = *set* $\{T.$ *arr-Par* α $T\}$
and *smc-Par* $\alpha(\!|Dom|\!) = (\lambda T \in_\circ set\ \{T.$ *arr-Par* α $T\}.$ $T(\!|ArrDom|\!))$
and *smc-Par* $\alpha(\!|Cod|\!) = (\lambda T \in_\circ set\ \{T.$ *arr-Par* α $T\}.$ $T(\!|ArrCod|\!))$
and *smc-Par* $\alpha(\!|Comp|\!) = (\lambda ST \in_\circ composable\text{-}arrs\ (dg\text{-}Par\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

Slicing.

lemma *smc-dg-smc-Par*: *smc-dg* (*smc-Par* α) = *dg-Par* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-smc-Par*, *unfolded slicing-simps*]:
smc-Par-Obj-iff = *dg-Par-Obj-iff*
and *smc-Par-Arr-iff*[*smc-Par-cs-simps*] = *dg-Par-Arr-iff*
and *smc-Par-Dom-vsv*[*smc-Par-cs-intros*] = *dg-Par-Dom-vsv*
and *smc-Par-Dom-vdomain*[*smc-Par-cs-simps*] = *dg-Par-Dom-vdomain*
and *smc-Par-Dom-vrange* = *dg-Par-Dom-vrange*

and *smc-Par-Dom-app*[*smc-Par-cs-simps*] = *dg-Par-Dom-app*
and *smc-Par-Cod-vsv*[*smc-Par-cs-intros*] = *dg-Par-Cod-vsv*
and *smc-Par-Cod-vdomain*[*smc-Par-cs-simps*] = *dg-Par-Cod-vdomain*
and *smc-Par-Cod-vrange* = *dg-Par-Cod-vrange*
and *smc-Par-Cod-app*[*smc-Par-cs-simps*] = *dg-Par-Cod-app*
and *smc-Par-is-arrI* = *dg-Par-is-arrI*
and *smc-Par-is-arrD* = *dg-Par-is-arrD*
and *smc-Par-is-arrE* = *dg-Par-is-arrE*

lemmas [*smc-cs-simps*] = *smc-Par-is-arrD*(2,3)

lemmas [*smc-Par-cs-intros*] = *smc-Par-is-arrI*

lemmas-with (in $\mathcal{Z}$**)** [*folded smc-dg-smc-Par*, *unfolded slicing-simps*]:
smc-Par-Hom-vifunion-in-Vset = *dg-Par-Hom-vifunion-in-Vset*
and *smc-Par-incl-Par-is-arr* = *dg-Par-incl-Par-is-arr*
and *smc-Par-incl-Par-is-arr′*[*smc-Par-cs-intros*] = *dg-Par-incl-Par-is-arr′*

lemmas [*smc-Par-cs-intros*] = $\mathcal{Z}$.*smc-Par-incl-Par-is-arr′*

Composable arrows

lemma *smc-Par-composable-arrs-dg-Par*:
composable-arrs (*dg-Par* α) = *composable-arrs* (*smc-Par* α)
⟨*proof*⟩

lemma *smc-Par-Comp*:
smc-Par α(‖*Comp*‖) = ($\lambda ST \in_\circ$ *composable-arrs* (*smc-Par* α). $ST(\|0\|) \circ_{Rel} ST(\|1_{\mathbf{N}}\|)$)
⟨*proof*⟩

Composition

lemma *smc-Par-Comp-app*[*smc-Par-cs-simps*]:
assumes $S : B \mapsto_{\textit{smc-Par } \alpha} C$ **and** $T : A \mapsto_{\textit{smc-Par } \alpha} B$
shows $S \circ_{A\,\textit{smc-Par } \alpha} T = S \circ_{Rel} T$
⟨*proof*⟩

lemma *smc-Par-Comp-vdomain*: $\mathcal{D}_\circ$ (*smc-Par* α(‖*Comp*‖)) = *composable-arrs* (*smc-Par* α)
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Par-Comp-vrange*: $\mathcal{R}_\circ$ (*smc-Par* α(‖*Comp*‖)) $\subseteq_\circ$ *set* {*T*. *arr-Par* α *T*}
⟨*proof*⟩

Par is a semicategory

lemma (in $\mathcal{Z}$**)** *semicategory-smc-Par*: *semicategory* α (*smc-Par* α)
⟨*proof*⟩

Par is a wide subsemicategory of *Rel*

lemma (in $\mathcal{Z}$**)** *wide-subsemicategory-smc-Par-smc-Rel*:
smc-Par $\alpha \subseteq_{SMC.wide\alpha}$ *smc-Rel* α
⟨*proof*⟩

4.12.3 Monic arrow and epic arrow

lemma *smc-Par-is-monic-arrI*[*intro*]:
assumes $T : A \mapsto_{\textit{smc-Par } \alpha} B$ **and** *v*11 (T(‖*ArrVal*‖)) **and** $\mathcal{D}_\circ$ (T(‖*ArrVal*‖)) = A
shows $T : A \mapsto_{mon\,\textit{smc-Par } \alpha} B$

⟨*proof*⟩

lemma *smc-Par-is-monic-arrD*:
assumes $T : A \mapsto_{mon} \textit{smc-Par}\ \alpha\ B$
shows $T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B$ **and** $v11\ (T(\!|\textit{ArrVal}|\!))$ **and** $\mathcal{D}_\circ\ (T(\!|\textit{ArrVal}|\!)) = A$
⟨*proof*⟩

lemma *smc-Par-is-monic-arr*:
$T : A \mapsto_{mon} \textit{smc-Par}\ \alpha\ B \longleftrightarrow$
$T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B \wedge v11\ (T(\!|\textit{ArrVal}|\!)) \wedge \mathcal{D}_\circ\ (T(\!|\textit{ArrVal}|\!)) = A$
⟨*proof*⟩

context
begin

private lemma *smc-Par-is-epic-arr-vsubset*:
assumes $T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B$
and $\mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!)) = B$
and $R : B \mapsto_{\textit{smc-Par}\ \alpha}\ C$
and $S : B \mapsto_{\textit{smc-Par}\ \alpha}\ C$
and $R \circ_{A\textit{smc-Par}\ \alpha}\ T = S \circ_{A\textit{smc-Par}\ \alpha}\ T$
shows $R(\!|\textit{ArrVal}|\!) \subseteq_\circ S(\!|\textit{ArrVal}|\!)$
⟨*proof*⟩

lemma *smc-Par-is-epic-arrI*:
assumes $T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B$ **and** $\mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!)) = B$
shows $T : A \mapsto_{epi\textit{smc-Par}\ \alpha}\ B$
⟨*proof*⟩

lemma *smc-Par-is-epic-arrD*:
assumes $T : A \mapsto_{epi\textit{smc-Par}\ \alpha}\ B$
shows $T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B$ **and** $\mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!)) = B$
⟨*proof*⟩

end

lemma *smc-Par-is-epic-arr*:
$T : A \mapsto_{epi\textit{smc-Par}\ \alpha}\ B \longleftrightarrow T : A \mapsto_{\textit{smc-Par}\ \alpha}\ B \wedge \mathcal{R}_\circ\ (T(\!|\textit{ArrVal}|\!)) = B$
⟨*proof*⟩

4.12.4 Terminal object, initial object and null object

lemma (in $\mathcal{Z}$**)** *smc-Par-obj-terminal*: *obj-terminal* (*smc-Par* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Par-obj-initial*: *obj-initial* (*smc-Par* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Par-obj-terminal-obj-initial*:
obj-initial (*smc-Par* α) $A \longleftrightarrow$ *obj-terminal* (*smc-Par* α) A
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Par-obj-null*: *obj-null* (*smc-Par* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

4.12.5 Zero arrow

lemma (in $\mathcal{Z}$**)** *smc-Par-is-zero-arr*:

assumes $A \in_{\circ} \textit{smc-Par}\ \alpha(\!|\textit{Obj}|\!)$ **and** $B \in_{\circ} \textit{smc-Par}\ \alpha(\!|\textit{Obj}|\!)$
shows $T : A \mapsto_{0\textit{smc-Par}\ \alpha} B \longleftrightarrow T = [0,\ A,\ B]_{\circ}$
⟨*proof*⟩

4.13 *Set* as a semicategory

4.13.1 Background

The methodology chosen for the exposition of *Set* as a semicategory is analogous to the one used in the previous chapter for the exposition of *Set* as a digraph.

named-theorems *smc-Set-cs-simps*
named-theorems *smc-Set-cs-intros*

lemmas (in *arr-Set*) [*smc-Set-cs-simps*] =
dg-Rel-shared-cs-simps

lemmas (in *arr-Set*) [*smc-cs-intros*, *smc-Set-cs-intros*] =
arr-Set-axioms'

lemmas [*smc-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 [*smc-Set-cs-intros*] =
dg-Rel-shared-cs-intros
arr-Set-comp-Set

4.13.2 *Set* as a semicategory

Definition and elementary properties

definition *smc-Set* :: $V \Rightarrow V$
where *smc-Set* α =
[
Vset α,
set $\{T.\ \textit{arr-Set}\ \alpha\ T\}$,
$(\lambda T \in_\circ \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$,
$(\lambda T \in_\circ \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$,
$(\lambda ST \in_\circ \textit{composable-arrs}\ (\textit{dg-Set}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
]$_\circ$

Components.

lemma *smc-Set-components*:
shows *smc-Set* $\alpha(\!|\textit{Obj}|\!)$ = *Vset* α
and *smc-Set* $\alpha(\!|\textit{Arr}|\!) = \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}$
and *smc-Set* $\alpha(\!|\textit{Dom}|\!) = (\lambda T \in_\circ \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrDom}|\!))$
and *smc-Set* $\alpha(\!|\textit{Cod}|\!) = (\lambda T \in_\circ \textit{set}\ \{T.\ \textit{arr-Set}\ \alpha\ T\}.\ T(\!|\textit{ArrCod}|\!))$
and *smc-Set* $\alpha(\!|\textit{Comp}|\!) = (\lambda ST \in_\circ \textit{composable-arrs}\ (\textit{dg-Set}\ \alpha).\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

Slicing.

lemma *smc-dg-smc-Set*: *smc-dg* (*smc-Set* α) = *dg-Set* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-smc-Set*, *unfolded slicing-simps*]:
smc-Set-Obj-iff = *dg-Set-Obj-iff*
and *smc-Set-Arr-iff*[*smc-Set-cs-simps*] = *dg-Set-Arr-iff*
and *smc-Set-Dom-vsv*[*smc-Set-cs-intros*] = *dg-Set-Dom-vsv*
and *smc-Set-Dom-vdomain*[*smc-Set-cs-simps*] = *dg-Set-Dom-vdomain*
and *smc-Set-Dom-vrange* = *dg-Set-Dom-vrange*

and *smc-Set-Dom-app*[*smc-Set-cs-simps*] = *dg-Set-Dom-app*
and *smc-Set-Cod-vsv*[*smc-Set-cs-intros*] = *dg-Set-Cod-vsv*
and *smc-Set-Cod-vdomain*[*smc-Set-cs-simps*] = *dg-Set-Cod-vdomain*
and *smc-Set-Cod-vrange* = *dg-Set-Cod-vrange*
and *smc-Set-Cod-app*[*smc-Set-cs-simps*] = *dg-Set-Cod-app*
and *smc-Set-is-arrI* = *dg-Set-is-arrI*
and *smc-Set-is-arrD* = *dg-Set-is-arrD*
and *smc-Set-is-arrE* = *dg-Set-is-arrE*
and *smc-Set-ArrVal-vdomain*[*smc-Set-cs-simps*] = *dg-Set-ArrVal-vdomain*
and *smc-Set-ArrVal-app-vrange*[*smc-Set-cs-intros*] = *dg-Set-ArrVal-app-vrange*

lemmas [*smc-cs-simps*] = *smc-Set-is-arrD*(2,3)

lemmas-with (in $\mathcal{Z}$**)** [*folded smc-dg-smc-Set*, *unfolded slicing-simps*]:
smc-Set-Hom-vifunion-in-Vset = *dg-Set-Hom-vifunion-in-Vset*
and *smc-Set-incl-Set-is-arr* = *dg-Set-incl-Set-is-arr*

lemmas [*smc-Set-cs-intros*] =
smc-Set-is-arrI

lemma (in $\mathcal{Z}$**)** *smc-Set-incl-Set-is-arr'*[*smc-cs-intros*, *smc-Set-cs-intros*]:
assumes $A \in_{\circ}$ *smc-Set* $\alpha(\!|Obj|\!)$
and $B \in_{\circ}$ *smc-Set* $\alpha(\!|Obj|\!)$
and $A \subseteq_{\circ} B$
and $A' = A$
and $B' = B$
and $\mathfrak{C}' =$ *smc-Set* α
shows *incl-Set* $A\ B : A' \mapsto_{\mathfrak{C}'} B'$
⟨*proof*⟩

lemmas [*smc-Set-cs-intros*] = $\mathcal{Z}$.*smc-Set-incl-Set-is-arr'*

Composable arrows

lemma *smc-Set-composable-arrs-dg-Set*:
composable-arrs (*dg-Set* α) = *composable-arrs* (*smc-Set* α)
⟨*proof*⟩

lemma *smc-Set-Comp*:
smc-Set $\alpha(\!|Comp|\!)$ =
VLambda (*composable-arrs* (*smc-Set* α)) ($\lambda ST.\ ST(\!|0|\!) \circ_{Rel} ST(\!|1_{\mathbf{N}}|\!)$)
⟨*proof*⟩

Composition

lemma *smc-Set-Comp-app*[*smc-Set-cs-simps*]:
assumes $S : b \mapsto_{smc\text{-}Set\ \alpha} c$ **and** $T : a \mapsto_{smc\text{-}Set\ \alpha} b$
shows $S \circ_{A\, smc\text{-}Set\ \alpha} T = S \circ_{Set} T$
⟨*proof*⟩

lemma *smc-Set-Comp-vdomain*: $\mathcal{D}_{\circ}$ (*smc-Set* $\alpha(\!|Comp|\!)$) = *composable-arrs* (*smc-Set* α)
⟨*proof*⟩

lemma (in $\mathcal{Z}$**)** *smc-Set-Comp-vrange*:
$\mathcal{R}_{\circ}$ (*smc-Set* $\alpha(\!|Comp|\!)$) $\subseteq_{\circ}$ *set* $\{T.$ *arr-Set* $\alpha\ T\}$
⟨*proof*⟩

lemma *smc-Set-composable-vrange-vdomain*[*smc-Set-cs-intros*]:

assumes $g : b \mapsto_{smc\text{-}Set\ \alpha} c$ **and** $f : a \mapsto_{smc\text{-}Set\ \alpha} b$
shows $\mathcal{R}_\circ\ (f(\!|ArrVal|\!)) \subseteq_\circ \mathcal{D}_\circ\ (g(\!|ArrVal|\!))$
⟨*proof*⟩

lemma *smc-Set-Comp-ArrVal*[*smc-cs-simps*]:
assumes $S : y \mapsto_{smc\text{-}Set\ \alpha} z$ **and** $T : x \mapsto_{smc\text{-}Set\ \alpha} y$ **and** $a \in_\circ x$
shows $(S \circ_{A\ smc\text{-}Set\ \alpha} T)(\!|ArrVal|\!)(\!|a|\!) = S(\!|ArrVal|\!)(\!|T(\!|ArrVal|\!)(\!|a|\!)|\!)$
⟨*proof*⟩

Set is a semicategory

lemma (in $\mathcal{Z}$**)** *semicategory-smc-Set*: *semicategory* α (*smc-Set* α)
⟨*proof*⟩

Set is a wide subsemicategory of *Par*

lemma (in $\mathcal{Z}$**)** *wide-subsemicategory-smc-Set-smc-Par*:
smc-Set $\alpha \subseteq_{SMC.wide\alpha}$ *smc-Par* α
⟨*proof*⟩

4.13.3 Monic arrow and epic arrow

lemma *smc-Set-is-monic-arrI*:
— See Chapter I-5 in [39]).
assumes $T : A \mapsto_{smc\text{-}Set\ \alpha} B$ **and** $v11\ (T(\!|ArrVal|\!))$ **and** $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
shows $T : A \mapsto_{mon smc\text{-}Set\ \alpha} B$
⟨*proof*⟩

lemma *smc-Set-is-monic-arrD*:
assumes $T : A \mapsto_{mon smc\text{-}Set\ \alpha} B$
shows $T : A \mapsto_{smc\text{-}Set\ \alpha} B$ **and** $v11\ (T(\!|ArrVal|\!))$ **and** $\mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
⟨*proof*⟩

lemma *smc-Set-is-monic-arr*:
$T : A \mapsto_{mon smc\text{-}Set\ \alpha} B \longleftrightarrow$
$T : A \mapsto_{smc\text{-}Set\ \alpha} B \wedge v11\ (T(\!|ArrVal|\!)) \wedge \mathcal{D}_\circ\ (T(\!|ArrVal|\!)) = A$
⟨*proof*⟩

An epic arrow in *Set* is a total surjective function (see Chapter I-5 in [39]).

lemma *smc-Set-is-epic-arrI*:
assumes $T : A \mapsto_{smc\text{-}Set\ \alpha} B$ **and** $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
shows $T : A \mapsto_{epi smc\text{-}Set\ \alpha} B$
⟨*proof*⟩

lemma *smc-Set-is-epic-arrD*:
assumes $T : A \mapsto_{epi smc\text{-}Set\ \alpha} B$
shows $T : A \mapsto_{smc\text{-}Set\ \alpha} B$ **and** $\mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

lemma *smc-Set-is-epic-arr*:
$T : A \mapsto_{epi smc\text{-}Set\ \alpha} B \longleftrightarrow T : A \mapsto_{smc\text{-}Set\ \alpha} B \wedge \mathcal{R}_\circ\ (T(\!|ArrVal|\!)) = B$
⟨*proof*⟩

4.13.4 Terminal object, initial object and null object

An object in *Set* is terminal if and only if it is a singleton set (see Chapter I-5 in [39]).

lemma (in $\mathcal{Z}$**)** *smc-Set-obj-terminal*:
obj-terminal (*smc-Set* α) $A \longleftrightarrow (\exists B \in_\circ Vset\ \alpha.\ A = set\ \{B\})$

⟨*proof*⟩

An object in *Set* is initial if and only if it is the empty set (see Chapter I-5 in [39]).

lemma (in $\mathcal{Z}$**)** *smc-Set-obj-initial*: *obj-initial* (*smc-Set* α) $A \longleftrightarrow A = 0$
⟨*proof*⟩

There are no null objects in *Set* (this is a trivial corollary of the above).

lemma (in $\mathcal{Z}$**)** *smc-Set-obj-null*: *obj-null* (*smc-Set* α) $A \longleftrightarrow$ *False*
⟨*proof*⟩

4.13.5 Zero arrow

There are no zero arrows in *Set* (this result is a trivial corollary of the absence of null objects).

lemma (in $\mathcal{Z}$**)** *smc-Set-is-zero-arr*: $T : A \mapsto_{0 smc\text{-}Set\ \alpha} B \longleftrightarrow$ *False*
⟨*proof*⟩

4.14 *GRPH* as a semicategory

4.14.1 Background

The methodology for the exposition of *GRPH* as a semicategory is analogous to the one used in the previous chapter for the exposition of *GRPH* as a digraph.

named-theorems *smc-GRPH-cs-simps*
named-theorems *smc-GRPH-cs-intros*

4.14.2 Definition and elementary properties

definition *smc-GRPH* :: $V \Rightarrow V$
where *smc-GRPH* α =
[
set $\{\mathfrak{C}.$ *digraph* $\alpha\ \mathfrak{C}\}$,
all-dghms α,
$(\lambda\mathfrak{F}\in_\circ$*all-dghms* $\alpha.\ \mathfrak{F}(\!|$*HomDom*$|\!))$,
$(\lambda\mathfrak{F}\in_\circ$*all-dghms* $\alpha.\ \mathfrak{F}(\!|$*HomCod*$|\!))$,
$(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-GRPH* α). $\mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
]$_\circ$

Components.

lemma *smc-GRPH-components*:
shows *smc-GRPH* $\alpha(\!|$*Obj*$|\!)$ = *set* $\{\mathfrak{C}.$ *digraph* $\alpha\ \mathfrak{C}\}$
and *smc-GRPH* $\alpha(\!|$*Arr*$|\!)$ = *all-dghms* α
and *smc-GRPH* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-dghms* $\alpha.\ \mathfrak{F}(\!|$*HomDom*$|\!))$
and *smc-GRPH* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-dghms* $\alpha.\ \mathfrak{F}(\!|$*HomCod*$|\!))$
and *smc-GRPH* $\alpha(\!|$*Comp*$|\!)$ =
$(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-GRPH* α). $\mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

Slicing.

lemma *smc-dg-GRPH*: *smc-dg* (*smc-GRPH* α) = *dg-GRPH* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-GRPH*, *unfolded slicing-simps*]:
smc-GRPH-ObjI = *dg-GRPH-ObjI*
and *smc-GRPH-ObjD* = *dg-GRPH-ObjD*
and *smc-GRPH-ObjE* = *dg-GRPH-ObjE*
and *smc-GRPH-Obj-iff*[*smc-GRPH-cs-simps*] = *dg-GRPH-Obj-iff*
and *smc-GRPH-Dom-app*[*smc-GRPH-cs-simps*] = *dg-GRPH-Dom-app*
and *smc-GRPH-Cod-app*[*smc-GRPH-cs-simps*] = *dg-GRPH-Cod-app*
and *smc-GRPH-is-arrI* = *dg-GRPH-is-arrI*
and *smc-GRPH-is-arrD* = *dg-GRPH-is-arrD*
and *smc-GRPH-is-arrE* = *dg-GRPH-is-arrE*
and *smc-GRPH-is-arr-iff*[*smc-GRPH-cs-simps*] = *dg-GRPH-is-arr-iff*

4.14.3 Composable arrows

lemma *smc-GRPH-composable-arrs-dg-GRPH*:
composable-arrs (*dg-GRPH* α) = *composable-arrs* (*smc-GRPH* α)
⟨*proof*⟩

lemma *smc-GRPH-Comp*:
smc-GRPH $\alpha(\!|$*Comp*$|\!)$ = $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*smc-GRPH* α). $\mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathbb{N}}|\!))$
⟨*proof*⟩

4.14.4 Composition

lemma *smc-GRPH-Comp-app*:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto_{smc\text{-}GRPH\ \alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto_{smc\text{-}GRPH\ \alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{A\,smc\text{-}GRPH\ \alpha} \mathfrak{F} = \mathfrak{G} \circ_{DGHM} \mathfrak{F}$
⟨*proof*⟩

lemma *smc-GRPH-Comp-vdomain*:
$\mathcal{D}_\circ$ (*smc-GRPH* α(|*Comp*|)) = *composable-arrs* (*smc-GRPH* α)
⟨*proof*⟩

4.14.5 *GRPH* is a semicategory

lemma (**in** $\mathcal{Z}$) *tiny-semicategory-smc-GRPH*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-semicategory* β (*smc-GRPH* α)
⟨*proof*⟩

4.14.6 Initial object

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-initialI*: *obj-initial* (*smc-GRPH* α) *dg*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-initialD*:
assumes *obj-initial* (*smc-GRPH* α) $\mathfrak{A}$
shows $\mathfrak{A}$ = *dg*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-initialE*:
assumes *obj-initial* (*smc-GRPH* α) $\mathfrak{A}$
obtains $\mathfrak{A}$ = *dg*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-initial-iff*[*smc-GRPH-cs-simps*]:
obj-initial (*smc-GRPH* α) $\mathfrak{A} \longleftrightarrow \mathfrak{A}$ = *dg*-0
⟨*proof*⟩

4.14.7 Terminal object

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-terminalI*[*smc-GRPH-cs-intros*]:
assumes $a \in_\circ$ *Vset* α **and** $f \in_\circ$ *Vset* α
shows *obj-terminal* (*smc-GRPH* α) (*dg*-1 a f)
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-GRPH-obj-terminalE*:
assumes *obj-terminal* (*smc-GRPH* α) $\mathfrak{B}$
obtains a f **where** $a \in_\circ$ *Vset* α **and** $f \in_\circ$ *Vset* α **and** $\mathfrak{B}$ = *dg*-1 a f
⟨*proof*⟩

4.15 *SemiCAT* as a digraph

4.15.1 Background

SemiCAT is usually defined as a category of semicategories and semifunctors (e.g., see [3][8]). However, there is little that can prevent one from exposing *SemiCAT* as a digraph and provide additional structure gradually in subsequent theories. Thus, in this section, α-*SemiCAT* is defined as a digraph of semicategories and semifunctors in V_α.

named-theorems *dg-SemiCAT-simps*
named-theorems *dg-SemiCAT-intros*

4.15.2 Definition and elementary properties

definition *dg-SemiCAT* :: $V \Rightarrow V$
  where *dg-SemiCAT* α =
    [
      set $\{\mathfrak{C}.$ *semicategory* α $\mathfrak{C}\}$,
      all-smcfs α,
      $(\lambda\mathfrak{F}\in_\circ$ *all-smcfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomDom*$\rfloor))$,
      $(\lambda\mathfrak{F}\in_\circ$ *all-smcfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomCod*$\rfloor))$
    ]$_\circ$

Components.

lemma *dg-SemiCAT-components*:
  shows *dg-SemiCAT* $\alpha(\lfloor$*Obj*$\rfloor)$ = *set* $\{\mathfrak{C}.$ *semicategory* α $\mathfrak{C}\}$
    and *dg-SemiCAT* $\alpha(\lfloor$*Arr*$\rfloor)$ = *all-smcfs* α
    and *dg-SemiCAT* $\alpha(\lfloor$*Dom*$\rfloor)$ = $(\lambda\mathfrak{F}\in_\circ$ *all-smcfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomDom*$\rfloor))$
    and *dg-SemiCAT* $\alpha(\lfloor$*Cod*$\rfloor)$ = $(\lambda\mathfrak{F}\in_\circ$ *all-smcfs* $\alpha.$ $\mathfrak{F}(\lfloor$*HomCod*$\rfloor))$
  ⟨*proof*⟩

4.15.3 Object

lemma *dg-SemiCAT-ObjI*:
  assumes *semicategory* α $\mathfrak{A}$
  shows $\mathfrak{A} \in_\circ$ *dg-SemiCAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  ⟨*proof*⟩

lemma *dg-SemiCAT-ObjD*:
  assumes $\mathfrak{A} \in_\circ$ *dg-SemiCAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  shows *semicategory* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-SemiCAT-ObjE*:
  assumes $\mathfrak{A} \in_\circ$ *dg-SemiCAT* $\alpha(\lfloor$*Obj*$\rfloor)$
  obtains *semicategory* α $\mathfrak{A}$
  ⟨*proof*⟩

lemma *dg-SemiCAT-Obj-iff*[*dg-SemiCAT-simps*]:
  $\mathfrak{A} \in_\circ$ *dg-SemiCAT* $\alpha(\lfloor$*Obj*$\rfloor) \longleftrightarrow$ *semicategory* α $\mathfrak{A}$
  ⟨*proof*⟩

4.15.4 Domain and codomain

lemma [*dg-SemiCAT-simps*]:
  assumes $\mathfrak{F} \in_\circ$ *all-smcfs* α
  shows *dg-SemiCAT-Dom-app*: *dg-SemiCAT* $\alpha(\lfloor$*Dom*$\rfloor)(\lfloor\mathfrak{F}\rfloor)$ = $\mathfrak{F}(\lfloor$*HomDom*$\rfloor)$

[8] https://ncatlab.org/nlab/show/semicategory

and *dg-SemiCAT-Cod-app*: *dg-SemiCAT* $\alpha(\!|Cod|\!)(\!|\mathfrak{F}|\!) = \mathfrak{F}(\!|HomCod|\!)$
⟨*proof*⟩

4.15.5 *SemiCAT* is a digraph

lemma (in $\mathcal{Z}$**)** *tiny-digraph-dg-SemiCAT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_{\circ} \beta$
shows *tiny-digraph* β (*dg-SemiCAT* α)
⟨*proof*⟩

4.15.6 Arrow with a domain and a codomain

lemma *dg-SemiCAT-is-arrI*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}SemiCAT\ \alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-SemiCAT-is-arrD*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}SemiCAT\ \alpha} \mathfrak{B}$
shows $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-SemiCAT-is-arrE*:
assumes $\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}SemiCAT\ \alpha} \mathfrak{B}$
obtains $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

lemma *dg-SemiCAT-is-arr-iff*[*dg-SemiCAT-simps*]:
$\mathfrak{F} : \mathfrak{A} \mapsto_{dg\text{-}SemiCAT\ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{SMC\alpha} \mathfrak{B}$
⟨*proof*⟩

4.16 *SemiCAT* as a semicategory

4.16.1 Background

The subsection presents the theory of the semicategories of α-semicategories. It continues the development that was initiated in section 4.15.

named-theorems *smc-SemiCAT-simps*
named-theorems *smc-SemiCAT-intros*

4.16.2 Definition and elementary properties

definition *smc-SemiCAT* :: $V \Rightarrow V$
  where *smc-SemiCAT* α =
    [
      set $\{\mathfrak{C}.$ *semicategory* α $\mathfrak{C}\}$,
      all-smcfs α,
      $(\lambda\mathfrak{F}\in_\circ$*all-smcfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$,
      $(\lambda\mathfrak{F}\in_\circ$*all-smcfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$,
      $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-SemiCAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{SMCF} \mathfrak{GF}(\!|1_{\mathfrak{N}}|\!))$
    ]$_\circ$

Components.

lemma *smc-SemiCAT-components*:
  shows *smc-SemiCAT* $\alpha(\!|$*Obj*$|\!)$ = *set* $\{\mathfrak{C}.$ *semicategory* α $\mathfrak{C}\}$
    and *smc-SemiCAT* $\alpha(\!|$*Arr*$|\!)$ = *all-smcfs* α
    and *smc-SemiCAT* $\alpha(\!|$*Dom*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-smcfs* $\alpha.$ $\mathfrak{F}(\!|$*HomDom*$|\!))$
    and *smc-SemiCAT* $\alpha(\!|$*Cod*$|\!)$ = $(\lambda\mathfrak{F}\in_\circ$*all-smcfs* $\alpha.$ $\mathfrak{F}(\!|$*HomCod*$|\!))$
    and *smc-SemiCAT* $\alpha(\!|$*Comp*$|\!)$ =
      $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*dg-SemiCAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{SMCF} \mathfrak{GF}(\!|1_{\mathfrak{N}}|\!))$
  ⟨*proof*⟩

Slicing.

lemma *smc-dg-SemiCAT*[*smc-SemiCAT-simps*]: *smc-dg* (*smc-SemiCAT* α) = *dg-SemiCAT* α
⟨*proof*⟩

lemmas-with [*folded smc-dg-SemiCAT*, *unfolded slicing-simps*]:
  smc-SemiCAT-ObjI = *dg-SemiCAT-ObjI*
  and *smc-SemiCAT-ObjD* = *dg-SemiCAT-ObjD*
  and *smc-SemiCAT-ObjE* = *dg-SemiCAT-ObjE*
  and *smc-SemiCAT-Obj-iff*[*smc-SemiCAT-simps*] = *dg-SemiCAT-Obj-iff*
  and *smc-SemiCAT-Dom-app*[*smc-SemiCAT-simps*] = *dg-SemiCAT-Dom-app*
  and *smc-SemiCAT-Cod-app*[*smc-SemiCAT-simps*] = *dg-SemiCAT-Cod-app*
  and *smc-SemiCAT-is-arrI* = *dg-SemiCAT-is-arrI*
  and *smc-SemiCAT-is-arrD* = *dg-SemiCAT-is-arrD*
  and *smc-SemiCAT-is-arrE* = *dg-SemiCAT-is-arrE*
  and *smc-SemiCAT-is-arr-iff*[*smc-SemiCAT-simps*] = *dg-SemiCAT-is-arr-iff*

4.16.3 Composable arrows

lemma *smc-SemiCAT-composable-arrs-dg-SemiCAT*:
  composable-arrs (*dg-SemiCAT* α) = *composable-arrs* (*smc-SemiCAT* α)
  ⟨*proof*⟩

lemma *smc-SemiCAT-Comp*:
  smc-SemiCAT $\alpha(\!|$*Comp*$|\!)$ =
    $(\lambda\mathfrak{GF}\in_\circ$*composable-arrs* (*smc-SemiCAT* α)$.$ $\mathfrak{GF}(\!|0|\!) \circ_{DGHM} \mathfrak{GF}(\!|1_{\mathfrak{N}}|\!))$
  ⟨*proof*⟩

4.16.4 Composition

lemma *smc-SemiCAT-Comp-app*[*smc-SemiCAT-simps*]:
assumes $\mathfrak{G} : \mathfrak{B} \mapsto_{\textit{smc-SemiCAT}\ \alpha} \mathfrak{C}$ **and** $\mathfrak{F} : \mathfrak{A} \mapsto_{\textit{smc-SemiCAT}\ \alpha} \mathfrak{B}$
shows $\mathfrak{G} \circ_{A\,\textit{smc-SemiCAT}\ \alpha} \mathfrak{F} = \mathfrak{G} \circ_{SMCF} \mathfrak{F}$
⟨*proof*⟩

lemma *smc-SemiCAT-Comp-vdomain*[*smc-SemiCAT-simps*]:
$\mathcal{D}_\circ$ (*smc-SemiCAT* α(|*Comp*|)) = *composable-arrs* (*smc-SemiCAT* α)
⟨*proof*⟩

lemma *smc-SemiCAT-Comp-vrange*: $\mathcal{R}_\circ$ (*smc-SemiCAT* α(|*Comp*|)) $\subseteq_\circ$ *all-smcfs* α
⟨*proof*⟩

4.16.5 *SemiCAT* is a semicategory

lemma (**in** $\mathcal{Z}$) *tiny-semicategory-smc-SemiCAT*:
assumes $\mathcal{Z}$ β **and** $\alpha \in_\circ \beta$
shows *tiny-semicategory* β (*smc-SemiCAT* α)
⟨*proof*⟩

4.16.6 Initial object

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-initialI*: *obj-initial* (*smc-SemiCAT* α) *smc*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-initialD*:
assumes *obj-initial* (*smc-SemiCAT* α) $\mathfrak{A}$
shows $\mathfrak{A}$ = *smc*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-initialE*:
assumes *obj-initial* (*smc-SemiCAT* α) $\mathfrak{A}$
obtains $\mathfrak{A}$ = *smc*-0
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-initial-iff*[*smc-SemiCAT-simps*]:
obj-initial (*smc-SemiCAT* α) $\mathfrak{A} \longleftrightarrow \mathfrak{A}$ = *smc*-0
⟨*proof*⟩

4.16.7 Terminal object

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-terminalI*[*smc-SemiCAT-intros*]:
assumes $a \in_\circ$ *Vset* α **and** $f \in_\circ$ *Vset* α
shows *obj-terminal* (*smc-SemiCAT* α) (*smc*-1 a f)
⟨*proof*⟩

lemma (**in** $\mathcal{Z}$) *smc-SemiCAT-obj-terminalE*:
assumes *obj-terminal* (*smc-SemiCAT* α) $\mathfrak{B}$
obtains a f **where** $a \in_\circ$ *Vset* α **and** $f \in_\circ$ *Vset* α **and** $\mathfrak{B}$ = *smc*-1 a f
⟨*proof*⟩

Bibliography

[1] Association of Mizar Users. Mizar home page., . URL http://mizar.org/.

[2] Encyclopedia of Mathematics, . URL https://www.encyclopediaofmath.org/index.php/Main_Page.

[3] nLab, . URL https://ncatlab.org/nlab/show/HomePage.

[4] ProofWiki, . URL https://proofwiki.org/wiki/Main_Page.

[5] Wikipedia, 2001. URL https://www.wikipedia.org/.

[6] Isabelle/HOL Standard Library, 2020. URL https://isabelle.in.tum.de/website-Isabelle2020/dist/library/HOL/HOL/index.html.

[7] J. Adamek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories - The Joy of Cats*. 2006.

[8] C. Ballarin. Locales and Locale Expressions in Isabelle/Isar. In S. Berardi, M. Coppo, and F. Damiani, editors, *Types for Proofs and Programs*, volume 3085, pages 34–50. Springer, Heidelberg, 2004. ISBN 978-3-540-22164-7.

[9] C. Ballarin. Tutorial to Locales and Locale Interpretation. 2020. URL https://isabelle.in.tum.de/website-Isabelle2020/dist/Isabelle2020/doc/locales.pdf.

[10] G. Bancerek. Categorial Categories and Slice Categories. *Formalized Mathematics*, 5(2): 157–165, 1996.

[11] G. Bancerek. Indexed Category. *Formalized Mathematics*, 5(3):329–337, 1996.

[12] G. Bancerek. Concrete Categories. *Formalized Mathematics*, 9(3):605–621, 2001.

[13] G. Bancerek and A. Darmochwał. Comma Category. *Formalized Mathematics*, 2(5): 679–681, 1991.

[14] E. D. Bloch. *The Real Numbers and Real Analysis*. Springer Science + Business Media, Heidelberg, 2010. ISBN 978-0-387-72176-7.

[15] P. Bodo. *Categories and Functors*. Academic Press, New York, 1970.

[16] N. Bourbaki. *Elements of Mathematics, Theory of Sets*. Originally published as Éléments de Mathématique Théorie des Ensembles; Paris: N. Bourbaki. Reprint, Heidelberg: Springer-Verlag, 2004., 1970. ISBN 978-3-540-22525-6.

[17] C. E. Brown, C. Kaliszyk, and K. Pak. Higher-Order Tarski Grothendieck as a Foundation for Formal Proof. In J. Harrison, J. O'Leary, and A. Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, pages 9:1–9:16, Portland, USA, 2019.

[18] C. Byliński. Introduction to Categories and Functors. *Formalized Mathematics*, 1(2): 409–420, 1990.

[19] C. Byliński. Subcategories and Products of Categories. *Formalized Mathematics*, 1(4): 725–732, 1990.

[20] C. Byliński. Category Ens. *Formalized Mathematics*, 2(4):527–533, 1991.

[21] C. Byliński. Opposite Categories and Contravariant Functors. *Formalized Mathematics*, 2 (3):419–424, 1991.

[22] C. Byliński. Products and Coproducts in Categories. *Formalized Mathematics*, 2(5): 701–709, 1991.

[23] C. Byliński. Cartesian Categories. *Formalized Mathematics*, 3(2):161–169, 1992.

[24] M. Cáccamo and G. Winskel. A Higher-Order Calculus for Categories. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher Order Logics*, pages 136–153, Berlin, Heidelberg, 2001. Springer. ISBN 978-3-540-44755-9. doi: 10.1007/3-540-44755-5_11.

[25] M. J. Cáccamo and G. Winskel. A Higher-Order Calculus for Categories. Technical report, University of Aarhus, Aarhus, Denmark, 2001.

[26] J. Chen, K. Kappelmann, and A. Krauss. HOTG, 2021. URL https://bitbucket.org/cezaryka/tyset/src.

[27] M. Eberl. Syntax proposal: multiway if, 2021. URL https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2021-February/msg00034.html.

[28] 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.

[29] M. Goliński and A. Korniłowicz. Coproducts in Categories without Uniqueness of cod and dom. *Formalized Mathematics*, 21(4):235–239, 2013.

[30] A. Grabowski and Y. Shidama. *Preface*, volume 22. 2014.

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

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

[33] F. Kammüller, M. Wenzel, and L. C. Paulson. Locales A Sectioning Concept for Isabelle. In Y. Bertot, G. Dowek, L. Théry, A. Hirschowitz, and C. Paulin-Mohring, editors, *Proceedings of the 12th International Conference on Theorem Proving in Higher Order Logics, TPHOLs'99, Nice, France, September, 1999*, Lecture Notes in Computer Science, pages 149–165, Heidelberg, Germany, 1999. Springer. ISBN 978-3-540-48256-7. doi: 10.1007/3-540-48256-3_11.

[34] A. Katovsky. Category Theory. *Archive of Formal Proofs*, 2010.

[35] J. L. Kelley. *General Topology*. Originally published as General Topology; New York, NY, USA: Van Nostrand Reinhold Company. Reprint, Mineola, NY, USA: Dover Publications, 2017, 1955. ISBN 978-0-486-81544-2.

[36] A. Korniłowicz. On the Categories Without Uniqueness of cod and dom. Some Properties of the Morphisms and the Functors. *Formalized Mathematics*, 6(4):475–481, 1997.

[37] A. Korniłowicz. The Composition of Functors and Transformations in Alternative Categories. *Formalized Mathematics*, 7(1):1–7, 1998.

[38] A. Korniłowicz. Products in Categories without Uniqueness of cod and dom. *Formalized Mathematics*, 20(4):303–307, 2012.

[39] 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.

[40] N. Megill and D. A. Wheeler. *Metamath: A Computer Language for Mathematical Proofs*. Lulu Press, Morrisville, North Carolina, 2019. ISBN 978-0-359-70223-7.

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

[42] B. Mitchell. The Dominion of Isbell. *Transactions of the American Mathematical Society*, 167, 1972.

[43] M. Muzalewski. Categories of Groups. *Formalized Mathematics*, 2(4):563–571, 1991.

[44] M. Muzalewski. Category of Rings. *Formalized Mathematics*, 2(5):643–648, 1991.

[45] M. Muzalewski. Category of Left Modules. *Formalized Mathematics*, 2(5):649–652, 1991.

[46] R. Nieszczerzewski. Category of Functors Between Alternative Categories. *Formalized Mathematics*, 6(3):371–375, 1997.

[47] 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.

[48] G. O'Keefe. Category Theory to Yoneda's Lemma. *Archive of Formal Proofs*, 2005.

[49] L. C. Paulson. Natural Deduction as Higher-Order Resolution. *The Journal of Logic Programming*, 3(3):237–258, 1986. doi: 10.1016/0743-1066(86)90015-4.

[50] L. C. Paulson. The Hereditarily Finite Sets. *Archive of Formal Proofs*, 2013.

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

[52] V. K. Rao. On Doing Category Theory within Set Theoretic Foundations. In G. Sica, editor, *What is Category Theory?* Polimetrica s.a.s., 2006. ISBN 978-88-7699-031-1.

[53] M. Riccardi. Object-Free Definition of Categories. *Formalized Mathematics*, 21(3):193–205, 2013.

[54] M. Riccardi. Categorical Pullbacks. *Formalized Mathematics*, 23(1):1–14, 2015. doi: 10.2478/forma-2015-0001.

[55] M. Riccardi. Exponential Objects. *Formalized Mathematics*, 23(4):351–369, 2015.

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

[57] M. A. Shulman. Set Theory for Category Theory. *arXiv:0810.1279 [math]*, 2008. URL http://arxiv.org/abs/0810.1279.

[58] E. W. Stark. Category Theory with Adjunctions and Limits. *Archive of Formal Proofs*, 2016.

[59] G. Takeuti and W. M. Zaring. *Introduction to Axiomatic Set Theory*. Springer-Verlag, Heidelberg, 1971. ISBN 0-387-05302-6.

[60] A. Trybulec. Isomorphisms of Categories. *Formalized Mathematics*, 2(5):629–634, 1991.

[61] A. Trybulec. Natural Transformations. Discrete Categories. *Formalized Mathematics*, 2(4): 467–474, 1991.

[62] A. Trybulec. Some Isomorphisms Between Functor Categories. *Formalized Mathematics*, 3(1):33–40, 1992.

[63] A. Trybulec. Categories without Uniqueness of cod and dom. *Formalized Mathematics*, 5 (2):259–267, 1996.

[64] A. Trybulec. Functors for Alternative Categories. *Formalized Mathematics*, 5(4):595–608, 1996.