

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

Mihails Milehins

February 6, 2026

## 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

---

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Background . . . . .	9
1.2	Related and previous work . . . . .	9
<b>2</b>	<b>Set Theory for Category Theory</b>	<b>11</b>
2.1	Introduction . . . . .	11
2.1.1	Background . . . . .	11
2.1.2	References, related and previous work . . . . .	11
2.1.3	Notation . . . . .	11
2.1.4	Further foundational results . . . . .	11
2.2	Further set algebra and other miscellaneous results . . . . .	13
2.2.1	Background . . . . .	13
2.2.2	Further notation . . . . .	13
2.2.3	Elementary results. . . . .	14
2.2.4	<i>VBall</i> . . . . .	15
2.2.5	<i>VBe<sub>x</sub></i> . . . . .	15
2.2.6	Subset . . . . .	16
2.2.7	Equality . . . . .	17
2.2.8	Binary intersection . . . . .	18
2.2.9	Binary union . . . . .	19
2.2.10	Set difference . . . . .	20
2.2.11	Augmenting a set with an element . . . . .	22
2.2.12	Power set . . . . .	24
2.2.13	Singletons, using insert . . . . .	24
2.2.14	Intersection of elements . . . . .	25
2.2.15	Union of elements . . . . .	27
2.2.16	Pairs . . . . .	28
2.2.17	Cartesian products . . . . .	30
2.2.18	Pairwise . . . . .	30
2.2.19	Disjoint sets . . . . .	31
2.3	Further properties of natural numbers . . . . .	33
2.3.1	Background . . . . .	33
2.3.2	Conversion between <i>V</i> and <i>nat</i> . . . . .	33
2.3.3	Elementary results . . . . .	34
2.3.4	Induction . . . . .	35
2.3.5	Methods . . . . .	35
2.3.6	Auxiliary . . . . .	35

2.4	Elementary binary relations . . . . .	37
2.4.1	Background . . . . .	37
2.4.2	Constructors . . . . .	37
2.4.3	Properties . . . . .	48
2.4.4	Classification of relations . . . . .	59
2.4.5	Tools: <i>mk-VLambda</i> . . . . .	75
2.5	Operations on indexed families of sets . . . . .	78
2.5.1	Background . . . . .	78
2.5.2	Intersection of an indexed family of sets . . . . .	78
2.5.3	Union of an indexed family of sets . . . . .	80
2.5.4	Additional simplification rules for indexed families of sets. . . . .	82
2.5.5	Knowledge transfer: union and intersection of indexed families . . . . .	83
2.5.6	Disjoint union . . . . .	84
2.5.7	Canonical injection . . . . .	84
2.5.8	Infinite Cartesian product . . . . .	85
2.5.9	Projection . . . . .	87
2.5.10	Cartesian power of a set . . . . .	87
2.6	Equipollence . . . . .	89
2.6.1	Background . . . . .	89
2.6.2	<i>veqpoll</i> . . . . .	89
2.6.3	<i>vlepoll</i> . . . . .	89
2.6.4	<i>vlespoll</i> . . . . .	90
2.7	Cardinality . . . . .	91
2.7.1	Background . . . . .	91
2.7.2	Cardinality of an arbitrary set . . . . .	91
2.7.3	Finite sets . . . . .	92
2.8	Further results about ordinal numbers . . . . .	95
2.8.1	Background . . . . .	95
2.8.2	Further ordinal arithmetic and inequalities . . . . .	95
2.9	Finite sequences . . . . .	97
2.9.1	Background . . . . .	97
2.9.2	Definition and common properties . . . . .	97
2.9.3	Appending an element to a finite sequence: <i>vcons</i> . . . . .	99
2.9.4	Transfer between the type <i>V list</i> and finite sequences . . . . .	102
2.9.5	Finite sequences and the Cartesian product . . . . .	106
2.9.6	Binary Cartesian product based on finite sequences: <i>ftimes</i> . . . . .	107
2.9.7	Sequence as an element of a Cartesian power of a set . . . . .	108
2.9.8	The set of all finite sequences on a set . . . . .	109
2.10	Binary relation as a finite sequence . . . . .	110
2.10.1	Background . . . . .	110
2.10.2	<i>fpairs</i> . . . . .	110
2.10.3	Constructors . . . . .	111
2.10.4	Properties . . . . .	120
2.10.5	Classification of relations . . . . .	128

2.11	Further results related to the von Neumann hierarchy of sets . . . . .	132
2.11.1	Background . . . . .	132
2.11.2	Further properties of $V_{from}$ . . . . .	132
2.11.3	Operations closed with respect to $V_{set}$ . . . . .	133
2.11.4	Axioms for $V_{set} \alpha$ . . . . .	139
2.11.5	Existence of a disjoint subset in $V_{set} \alpha$ . . . . .	140
2.12	$n$ -ary operation . . . . .	142
2.12.1	Partial $n$ -ary operation . . . . .	142
2.12.2	Total $n$ -ary operation . . . . .	142
2.12.3	Injective $n$ -ary operation . . . . .	143
2.12.4	Surjective $n$ -ary operation . . . . .	143
2.12.5	Bijjective $n$ -ary operation . . . . .	144
2.12.6	Scalar . . . . .	145
2.12.7	Unary operation . . . . .	145
2.12.8	Injective unary operation . . . . .	145
2.12.9	Surjective unary operation . . . . .	146
2.12.10	Bijjective unary operation . . . . .	146
2.12.11	Partial binary operation . . . . .	147
2.12.12	Total binary operation . . . . .	147
2.12.13	Injective binary operation . . . . .	148
2.12.14	Surjective binary operation . . . . .	148
2.12.15	Bijjective binary operation . . . . .	149
2.12.16	Flip . . . . .	149
2.13	Construction of integer numbers, rational numbers and real numbers . . . . .	151
2.13.1	Background . . . . .	151
2.13.2	Real numbers . . . . .	151
2.13.3	Integer numbers . . . . .	158
2.13.4	Rational numbers . . . . .	164
2.13.5	Upper bound on the cardinality of the continuum for $V$ . . . . .	170
2.14	Example I: absence of replacement in $V_{\omega+\omega}$ . . . . .	171
2.15	Example II: topological spaces . . . . .	172
2.15.1	Background . . . . .	172
2.15.2	$\mathcal{Z}$ -sequence . . . . .	172
2.15.3	Topological space . . . . .	172
2.15.4	Indiscrete topology . . . . .	173
2.16	Example III: abstract algebra . . . . .	174
2.16.1	Background . . . . .	174
2.16.2	Semigroup . . . . .	174
2.16.3	Commutative semigroup . . . . .	175
2.16.4	Semiring . . . . .	175
2.16.5	Integer numbers form a semiring . . . . .	177
<b>3</b>	<b>Digraphs</b> . . . . .	<b>178</b>
3.1	Introduction . . . . .	178

3.1.1	Background . . . . .	178
3.1.2	Preliminaries . . . . .	178
3.1.3	CS setup for foundations . . . . .	178
3.2	Digraph . . . . .	186
3.2.1	Background . . . . .	186
3.2.2	Arrow with a domain and a codomain . . . . .	186
3.2.3	<i>Hom</i> -set . . . . .	186
3.2.4	Digraph: background information . . . . .	187
3.2.5	Digraph: definition and elementary properties . . . . .	187
3.2.6	Opposite digraph . . . . .	191
3.3	Smallness for digraphs . . . . .	192
3.3.1	Background . . . . .	192
3.3.2	Tiny digraph . . . . .	192
3.3.3	Finite digraph . . . . .	193
3.4	Homomorphism of digraphs . . . . .	195
3.4.1	Background . . . . .	195
3.4.2	Definition and elementary properties . . . . .	195
3.4.3	Opposite digraph homomorphism . . . . .	198
3.4.4	Composition of covariant digraph homomorphisms . . . . .	199
3.4.5	Composition of contravariant digraph homomorphisms . . . . .	200
3.4.6	Identity digraph homomorphism . . . . .	202
3.4.7	Constant digraph homomorphism . . . . .	203
3.4.8	Faithful digraph homomorphism . . . . .	204
3.4.9	Full digraph homomorphism . . . . .	206
3.4.10	Fully faithful digraph homomorphism . . . . .	206
3.4.11	Isomorphism of digraphs . . . . .	207
3.4.12	Inverse digraph homomorphism . . . . .	209
3.4.13	An isomorphism of digraphs is an isomorphism in the category <i>GRPH</i> . . . . .	211
3.4.14	Isomorphic digraphs . . . . .	211
3.5	Smallness for digraph homomorphisms . . . . .	213
3.5.1	Digraph homomorphism with tiny maps . . . . .	213
3.5.2	Tiny digraph homomorphism . . . . .	215
3.6	Transformation of digraph homomorphisms . . . . .	217
3.6.1	Background . . . . .	217
3.6.2	Definition and elementary properties . . . . .	217
3.6.3	Opposite transformation of digraph homomorphisms . . . . .	219
3.6.4	Composition of a transformation of digraph homomorphisms and a digraph homomorphism . . . . .	220
3.6.5	Composition of a digraph homomorphism and a transformation of digraph homomorphisms . . . . .	222
3.7	Smallness for transformations of digraph homomorphisms . . . . .	224
3.7.1	Transformation of digraph homomorphisms with tiny maps . . . . .	224
3.7.2	Transformation of homomorphisms of tiny digraphs . . . . .	225
3.8	Product digraph . . . . .	228

3.8.1	Background . . . . .	228
3.8.2	Product digraph: definition and elementary properties . . . . .	228
3.8.3	Local assumptions for a product digraph . . . . .	228
3.8.4	Further local assumptions for product digraphs . . . . .	231
3.8.5	Local assumptions for a finite product digraph . . . . .	232
3.8.6	Binary union and complement . . . . .	232
3.8.7	Projection . . . . .	234
3.8.8	Digraph product universal property digraph homomorphism . . . . .	235
3.8.9	Singleton digraph . . . . .	236
3.8.10	Singleton digraph homomorphism . . . . .	237
3.8.11	Product of two digraphs . . . . .	238
3.8.12	Projections for the product of two digraphs . . . . .	242
3.8.13	Product of three digraphs . . . . .	243
3.9	Subdigraph . . . . .	247
3.9.1	Background . . . . .	247
3.9.2	Simple subdigraph . . . . .	247
3.9.3	Inclusion digraph homomorphism . . . . .	248
3.9.4	Full subdigraph . . . . .	249
3.9.5	Wide subdigraph . . . . .	250
3.10	Simple digraphs . . . . .	252
3.10.1	Background . . . . .	252
3.10.2	Empty digraph $0$ . . . . .	252
3.10.3	Empty digraph homomorphism . . . . .	252
3.10.4	Empty transformation of digraph homomorphisms . . . . .	254
3.10.5	$10$ : digraph with one object and no arrows . . . . .	254
3.10.6	$1$ : digraph with one object and one arrow . . . . .	255
3.11	$GRPH$ as a digraph . . . . .	257
3.11.1	Background . . . . .	257
3.11.2	Definition and elementary properties . . . . .	257
3.11.3	Object . . . . .	257
3.11.4	Domain . . . . .	257
3.11.5	Codomain . . . . .	258
3.11.6	$GRPH$ is a digraph . . . . .	258
3.11.7	Arrow with a domain and a codomain . . . . .	258
3.12	$Rel$ as a digraph . . . . .	259
3.12.1	Background . . . . .	259
3.12.2	Canonical arrow for $V$ . . . . .	259
3.12.3	Arrow for $Rel$ . . . . .	259
3.12.4	$Rel$ as a digraph . . . . .	264
3.12.5	Canonical dagger for $Rel$ . . . . .	265
3.13	$Par$ as a digraph . . . . .	268
3.13.1	Background . . . . .	268
3.13.2	Arrow for $Par$ . . . . .	268
3.13.3	$Par$ as a digraph . . . . .	270

3.14	<i>Set</i> as a digraph . . . . .	273
3.14.1	Background . . . . .	273
3.14.2	Arrow for <i>Set</i> . . . . .	273
3.14.3	<i>Set</i> as a digraph . . . . .	275
<b>4</b>	<b>Semicategories</b>	<b>278</b>
4.1	Introduction . . . . .	278
4.1.1	Background . . . . .	278
4.1.2	Preliminaries . . . . .	278
4.1.3	CS setup for foundations . . . . .	278
4.2	Semicategory . . . . .	279
4.2.1	Background . . . . .	279
4.2.2	Definition and elementary properties . . . . .	280
4.2.3	Opposite semicategory . . . . .	284
4.2.4	Arrow with a domain and a codomain . . . . .	285
4.2.5	Monic arrow and epic arrow . . . . .	286
4.2.6	Idempotent arrow . . . . .	287
4.2.7	Terminal object and initial object . . . . .	288
4.2.8	Null object . . . . .	289
4.2.9	Zero arrow . . . . .	289
4.3	Smallness for semicategories . . . . .	291
4.3.1	Background . . . . .	291
4.3.2	Tiny semicategory . . . . .	291
4.3.3	Finite semicategory . . . . .	293
4.4	Semifunctor . . . . .	295
4.4.1	Background . . . . .	295
4.4.2	Definition and elementary properties . . . . .	295
4.4.3	Opposite semifunctor . . . . .	299
4.4.4	Composition of covariant semifunctors . . . . .	300
4.4.5	Composition of contravariant semifunctors . . . . .	301
4.4.6	Identity semifunctor . . . . .	304
4.4.7	Constant semifunctor . . . . .	305
4.4.8	Faithful semifunctor . . . . .	306
4.4.9	Full semifunctor . . . . .	307
4.4.10	Fully faithful semifunctor . . . . .	309
4.4.11	Isomorphism of semicategories . . . . .	310
4.4.12	Inverse semifunctor . . . . .	312
4.4.13	An isomorphism of semicategories is an isomorphism in the category <i>Sem- iCAT</i> . . . . .	313
4.4.14	Isomorphic semicategories . . . . .	313
4.5	Smallness for semifunctors . . . . .	315
4.5.1	Semifunctor with tiny maps . . . . .	315
4.5.2	Tiny semifunctor . . . . .	317
4.6	Natural transformation of a semifunctor . . . . .	320

4.6.1	Background . . . . .	320
4.6.2	Definition and elementary properties . . . . .	320
4.6.3	Opposite natural transformation of semifunctors . . . . .	324
4.6.4	Vertical composition of natural transformations . . . . .	325
4.6.5	Horizontal composition of natural transformations . . . . .	326
4.6.6	Interchange law . . . . .	328
4.6.7	Composition of a natural transformation of semifunctors and a semifunctor	329
4.6.8	Composition of a semifunctor and a natural transformation of semifunctors	330
4.7	Smallness for natural transformations of semifunctors . . . . .	332
4.7.1	Natural transformation of semifunctors with tiny maps . . . . .	332
4.7.2	Tiny natural transformation of semifunctors . . . . .	334
4.8	Product semicategory . . . . .	338
4.8.1	Background . . . . .	338
4.8.2	Product semicategory: definition and elementary properties . . . . .	338
4.8.3	Local assumptions for a product semicategory . . . . .	339
4.8.4	Further local assumptions for product semicategories . . . . .	341
4.8.5	Local assumptions for a finite product semicategory . . . . .	342
4.8.6	Binary union and complement . . . . .	342
4.8.7	Projection . . . . .	343
4.8.8	Semicategory product universal property semifunctor . . . . .	344
4.8.9	Singleton semicategory . . . . .	345
4.8.10	Singleton semifunctor . . . . .	346
4.8.11	Product of two semicategories . . . . .	347
4.8.12	Projections for the product of two semicategories . . . . .	349
4.8.13	Product of three semicategories . . . . .	351
4.9	Subsemicategory . . . . .	353
4.9.1	Background . . . . .	353
4.9.2	Simple subsemicategory . . . . .	353
4.9.3	Inclusion semifunctor . . . . .	355
4.9.4	Full subsemicategory . . . . .	355
4.9.5	Wide subsemicategory . . . . .	356
4.10	Simple semicategories . . . . .	358
4.10.1	Background . . . . .	358
4.10.2	Empty semicategory $0$ . . . . .	358
4.10.3	Empty semifunctor . . . . .	358
4.10.4	Empty natural transformation of semifunctors . . . . .	360
4.10.5	$1_0$ : semicategory with one object and no arrows . . . . .	361
4.10.6	$1_1$ : semicategory with one object and one arrow . . . . .	362
4.11	$Rel$ as a semicategory . . . . .	363
4.11.1	Background . . . . .	363
4.11.2	$Rel$ as a semicategory . . . . .	363
4.11.3	Canonical dagger for $Rel$ . . . . .	365
4.11.4	Monic arrow and epic arrow . . . . .	366
4.11.5	Terminal object, initial object and null object . . . . .	367

4.11.6	Zero arrow	368
4.12	<i>Par</i> as a semicategory	369
4.12.1	Background	369
4.12.2	<i>Par</i> as a semicategory	369
4.12.3	Monic arrow and epic arrow	370
4.12.4	Terminal object, initial object and null object	371
4.12.5	Zero arrow	371
4.13	<i>Set</i> as a semicategory	373
4.13.1	Background	373
4.13.2	<i>Set</i> as a semicategory	373
4.13.3	Monic arrow and epic arrow	375
4.13.4	Terminal object, initial object and null object	375
4.13.5	Zero arrow	376
4.14	<i>GRPH</i> as a semicategory	377
4.14.1	Background	377
4.14.2	Definition and elementary properties	377
4.14.3	Composable arrows	377
4.14.4	Composition	378
4.14.5	<i>GRPH</i> is a semicategory	378
4.14.6	Initial object	378
4.14.7	Terminal object	378
4.15	<i>SemiCAT</i> as a digraph	379
4.15.1	Background	379
4.15.2	Definition and elementary properties	379
4.15.3	Object	379
4.15.4	Domain and codomain	379
4.15.5	<i>SemiCAT</i> is a digraph	380
4.15.6	Arrow with a domain and a codomain	380
4.16	<i>SemiCAT</i> as a semicategory	381
4.16.1	Background	381
4.16.2	Definition and elementary properties	381
4.16.3	Composable arrows	381
4.16.4	Composition	382
4.16.5	<i>SemiCAT</i> is a semicategory	382
4.16.6	Initial object	382
4.16.7	Terminal object	382

---

# 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*  $\langle (- ? - : -) \rangle [0, 0, 10] 10$

**where**  $C ? A : B \equiv \text{if } C \text{ then } A \text{ else } B$

**abbreviation (input)** *if2 where if2*  $a b \equiv (\lambda i. (i = 0 ? a : b))$

### 2.1.4 Further foundational results

**lemma** *theD*:

**assumes**  $\exists! x. P x$  **and**  $x = (THE x. P x)$

**shows**  $P x$  **and**  $P y \implies x = y$

*<proof>*

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

**lemma** *bij-betwE[elim]*:

**assumes** *bij-betw f A B* **and**  $[[ \textit{inj-on } f A; f ' A = B ]]$   $\implies P$

**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*:  $\exists !x. P x \implies P x \implies P y \implies x = y$  *<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* ::  $V \Rightarrow V \Rightarrow \text{bool}$  ( $\langle(-/\epsilon_{\circ} -)\rangle$  [51, 51] 50)

**where** *vmember*  $x A \equiv (x \in \text{elts } A)$

**notation** *vmember* ( $\langle'(\epsilon_{\circ}')\rangle$ )

**and** *vmember* ( $\langle(-/\epsilon_{\circ} -)\rangle$  [51, 51] 50)

**abbreviation** *not-vmember* ::  $V \Rightarrow V \Rightarrow \text{bool}$  ( $\langle(-/\notin_{\circ} -)\rangle$  [51, 51] 50)

**where** *not-vmember*  $x A \equiv (x \notin \text{elts } A)$

**notation**

*not-vmember* ( $\langle'(\notin_{\circ}')\rangle$ ) **and**

*not-vmember* ( $\langle(-/\notin_{\circ} -)\rangle$  [51, 51] 50)

#### Subsets

**abbreviation** *vsubset* ::  $V \Rightarrow V \Rightarrow \text{bool}$

**where** *vsubset*  $\equiv \text{less}$

**abbreviation** *vsubset-eq* ::  $V \Rightarrow V \Rightarrow \text{bool}$

**where** *vsubset-eq*  $\equiv \text{less-eq}$

**notation** *vsubset* ( $\langle'(\subseteq_{\circ}')\rangle$ )

**and** *vsubset* ( $\langle(-/\subseteq_{\circ} -)\rangle$  [51, 51] 50)

**and** *vsubset-eq* ( $\langle'(\subseteq_{\circ}')\rangle$ )

**and** *vsubset-eq* ( $\langle(-/\subseteq_{\circ} -)\rangle$  [51, 51] 50)

#### Ball

**syntax**

-*VBall* ::  $\text{pttrn} \Rightarrow V \Rightarrow \text{bool} \Rightarrow \text{bool}$  ( $\langle(3\forall(-/\epsilon_{\circ}-)/-)\rangle$  [0, 0, 10] 10)

-*VBex* ::  $\text{pttrn} \Rightarrow V \Rightarrow \text{bool} \Rightarrow \text{bool}$  ( $\langle(3\exists(-/\epsilon_{\circ}-)/-)\rangle$  [0, 0, 10] 10)

-*VBex1* ::  $\text{pttrn} \Rightarrow V \Rightarrow \text{bool} \Rightarrow \text{bool}$  ( $\langle(3\exists!(-/\epsilon_{\circ}-)/-)\rangle$  [0, 0, 10] 10)

**syntax-consts**

-*VBall*  $\hat{=} \text{Ball}$  **and**

-*VBex*  $\hat{=} \text{Bex}$  **and**

-*VBex1*  $\hat{=} \text{Ex1}$

**translations**

$\forall x \in_{\circ} A. P \hat{=} \text{CONST Ball } (\text{CONST elts } A) (\lambda x. P)$

$\exists x \in_{\circ} A. P \hat{=} \text{CONST Bex } (\text{CONST elts } A) (\lambda x. P)$

$\exists! x \in_{\circ} A. P \rightarrow \exists! x. x \in_{\circ} A \wedge P$

*VLambda*

The following notation was adapted from [50].

**syntax** *-vlam* ::  $[\text{pttrn}, V, V] \Rightarrow V$  ( $\langle(3\lambda-\epsilon_{\circ}-/-)\rangle$  10)

**syntax-consts**  $-vlam \Rightarrow VLambda$

**translations**  $\lambda x \in_o A. f \Rightarrow CONST VLambda A (\lambda x. f)$

### Intersection and union

**abbreviation**  $vintersection :: V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \cap_o \rangle$  70)

**where**  $\cap_o \equiv (\cap)$

**notation**  $vintersection$  (**infixl**  $\langle \cap_o \rangle$  70)

**abbreviation**  $vunion :: V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \cup_o \rangle$  65)

**where**  $vunion \equiv sup$

**notation**  $vunion$  (**infixl**  $\langle \cup_o \rangle$  65)

**abbreviation**  $VInter :: V \Rightarrow V$  ( $\langle \cap_o \rangle$ )

**where**  $\cap_o A \equiv \sqcap$  (*elts*  $A$ )

**notation**  $VInter$  ( $\langle \cap_o \rangle$ )

**abbreviation**  $VUnion :: V \Rightarrow V$  ( $\langle \cup_o \rangle$ )

**where**  $\cup_o A \equiv \sqcup$  (*elts*  $A$ )

**notation**  $VUnion$  ( $\langle \cup_o \rangle$ )

### Miscellaneous

**notation**  $app$  ( $\langle \cdot \rangle$ ) [999, 50] 999)

**notation**  $vtimes$  (**infixr**  $\langle \times_o \rangle$  80)

### 2.2.3 Elementary results.

**lemma**  $vempty-nin[simp]$ :  $a \notin_o 0$  (*proof*)

**lemma**  $vemptyE$ :

**assumes**  $A \neq 0$

**obtains**  $x$  **where**  $x \in_o A$

(*proof*)

**lemma**  $in-set-CollectI$ :

**assumes**  $P x$  **and**  $small \{x. P x\}$

**shows**  $x \in_o set \{x. P x\}$

(*proof*)

**lemma**  $small-setcompr2$ :

**assumes**  $small \{f x y \mid x y. P x y\}$  **and**  $a \in_o set \{f x y \mid x y. P x y\}$

**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**  $x \in_o set A$

(*proof*)

**lemma**  $in-small-setD$ :

**assumes**  $x \in_o set A$  **and**  $small A$

**shows**  $x \in A$

(*proof*)

**lemma**  $in-small-setE$ :

**assumes**  $a \in_o set A$  **and**  $small A$

**obtains**  $a \in A$

(*proof*)

**lemma** *small-set-vsubset*:

**assumes** *small*  $A$  **and**  $A \subseteq \text{elts } B$

**shows**  $\text{set } A \subseteq_{\circ} B$

*<proof>*

**lemma** *some-in-set-if-set-neq-vempty[simp]*:

**assumes**  $A \neq 0$

**shows**  $(\text{SOME } x. x \in_{\circ} A) \in_{\circ} A$

*<proof>*

**lemma** *small-vsubset-set[intro, simp]*:

**assumes** *small*  $B$  **and**  $A \subseteq B$

**shows**  $\text{set } A \subseteq_{\circ} \text{set } B$

*<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**  $a \notin_{\circ} \text{vinsert } b A$

*<proof>*

**lemma** *vsubset-if-subset*:

**assumes**  $\text{elts } A \subseteq \text{elts } B$

**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**  $\bigwedge x. x \in_{\circ} B \implies P x \longleftrightarrow Q x$

**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**  $\bigwedge x. x \in_{\circ} B =_{\text{simp}} \implies P x \longleftrightarrow Q x$

**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**  $\bigwedge x. x \in_{\circ} B \implies P x \longleftrightarrow Q x$

**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**  $\wedge x. x \in_o B = \text{simp} \Rightarrow P x \leftrightarrow Q x$   
**shows**  $(\exists x \in_o A. P x) \leftrightarrow (\exists x \in_o B. Q x)$   
*<proof>*

## 2.2.6 Subset

Rules.

**lemma** *vsubset-antisym*:  
**assumes**  $A \subseteq_o B$  **and**  $B \subseteq_o A$   
**shows**  $A = B$   
*<proof>*

**lemma** *vsubsetI*:  
**assumes**  $\wedge x. x \in_o A \Longrightarrow x \in_o B$   
**shows**  $A \subseteq_o B$   
*<proof>*

**lemma** *vpsubsetI*:  
**assumes**  $A \subseteq_o B$  **and**  $x \notin_o A$  **and**  $x \in_o B$   
**shows**  $A \subset_o B$   
*<proof>*

**lemma** *vsubsetD*:  
**assumes**  $A \subseteq_o B$   
**shows**  $\wedge x. x \in_o A \Longrightarrow x \in_o B$   
*<proof>*

**lemma** *vsubsetE*:  
**assumes**  $A \subseteq_o B$  **and**  $(\wedge x. x \in_o A \Longrightarrow x \in_o B) \Longrightarrow P$   
**shows**  $P$   
*<proof>*

**lemma** *vpsubsetE*:  
**assumes**  $A \subset_o B$   
**obtains**  $x$  **where**  $A \subseteq_o B$  **and**  $x \notin_o A$  **and**  $x \in_o B$   
*<proof>*

**lemma** *vsubset-iff*:  $A \subseteq_o B \leftrightarrow (\forall t. t \in_o A \longrightarrow t \in_o B)$  *<proof>*

Elementary properties.

**lemma** *vsubset-eq*:  $A \subseteq_o B \leftrightarrow (\forall x \in_o A. x \in_o B)$  *<proof>*

**lemma** *vsubset-transitive*[*intro*]:  
**assumes**  $A \subseteq_o B$  **and**  $B \subseteq_o C$   
**shows**  $A \subseteq_o C$   
*<proof>*

**lemma** *vsubset-reflexive*:  $A \subseteq_o A$  *<proof>*

Set operations.

**lemma** *vsubset-vempty*:  $0 \subseteq_o A$  *<proof>*

**lemma** *vsubset-vsingleton-left*:  $\text{set } \{a\} \subseteq_o A \leftrightarrow a \in_o A$  *<proof>*

**lemmas** *vsubset-vsingleton-leftD*[*dest*] = *vsubset-vsingleton-left*[*THEN iffD1*]  
**and** *vsubset-vsingleton-leftI*[*intro*] = *vsubset-vsingleton-left*[*THEN iffD2*]

**lemma** *vsubset-vsingleton-right*:  $A \subseteq_o \text{set } \{a\} \leftrightarrow A = \text{set } \{a\} \vee A = 0$   
 ⟨proof⟩

**lemmas** *vsubset-vsingleton-rightD*[*dest*] = *vsubset-vsingleton-right*[*THEN iffD1*]  
 and *vsubset-vsingleton-rightI*[*intro*] = *vsubset-vsingleton-right*[*THEN iffD2*]

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

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

**lemma** *vsubset-vinsert-leftD*[*dest*]:  
**assumes**  $\text{vinsert } a A \subseteq_o B$   
**shows**  $A \subseteq_o B$   
 ⟨proof⟩

**lemma** *vsubset-vinsert-leftI*[*intro*]:  
**assumes**  $A \subseteq_o B$  and  $a \in_o B$   
**shows**  $\text{vinsert } a A \subseteq_o B$   
 ⟨proof⟩

**lemma** *vsubset-vinsert-vinsertI*[*intro*]:  
**assumes**  $A \subseteq_o \text{vinsert } b B$   
**shows**  $\text{vinsert } b A \subseteq_o \text{vinsert } b B$   
 ⟨proof⟩

**lemma** *vsubset-vinsert-rightI*[*intro*]:  
**assumes**  $A \subseteq_o B$   
**shows**  $A \subseteq_o \text{vinsert } b B$   
 ⟨proof⟩

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

**lemmas** *vsubset-VPowD* = *vsubset-VPow*[*THEN iffD1*]  
 and *vsubset-VPowI* = *vsubset-VPow*[*THEN iffD2*]

Special properties.

**lemma** *vsubset-contraD*:  
**assumes**  $A \subseteq_o B$  and  $c \notin_o B$   
**shows**  $c \notin_o A$   
 ⟨proof⟩

## 2.2.7 Equality

Rules.

**lemma** *vequalityD1*:  
**assumes**  $A = B$   
**shows**  $A \subseteq_o B$   
 ⟨proof⟩

**lemma** *vequalityD2*:

**assumes**  $A = B$   
**shows**  $B \subseteq_0 A$   
 $\langle proof \rangle$

**lemma** *vequalityE*:

**assumes**  $A = B$  **and**  $\llbracket A \subseteq_0 B; B \subseteq_0 A \rrbracket \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma** *vequalityCE[elim]*:

**assumes**  $A = B$  **and**  $\llbracket c \in_0 A; c \in_0 B \rrbracket \implies P$  **and**  $\llbracket c \notin_0 A; c \notin_0 B \rrbracket \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

### 2.2.8 Binary intersection

**lemma** *vintersection-def*:  $A \cap_0 B = \text{set } \{x. x \in_0 A \wedge x \in_0 B\}$   
 $\langle proof \rangle$

**lemma** *small-vintersection-set[simp]*:  $\text{small } \{x. x \in_0 A \wedge x \in_0 B\}$   
 $\langle proof \rangle$

Rules.

**lemma** *vintersection-iff[simp]*:  $x \in_0 A \cap_0 B \longleftrightarrow x \in_0 A \wedge x \in_0 B$   
 $\langle proof \rangle$

**lemma** *vintersectionI[intro!]*:

**assumes**  $x \in_0 A$  **and**  $x \in_0 B$   
**shows**  $x \in_0 A \cap_0 B$   
 $\langle proof \rangle$

**lemma** *vintersectionD1[dest]*:

**assumes**  $x \in_0 A \cap_0 B$   
**shows**  $x \in_0 A$   
 $\langle proof \rangle$

**lemma** *vintersectionD2[dest]*:

**assumes**  $x \in_0 A \cap_0 B$   
**shows**  $x \in_0 B$   
 $\langle proof \rangle$

**lemma** *vintersectionE[elim!]*:

**assumes**  $x \in_0 A \cap_0 B$  **and**  $x \in_0 A \implies x \in_0 B \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

Elementary properties.

**lemma** *vintersection-intersection*:  $A \cap_0 B = \text{set } (\text{elts } A \cap \text{elts } B)$   
 $\langle proof \rangle$

**lemma** *vintersection-assoc*:  $A \cap_0 (B \cap_0 C) = (A \cap_0 B) \cap_0 C$   $\langle proof \rangle$

**lemma** *vintersection-commute*:  $A \cap_0 B = B \cap_0 A$   $\langle proof \rangle$

Previous set operations.

**lemma** *vsubset-vintersection-right*:  $A \subseteq_0 (B \cap_0 C) = (A \subseteq_0 B \wedge A \subseteq_0 C)$   
 $\langle proof \rangle$

**lemma** *vsubset-vintersection-rightD*[*dest*]:  
**assumes**  $A \sqsubseteq_0 (B \cap_0 C)$   
**shows**  $A \sqsubseteq_0 B$  **and**  $A \sqsubseteq_0 C$   
*<proof>*

**lemma** *vsubset-vintersection-rightI*[*intro*]:  
**assumes**  $A \sqsubseteq_0 B$  **and**  $A \sqsubseteq_0 C$   
**shows**  $A \sqsubseteq_0 (B \cap_0 C)$   
*<proof>*

Set operations.

**lemma** *vintersection-vempty*:  $0 \sqsubseteq_0 A$  *<proof>*

**lemma** *vintersection-vsingleton*:  $a \in_0 \text{set } \{a\}$  *<proof>*

**lemma** *vintersection-vdoubleton*:  $a \in_0 \text{set } \{a, b\}$  **and**  $b \in_0 \text{set } \{a, b\}$   
*<proof>*

**lemma** *vintersection-VPow*[*simp*]:  $VPow (A \cap_0 B) = VPow A \cap_0 VPow B$  *<proof>*

Special properties.

**lemma** *vintersection-function-mono*:  
**assumes** *mono f*  
**shows**  $f (A \cap_0 B) \sqsubseteq_0 f A \cap_0 f B$   
*<proof>*

## 2.2.9 Binary union

**lemma** *small-vunion-set*: *small*  $\{x. x \in_0 A \vee x \in_0 B\}$   
*<proof>*

Rules.

**lemma** *vunion-def*:  $A \cup_0 B = \text{set } \{x. x \in_0 A \vee x \in_0 B\}$   
*<proof>*

**lemma** *vunion-iff*[*simp*]:  $x \in_0 A \cup_0 B \iff x \in_0 A \vee x \in_0 B$   
*<proof>*

**lemma** *vunionI1*:  
**assumes**  $a \in_0 A$   
**shows**  $a \in_0 A \cup_0 B$   
*<proof>*

**lemma** *vunionI2*:  
**assumes**  $a \in_0 B$   
**shows**  $a \in_0 A \cup_0 B$   
*<proof>*

**lemma** *vunionCI*[*intro!*]:  
**assumes**  $x \notin_0 B \implies x \in_0 A$   
**shows**  $x \in_0 A \cup_0 B$   
*<proof>*

**lemma** *vunionE*[*elim!*]:  
**assumes**  $x \in_0 A \cup_0 B$  **and**  $x \in_0 A \implies P$  **and**  $x \in_0 B \implies P$   
**shows**  $P$   
*<proof>*

Elementary properties.

**lemma** *union-union*:  $A \cup_0 B = \text{set } (\text{elts } A \cup \text{elts } B)$  *<proof>*

**lemma** *union-assoc*:  $A \cup_0 (B \cup_0 C) = (A \cup_0 B) \cup_0 C$  *<proof>*

**lemma** *union-commute*:  $A \cup_0 B = B \cup_0 A$  *<proof>*

Previous set operations.

**lemma** *vsubset-union-left*:  $(A \cup_0 B) \subseteq_0 C \leftrightarrow (A \subseteq_0 C \wedge B \subseteq_0 C)$  *<proof>*

**lemma** *vsubset-union-leftD[dest]*:

**assumes**  $(A \cup_0 B) \subseteq_0 C$   
**shows**  $A \subseteq_0 C$  **and**  $B \subseteq_0 C$   
*<proof>*

**lemma** *vsubset-union-leftI[intro]*:

**assumes**  $A \subseteq_0 C$  **and**  $B \subseteq_0 C$   
**shows**  $(A \cup_0 B) \subseteq_0 C$   
*<proof>*

**lemma** *vintersection-union-left*:  $(A \cup_0 B) \cap_0 C = (A \cap_0 C) \cup_0 (B \cap_0 C)$   
*<proof>*

**lemma** *vintersection-union-right*:  $A \cap_0 (B \cup_0 C) = (A \cap_0 B) \cup_0 (A \cap_0 C)$   
*<proof>*

Set operations.

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

**lemma** *union-vsingleton[simp]*:  $\text{set } \{a\} \cup_0 A = \text{vinsert } a \ A$  *<proof>*

**lemma** *union-vdoubleton[simp]*:  $\text{set } \{a, b\} \cup_0 A = \text{vinsert } a \ (\text{vinsert } b \ A)$   
*<proof>*

**lemma** *union-vinsert-comm-left*:

$(\text{vinsert } a \ A) \cup_0 B = A \cup_0 (\text{vinsert } a \ B)$   
*<proof>*

**lemma** *union-vinsert-comm-right*:

$A \cup_0 (\text{vinsert } a \ B) = (\text{vinsert } a \ A) \cup_0 B$   
*<proof>*

**lemma** *vinsert-def*:  $\text{vinsert } y \ B = \text{set } \{x. x = y\} \cup_0 B$  *<proof>*

**lemma** *union-vinsert-left*:  $(\text{vinsert } a \ A) \cup_0 B = \text{vinsert } a \ (A \cup_0 B)$  *<proof>*

**lemma** *union-vinsert-right*:  $A \cup_0 (\text{vinsert } a \ B) = \text{vinsert } a \ (A \cup_0 B)$  *<proof>*

Special properties.

**lemma** *union-fun-mono*:

**assumes** *mono f*  
**shows**  $f \ A \cup_0 f \ B \subseteq_0 f \ (A \cup_0 B)$   
*<proof>*

## 2.2.10 Set difference

**definition** *vdiff* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle -_0 \rangle$  65)

where  $A \dot{-} B = \text{set } \{x. x \in_o A \wedge x \notin_o B\}$

**notation** *vdiff* (**infixl**  $\langle \dot{-} \rangle$  65)

**lemma** *small-set-vdiff[simp]*: *small*  $\{x. x \in_o A \wedge x \notin_o B\}$   
*<proof>*

Rules.

**lemma** *vdiff-iff[simp]*:  $x \in_o A \dot{-} B \longleftrightarrow x \in_o A \wedge x \notin_o B$   
*<proof>*

**lemma** *vdiffI[intro!]*:  
**assumes**  $x \in_o A$  **and**  $x \notin_o B$   
**shows**  $x \in_o A \dot{-} B$   
*<proof>*

**lemma** *vdiffD1*:  
**assumes**  $x \in_o A \dot{-} B$   
**shows**  $x \in_o A$   
*<proof>*

**lemma** *vdiffD2*:  
**assumes**  $x \in_o A \dot{-} B$  **and**  $x \in_o B$   
**shows**  $P$   
*<proof>*

**lemma** *vdiffE[elim!]*:  
**assumes**  $x \in_o A \dot{-} B$  **and**  $[[x \in_o A; x \notin_o B]] \Longrightarrow P$   
**shows**  $P$   
*<proof>*

Previous set operations.

**lemma** *vsubset-vdiff*:  
**assumes**  $A \subseteq_o B \dot{-} C$   
**shows**  $A \subseteq_o B$   
*<proof>*

**lemma** *vinsert-vdiff-nin[simp]*:  
**assumes**  $a \notin_o B$   
**shows**  $vinsert\ a\ (A \dot{-} B) = vinsert\ a\ A \dot{-} B$   
*<proof>*

Set operations.

**lemma** *vdiff-vempty-left[simp]*:  $0 \dot{-} A = 0$  *<proof>*

**lemma** *vdiff-vempty-right[simp]*:  $A \dot{-} 0 = A$  *<proof>*

**lemma** *vdiff-vsingleton-vinsert[simp]*:  $\text{set } \{a\} \dot{-} vinsert\ a\ A = 0$  *<proof>*

**lemma** *vdiff-vsingleton-in[simp]*:  
**assumes**  $a \in_o A$   
**shows**  $\text{set } \{a\} \dot{-} A = 0$   
*<proof>*

**lemma** *vdiff-vsingleton-nin[simp]*:  
**assumes**  $a \notin_o A$   
**shows**  $\text{set } \{a\} \dot{-} A = \text{set } \{a\}$   
*<proof>*

**lemma** *vdiff-vinsert-singleton[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]*:  $x \in\circ vinsert\ y\ A \leftrightarrow x = y \vee x \in\circ A$  ⟨proof⟩

**lemma** *vinsertI1*:  $x \in\circ vinsert\ x\ A$  ⟨proof⟩

**lemma** *vinsertI2*:  
**assumes**  $x \in\circ A$   
**shows**  $x \in\circ vinsert\ y\ A$

$\langle proof \rangle$

**lemma** *vinsertE1*[*elim!*]:

**assumes**  $x \in_o vinsert\ y\ A$  **and**  $x = y \implies P$  **and**  $x \in_o A \implies P$

**shows**  $P$

$\langle proof \rangle$

**lemma** *vinsertCI*[*intro!*]:

**assumes**  $x \notin_o A \implies x = y$

**shows**  $x \in_o vinsert\ y\ A$

$\langle proof \rangle$

Elementary properties.

**lemma** *vinsert-insert*:  $vinsert\ a\ A = set\ (insert\ a\ (elts\ A))$   $\langle proof \rangle$

**lemma** *vinsert-commute*:  $vinsert\ a\ (vinsert\ b\ C) = vinsert\ b\ (vinsert\ a\ C)$

$\langle proof \rangle$

**lemma** *vinsert-ident*:

**assumes**  $x \notin_o A$  **and**  $x \notin_o B$

**shows**  $vinsert\ x\ A = vinsert\ x\ B \iff A = B$

$\langle proof \rangle$

**lemmas** *vinsert-identD*[*dest*] = *vinsert-ident*[*THEN iffD1, rotated 2*]

**and** *vinsert-identI*[*intro*] = *vinsert-ident*[*THEN iffD2*]

Set operations.

**lemma** *vinsert-vempty*:  $vinsert\ a\ 0 = set\ \{a\}$   $\langle proof \rangle$

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

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

**lemma** *vinsert-vinsert*:  $vinsert\ a\ (vinsert\ b\ C) = set\ \{a, b\} \cup_o C$   $\langle proof \rangle$

**lemma** *vinsert-vunion-left*:  $vinsert\ a\ (A \cup_o B) = vinsert\ a\ A \cup_o B$   $\langle proof \rangle$

**lemma** *vinsert-vunion-right*:  $vinsert\ a\ (A \cup_o B) = A \cup_o vinsert\ a\ B$   $\langle proof \rangle$

**lemma** *vinsert-vintersection*:  $vinsert\ a\ (A \cap_o B) = vinsert\ a\ A \cap_o vinsert\ a\ B$

$\langle proof \rangle$

Special properties.

**lemma** *vinsert-set-insert-empty-anyI*:

**assumes**  $P\ (vinsert\ a\ 0)$

**shows**  $P\ (set\ (insert\ a\ \{\}))$

$\langle proof \rangle$

**lemma** *vinsert-set-insert-anyI*:

**assumes** *small*  $B$  **and**  $P\ (vinsert\ a\ (set\ (insert\ b\ B)))$

**shows**  $P\ (set\ (insert\ a\ (insert\ b\ B)))$

$\langle proof \rangle$

**lemma** *vinsert-set-insert-eq*:

**assumes** *small*  $B$

**shows**  $set\ (insert\ a\ (insert\ b\ B)) = vinsert\ a\ (set\ (insert\ b\ B))$

$\langle proof \rangle$

**lemma** *vsubset-vinsert*:

$A \subseteq_o \text{vinsert } x \ B \leftrightarrow (\text{if } x \in_o A \text{ then } A -_o \text{set } \{x\} \subseteq_o B \text{ else } A \subseteq_o B)$   
 ⟨proof⟩

**lemma** *vinsert-obtain-ne*:

**assumes**  $A \neq 0$   
**obtains**  $a \ A'$  where  $A = \text{vinsert } a \ A'$  and  $a \notin_o A'$   
 ⟨proof⟩

### 2.2.12 Power set

Rules.

**lemma** *VPowI*:

**assumes**  $A \subseteq_o B$   
**shows**  $A \in_o \text{VPow } B$   
 ⟨proof⟩

**lemma** *VPowD*:

**assumes**  $A \in_o \text{VPow } B$   
**shows**  $A \subseteq_o B$   
 ⟨proof⟩

**lemma** *VPowE[elim]*:

**assumes**  $A \in_o \text{VPow } B$  and  $A \subseteq_o B \implies P$   
**shows**  $P$   
 ⟨proof⟩

Elementary properties.

**lemma** *VPow-bottom*:  $0 \in_o \text{VPow } B$  ⟨proof⟩

**lemma** *VPow-top*:  $A \in_o \text{VPow } A$  ⟨proof⟩

Set operations.

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

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

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

**lemma** *VPow-mono*:

**assumes**  $A \subseteq_o B$   
**shows**  $\text{VPow } A \subseteq_o \text{VPow } B$   
 ⟨proof⟩

**lemma** *VPow-vunion-subset*:  $\text{VPow } A \cup_o \text{VPow } B \subseteq_o \text{VPow } (A \cup_o B)$  ⟨proof⟩

### 2.2.13 Singletons, using insert

Rules.

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

**lemma** *vsingletonD[dest!]*:

**assumes**  $y \in_o \text{set } \{x\}$   
**shows**  $y = x$

$\langle proof \rangle$

**lemma** *vsingleton-iff*:  $y \in_o \text{set } \{x\} \leftrightarrow y = x$   $\langle proof \rangle$

Previous set operations.

**lemma** *VPow-vdoubleton[simp]*:

$VPow (\text{set } \{a, b\}) = \text{set } \{0, \text{set } \{a\}, \text{set } \{b\}, \text{set } \{a, b\}\}$   
 $\langle proof \rangle$

**lemma** *vsubset-vinsertI*:

**assumes**  $A -_o \text{set } \{x\} \subseteq_o B$   
**shows**  $A \subseteq_o \text{vinsert } x B$   
 $\langle proof \rangle$

Special properties.

**lemma** *vsingleton-inject*:

**assumes**  $\text{set } \{x\} = \text{set } \{y\}$   
**shows**  $x = y$   
 $\langle proof \rangle$

**lemma** *vsingleton-insert-inj-eq[iff]*:

$\text{set } \{y\} = \text{vinsert } x A \leftrightarrow x = y \wedge A \subseteq_o \text{set } \{y\}$   
 $\langle proof \rangle$

**lemma** *vsingleton-insert-inj-eq'[iff]*:

$\text{vinsert } x A = \text{set } \{y\} \leftrightarrow x = y \wedge A \subseteq_o \text{set } \{y\}$   
 $\langle proof \rangle$

**lemma** *vsubset-vsingletonD*:

**assumes**  $A \subseteq_o \text{set } \{x\}$   
**shows**  $A = 0 \vee A = \text{set } \{x\}$   
 $\langle proof \rangle$

**lemma** *vsubset-vsingleton-iff*:  $a \subseteq_o \text{set } \{x\} \leftrightarrow a = 0 \vee a = \text{set } \{x\}$   $\langle proof \rangle$

**lemma** *vsubset-vdiff-vinsert*:  $A \subseteq_o B -_o \text{vinsert } x C \leftrightarrow A \subseteq_o B -_o C \wedge x \notin_o A$   
 $\langle proof \rangle$

**lemma** *vunion-vsingleton-iff*:

$A \cup_o B = \text{set } \{x\} \leftrightarrow$   
 $A = 0 \wedge B = \text{set } \{x\} \vee A = \text{set } \{x\} \wedge B = 0 \vee A = \text{set } \{x\} \wedge B = \text{set } \{x\}$   
 $\langle proof \rangle$

**lemma** *vsingleton-Un-iff*:

$\text{set } \{x\} = A \cup_o B \leftrightarrow$   
 $A = 0 \wedge B = \text{set } \{x\} \vee A = \text{set } \{x\} \wedge B = 0 \vee A = \text{set } \{x\} \wedge B = \text{set } \{x\}$   
 $\langle proof \rangle$

**lemma** *VPow-vsingleton-iff[simp]*:  $VPow X = \text{set } \{Y\} \leftrightarrow X = 0 \wedge Y = 0$   
 $\langle proof \rangle$

## 2.2.14 Intersection of elements

**lemma** *small-VInter[simp]*:

**assumes**  $A \neq 0$   
**shows**  $\text{small } \{a. \forall x \in_o A. a \in_o x\}$   
 $\langle proof \rangle$

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

Rules.

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

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

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

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

**lemma** *VInterE1[elim]*:  
**assumes**  $a \in_{\circ} \cap_{\circ} A$  **and**  $x \notin_{\circ} A \implies R$  **and**  $a \in_{\circ} x \implies R$   
**shows**  $R$   
 ⟨proof⟩

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

**lemma** *VInterE3*:  
**assumes**  $a \in_{\circ} \cap_{\circ} A$  **and**  $(\bigwedge y. y \in_{\circ} A \implies a \in_{\circ} y) \implies P$   
**shows**  $P$   
 ⟨proof⟩

Elementary properties.

**lemma** *VInter-Inter*:  $\cap_{\circ} A = \text{set } (\bigcap (\text{elts } '(\text{elts } A)))$   
 ⟨proof⟩

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

Set operations.

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

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

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

**lemma** *VInter-antimono*:

**assumes**  $B \neq 0$  and  $B \subseteq_0 A$   
**shows**  $\bigcap_0 A \subseteq_0 \bigcap_0 B$   
 ⟨proof⟩

**lemma** *VInter-vsubset*:  
**assumes**  $\bigwedge x. x \in_0 A \implies x \subseteq_0 B$  and  $A \neq 0$   
**shows**  $\bigcap_0 A \subseteq_0 B$   
 ⟨proof⟩

**lemma** *VInter-vinsert*:  
**assumes**  $A \neq 0$   
**shows**  $\bigcap_0 (\text{vinsert } a \ A) = a \cap_0 \bigcap_0 A$   
 ⟨proof⟩

**lemma** *VInter-vunion*:  
**assumes**  $A \neq 0$  and  $B \neq 0$   
**shows**  $\bigcap_0 (A \cup_0 B) = \bigcap_0 A \cap_0 \bigcap_0 B$   
 ⟨proof⟩

**lemma** *VInter-vintersection*:  
**assumes**  $A \cap_0 B \neq 0$   
**shows**  $\bigcap_0 A \cup_0 \bigcap_0 B \subseteq_0 \bigcap_0 (A \cap_0 B)$   
 ⟨proof⟩

**lemma** *VInter-VPow*:  $\bigcap_0 (\text{VPow } A) \subseteq_0 \text{VPow } (\bigcap_0 A)$  ⟨proof⟩

Elementary properties.

**lemma** *VInter-lower*:  
**assumes**  $x \in_0 A$   
**shows**  $\bigcap_0 A \subseteq_0 x$   
 ⟨proof⟩

**lemma** *VInter-greatest*:  
**assumes**  $A \neq 0$  and  $\bigwedge x. x \in_0 A \implies B \subseteq_0 x$   
**shows**  $B \subseteq_0 \bigcap_0 A$   
 ⟨proof⟩

### 2.2.15 Union of elements

**lemma** *Union-eq-VUnion*:  $\bigcup (\text{elts } ' \text{elts } A) = \{a. \exists x \in_0 A. a \in_0 x\}$  ⟨proof⟩

**lemma** *small-VUnion[simp]*: *small*  $\{a. \exists x \in_0 A. a \in_0 x\}$   
 ⟨proof⟩

**lemma** *VUnion-def*:  $\bigcup_0 A = \text{set } \{a. \exists x \in_0 A. a \in_0 x\}$   
 ⟨proof⟩

Rules.

**lemma** *VUnion-iff[simp]*:  $A \in_0 \bigcup_0 C \longleftrightarrow (\exists x \in_0 C. A \in_0 x)$  ⟨proof⟩

**lemma** *VUnionI[intro]*:  
**assumes**  $x \in_0 A$  and  $a \in_0 x$   
**shows**  $a \in_0 \bigcup_0 A$   
 ⟨proof⟩

**lemma** *VUnionE[elim!]*:  
**assumes**  $a \in_0 \bigcup_0 A$  and  $\bigwedge x. a \in_0 x \implies x \in_0 A \implies R$   
**shows**  $R$

*<proof>*

Elementary properties.

**lemma** *VUnion-Union*:  $\bigcup_{\circ} A = \text{set } (\bigcup (\text{elts } ' (\text{elts } A)))$   
*<proof>*

Set operations.

**lemma** *VUnion-venempty[simp]*:  $\bigcup_{\circ} 0 = 0$  *<proof>*

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

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

**lemma** *VUnion-mono*:

**assumes**  $A \subseteq_{\circ} B$

**shows**  $\bigcup_{\circ} A \subseteq_{\circ} \bigcup_{\circ} B$

*<proof>*

**lemma** *VUnion-vinsert*:  $\bigcup_{\circ} (\text{vinsert } x A) = x \cup_{\circ} \bigcup_{\circ} A$  *<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]*:  $\bigcup_{\circ} (\text{VPow } A) = A$  *<proof>*

Special properties.

**lemma** *VUnion-venempty-conv-left*:  $0 = \bigcup_{\circ} A \iff (\forall x \in_{\circ} A. x = 0)$  *<proof>*

**lemma** *VUnion-venempty-conv-right*:  $\bigcup_{\circ} A = 0 \iff (\forall x \in_{\circ} A. x = 0)$  *<proof>*

**lemma** *vsubset-VPow-VUnion*:  $A \subseteq_{\circ} \text{VPow } (\bigcup_{\circ} A)$  *<proof>*

**lemma** *VUnion-vsubsetI*:

**assumes**  $\bigwedge x. x \in_{\circ} A \implies \exists y. y \in_{\circ} B \wedge x \subseteq_{\circ} y$

**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**  $\bigwedge x. x \in_{\circ} A \implies x \subseteq_{\circ} B$

**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**  $\text{elts } (\text{set } x) = x$

*<proof>*

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

**assumes** *small*  $\{a. P a\}$   
**shows** *small*  $\{\langle a, b \rangle \mid a. P a\}$   
 $\langle \text{proof} \rangle$

*vpairs*

**definition** *vpairs*  $:: V \Rightarrow V$  **where**  
*vpairs*  $r = \text{set } \{x. x \in_o r \wedge (\exists a b. x = \langle a, b \rangle)\}$

**lemma** *small-vpairs[simp]*: *small*  $\{\langle a, b \rangle \mid a b. \langle a, b \rangle \in_o r\}$   
 $\langle \text{proof} \rangle$

Rules.

**lemma** *vpairsI[intro]*:  
**assumes**  $x \in_o r$  **and**  $x = \langle a, b \rangle$   
**shows**  $x \in_o \text{vpairs } r$   
 $\langle \text{proof} \rangle$

**lemma** *vpairsD[dest]*:  
**assumes**  $x \in_o \text{vpairs } r$   
**shows**  $x \in_o r$  **and**  $\exists a b. x = \langle a, b \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *vpairsE[elim]*:  
**assumes**  $x \in_o \text{vpairs } r$   
**obtains**  $a b$  **where**  $x = \langle a, b \rangle$  **and**  $\langle a, b \rangle \in_o r$   
 $\langle \text{proof} \rangle$

**lemma** *vpairs-iff*:  $x \in_o \text{vpairs } r \longleftrightarrow x \in_o r \wedge (\exists a b. x = \langle a, b \rangle)$   $\langle \text{proof} \rangle$

Elementary properties.

**lemma** *vpairs-iff-elts*:  $\langle a, b \rangle \in_o \text{vpairs } r \longleftrightarrow \langle a, b \rangle \in_o r$   $\langle \text{proof} \rangle$

**lemma** *vpairs-iff-pairs*:  $\langle a, b \rangle \in_o \text{vpairs } r \longleftrightarrow \langle a, b \rangle \in \text{pairs } r$   
 $\langle \text{proof} \rangle$

Set operations.

**lemma** *vpairs-vempty[simp]*:  $\text{vpairs } 0 = 0$   $\langle \text{proof} \rangle$

**lemma** *vpairs-vsingleton[simp]*:  $\text{vpairs } (\text{set } \{\langle a, b \rangle\}) = \text{set } \{\langle a, b \rangle\}$   $\langle \text{proof} \rangle$

**lemma** *vpairs-vinsert*:  $\text{vpairs } (\text{vinsert } \langle a, b \rangle A) = \text{set } \{\langle a, b \rangle\} \cup_o \text{vpairs } A$   
 $\langle \text{proof} \rangle$

**lemma** *vpairs-mono*:  
**assumes**  $r \subseteq_o s$   
**shows**  $\text{vpairs } r \subseteq_o \text{vpairs } s$   
 $\langle \text{proof} \rangle$

**lemma** *vpairs-vunion*:  $\text{vpairs } (A \cup_o B) = \text{vpairs } A \cup_o \text{vpairs } B$   $\langle \text{proof} \rangle$

**lemma** *vpairs-vintersection*:  $\text{vpairs } (A \cap_o B) = \text{vpairs } A \cap_o \text{vpairs } B$   $\langle \text{proof} \rangle$

**lemma** *vpairs-vdiff*:  $\text{vpairs } (A -_o B) = \text{vpairs } A -_o \text{vpairs } B$   $\langle \text{proof} \rangle$

Special properties.

**lemma** *vpairs-ex-vfst*:

**assumes**  $x \in_o \text{vpairs } r$   
**shows**  $\exists b. \langle \text{fst } x, b \rangle \in_o r$   
 $\langle \text{proof} \rangle$

**lemma** *vpairs-ex-vsnd*:  
**assumes**  $y \in_o \text{vpairs } r$   
**shows**  $\exists a. \langle a, \text{vsnd } y \rangle \in_o r$   
 $\langle \text{proof} \rangle$

### 2.2.17 Cartesian products

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

**lemma** *vtimes-vsubset-VPowVPow*:  $A \times_o B \subseteq_o \text{VPow } (\text{VPow } (A \cup_o B))$   
 $\langle \text{proof} \rangle$

### 2.2.18 Pairwise

**definition** *vpairwise* ::  $(V \Rightarrow V \Rightarrow \text{bool}) \Rightarrow V \Rightarrow \text{bool}$   
**where** *vpairwise*  $R S \leftrightarrow (\forall x \in_o S. \forall y \in_o S. x \neq y \longrightarrow R x y)$

Rules.

**lemma** *vpairwiseI[intro?]*:  
**assumes**  $\bigwedge x y. x \in_o S \Longrightarrow y \in_o S \Longrightarrow x \neq y \Longrightarrow R x y$   
**shows** *vpairwise*  $R S$   
 $\langle \text{proof} \rangle$

**lemma** *vpairwiseD[dest]*:  
**assumes** *vpairwise*  $R S$  **and**  $x \in_o S$  **and**  $y \in_o S$  **and**  $x \neq y$   
**shows**  $R x y$  **and**  $R y x$   
 $\langle \text{proof} \rangle$

Elementary properties.

**lemma** *vpairwise-trivial[simp]*: *vpairwise*  $(\lambda i j. j \neq i) I$   
 $\langle \text{proof} \rangle$

Set operations.

**lemma** *vpairwise-vempty[simp]*: *vpairwise*  $P 0$   $\langle \text{proof} \rangle$

**lemma** *vpairwise-vsingleton[simp]*: *vpairwise*  $P (\text{set } \{A\})$   
 $\langle \text{proof} \rangle$

**lemma** *vpairwise-vinsert*:  
*vpairwise*  $r (\text{vinsert } x s) \leftrightarrow$   
 $(\forall y. y \in_o s \wedge y \neq x \longrightarrow r x y \wedge r y x) \wedge \text{vpairwise } r s$   
 $\langle \text{proof} \rangle$

**lemma** *vpairwise-vsubset*:  
**assumes** *vpairwise*  $P S$  **and**  $T \subseteq_o S$   
**shows** *vpairwise*  $P T$   
 $\langle \text{proof} \rangle$

**lemma** *vpairwise-mono*:  
**assumes** *vpairwise*  $P A$  **and**  $\bigwedge x y. P x y \Longrightarrow Q x y$  **and**  $B \subseteq_o A$   
**shows** *vpairwise*  $Q B$   
 $\langle \text{proof} \rangle$

### 2.2.19 Disjoint sets

**abbreviation**  $\text{vdisjnt} :: V \Rightarrow V \Rightarrow \text{bool}$   
**where**  $\text{vdisjnt } A \ B \equiv A \cap_o B = 0$

Elementary properties.

**lemma**  $\text{vdisjnt-sym}$ :  
**assumes**  $\text{vdisjnt } A \ B$   
**shows**  $\text{vdisjnt } B \ A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-iff}$ :  $\text{vdisjnt } A \ B \longleftrightarrow (\forall x. \sim (x \in_o A \wedge x \in_o B))$   $\langle \text{proof} \rangle$

Set operations.

**lemma**  $\text{vdisjnt-vempty1[simp]}$ :  $\text{vdisjnt } 0 \ A$   
**and**  $\text{vdisjnt-vempty2[simp]}$ :  $\text{vdisjnt } A \ 0$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-singleton0[simp]}$ :  $\text{vdisjnt } (\text{set } \{a\}) \ (\text{set } \{b\}) \longleftrightarrow a \neq b$   
**and**  $\text{vdisjnt-singleton1[simp]}$ :  $\text{vdisjnt } (\text{set } \{a\}) \ A \longleftrightarrow a \notin_o A$   
**and**  $\text{vdisjnt-singleton2[simp]}$ :  $\text{vdisjnt } A \ (\text{set } \{a\}) \longleftrightarrow a \notin_o A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vinsert-left}$ :  $\text{vdisjnt } (\text{vinsert } a \ X) \ Y \longleftrightarrow a \notin_o Y \wedge \text{vdisjnt } X \ Y$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vinsert-right}$ :  $\text{vdisjnt } Y \ (\text{vinsert } a \ X) \longleftrightarrow a \notin_o Y \wedge \text{vdisjnt } Y \ X$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vsubset-left}$ :  
**assumes**  $\text{vdisjnt } X \ Y$  **and**  $Z \subseteq_o X$   
**shows**  $\text{vdisjnt } Z \ Y$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vsubset-right}$ :  
**assumes**  $\text{vdisjnt } X \ Y$  **and**  $Z \subseteq_o Y$   
**shows**  $\text{vdisjnt } X \ Z$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vunion-left}$ :  $\text{vdisjnt } (A \cup_o B) \ C \longleftrightarrow \text{vdisjnt } A \ C \wedge \text{vdisjnt } B \ C$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-vunion-right}$ :  $\text{vdisjnt } C \ (A \cup_o B) \longleftrightarrow \text{vdisjnt } C \ A \wedge \text{vdisjnt } C \ B$   
 $\langle \text{proof} \rangle$

Special properties.

**lemma**  $\text{vdisjnt-vemptyI[intro]}$ :  
**assumes**  $\bigwedge x. x \in_o A \implies x \in_o B \implies \text{False}$   
**shows**  $\text{vdisjnt } A \ B$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjnt-self-iff-vempty[simp]}$ :  $\text{vdisjnt } S \ S \longleftrightarrow S = 0$   $\langle \text{proof} \rangle$

**lemma**  $\text{vdisjntI}$ :  
**assumes**  $\bigwedge x \ y. x \in_o A \implies y \in_o B \implies x \neq y$   
**shows**  $\text{vdisjnt } A \ B$   
 $\langle \text{proof} \rangle$

**lemma** *vdisjnt-nin-right*:

**assumes** *vdisjnt*  $A$   $B$  **and**  $a \in_o A$

**shows**  $a \notin_o B$

*<proof>*

**lemma** *vdisjnt-nin-left*:

**assumes** *vdisjnt*  $B$   $A$  **and**  $a \in_o A$

**shows**  $a \notin_o B$

*<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* ( $\langle \cdot \rangle_{\mathbb{N}}$ ) [999] 999

**named-theorems** *nat-omega-simps*

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

**abbreviation** (*input*) *vpfst* **where** *vpfst*  $a \equiv a(0)$

**abbreviation** (*input*) *vpsnd* **where** *vpsnd*  $a \equiv a(1_{\mathbb{N}})$

**abbreviation** (*input*) *vpthrd* **where** *vpthrd*  $a \equiv a(2_{\mathbb{N}})$

### 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*]:  $\text{succ } (a_{\mathbb{N}}) = (\text{Suc } a)_{\mathbb{N}}$  (*proof*)

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

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

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

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

#### Transfer

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

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*  $\omega$ ) (*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*  $\implies$  *cr-omega*) *succ* *Suc*  
 ⟨*proof*⟩

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

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

**lemma** *ord-of-nat-card-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*rel-set* (=)  $\implies$  *cr-omega*) ( $\lambda x.$  *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*:  $0 = 0_{\mathbf{N}}$  ⟨*proof*⟩

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

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

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

**lemma** *omega-prev*:  
**assumes**  $n \in_{\circ} \omega$  **and**  $0 \in_{\circ} n$   
**obtains** *k* **where**  $n = \text{succ } k$   
 ⟨*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**  $n \in_o \omega$  **and**  $\wedge x. [[x \in_o \omega; \wedge y. y \in_o x \implies P y]] \implies P x$   
**shows**  $P n$   
*<proof>*

**lemma** *omega-induct*[*consumes 1, case-names 0 succ*]:  
**assumes**  $n \in_o \omega$  **and**  $P 0$  **and**  $\wedge n. [[n \in_o \omega; P n]] \implies P (\text{succ } n)$   
**shows**  $P n$   
*<proof>*

### 2.3.5 Methods

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

**lemma** *in-succE*:  
**assumes**  $a \in_o \text{succ } n$  **and**  $\wedge a. a \in_o n \implies P a$  **and**  $P n$   
**shows**  $P a$   
*<proof>*

**method** *Suc-of-numeral* =  
 (  
   *unfold numeral.simps add.assoc,*  
   *use nothing in <unfold Suc-eq-plus1-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*:  $1_{\mathbb{N}} = \text{set } \{0\}$  *<proof>*

**lemma two:**  $2_{\mathbf{N}} = \text{set } \{0, 1_{\mathbf{N}}\}$  *<proof>*

**lemma three:**  $3_{\mathbf{N}} = \text{set } \{0, 1_{\mathbf{N}}, 2_{\mathbf{N}}\}$  *<proof>*

**lemma four:**  $4_{\mathbf{N}} = \text{set } \{0, 1_{\mathbf{N}}, 2_{\mathbf{N}}, 3_{\mathbf{N}}\}$  *<proof>*

**lemma two-vdiff-zero[simp]:**  $\text{set } \{0, 1_{\mathbf{N}}\} -\circ \text{set } \{0\} = \text{set } \{1_{\mathbf{N}}\}$  *<proof>*

**lemma two-vdiff-one[simp]:**  $\text{set } \{0, 1_{\mathbf{N}}\} -\circ \text{set } \{1_{\mathbf{N}}\} = \text{set } \{0\}$  *<proof>*

## 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**  $vid-on A = set \{ \langle a, a \rangle \mid a. a \in_o A \}$

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

Rules.

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

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

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

**lemma**  $vid-onE[elim!]$ :  
**assumes**  $x \in_o vid-on A$  **and**  $\exists a \in_o A. x = \langle a, a \rangle \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma**  $vid-on-iff$ :  $\langle a, b \rangle \in_o vid-on A \iff a = b \wedge a \in_o A$   $\langle proof \rangle$

Set operations.

**lemma**  $vid-on-vempty[simp]$ :  $vid-on 0 = 0$   $\langle proof \rangle$

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

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

**lemma**  $vid-on-mono$ :  
**assumes**  $A \subseteq_o B$   
**shows**  $vid-on A \subseteq_o vid-on B$   
 $\langle proof \rangle$

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

*<proof>*

**lemma** *vid-on-vintersection*:  $\text{vid-on } (A \cap_o B) = \text{vid-on } A \cap_o \text{vid-on } B$  *<proof>*

**lemma** *vid-on-vunion*:  $\text{vid-on } (A \cup_o B) = \text{vid-on } A \cup_o \text{vid-on } B$  *<proof>*

**lemma** *vid-on-vidiff*:  $\text{vid-on } (A -_o B) = \text{vid-on } A -_o \text{vid-on } B$  *<proof>*

Special properties.

**lemma** *vid-on-vsubset-vtimes*:  $\text{vid-on } A \subseteq_o A \times_o A$  *<proof>*

**lemma** *VLambda-id[simp]*:  $VLambda A \text{ id} = \text{vid-on } A$   
*<proof>*

### Constant function

**definition** *vconst-on* ::  $V \Rightarrow V \Rightarrow V$   
**where**  $vconst-on A c = \text{set } \{ \langle a, c \rangle \mid a. a \in_o A \}$

**lemma** *small-vconst-on[simp]*:  $\text{small } \{ \langle a, c \rangle \mid a. a \in_o A \}$   
*<proof>*

Rules.

**lemma** *vconst-onI[intro!]*:  
**assumes**  $a \in_o A$   
**shows**  $\langle a, c \rangle \in_o vconst-on A c$   
*<proof>*

**lemma** *vconst-onD[dest!]*:  
**assumes**  $\langle a, c \rangle \in_o vconst-on A c$   
**shows**  $a \in_o A$   
*<proof>*

**lemma** *vconst-onE[elim!]*:  
**assumes**  $x \in_o vconst-on A c$   
**obtains**  $a$  **where**  $a \in_o A$  **and**  $x = \langle a, c \rangle$   
*<proof>*

**lemma** *vconst-on-iff*:  $\langle a, c \rangle \in_o vconst-on A c \longleftrightarrow a \in_o A$  *<proof>*

Set operations.

**lemma** *vconst-on-vempty[simp]*:  $vconst-on 0 c = 0$   
*<proof>*

**lemma** *vconst-on-vsingleton[simp]*:  $vconst-on (\text{set } \{a\}) c = \text{set } \{ \langle a, c \rangle \}$  *<proof>*

**lemma** *vconst-on-vdoubleton[simp]*:  $vconst-on (\text{set } \{a, b\}) c = \text{set } \{ \langle a, c \rangle, \langle b, c \rangle \}$   
*<proof>*

**lemma** *vconst-on-mono*:  
**assumes**  $A \subseteq_o B$   
**shows**  $vconst-on A c \subseteq_o vconst-on B c$   
*<proof>*

**lemma** *vconst-on-vinsert*:  
 $(vinsert \langle a, c \rangle (vconst-on A c)) = (vconst-on (vinsert a A) c)$   
*<proof>*

**lemma** *vconst-on-vintersection*:

$$vconst-on (A \cap_o B) c = vconst-on A c \cap_o vconst-on B c$$

*<proof>*

**lemma** *vconst-on-vunion*:  $vconst-on (A \cup_o B) c = vconst-on A c \cup_o vconst-on B c$

*<proof>*

**lemma** *vconst-on-vdiff*:  $vconst-on (A -_o B) c = vconst-on A c -_o vconst-on B c$

*<proof>*

Special properties.

**lemma** *vconst-on-eq-vtimes*:  $vconst-on A c = A \times_o set \{c\}$

*<proof>*

*VLambda*

Rules.

**lemma** *VLambdaI[intro!]*:

**assumes**  $a \in_o A$

**shows**  $\langle a, f a \rangle \in_o (\lambda a \in_o A. f a)$

*<proof>*

**lemma** *VLambdaD[dest!]*:

**assumes**  $\langle a, f a \rangle \in_o (\lambda a \in_o A. f a)$

**shows**  $a \in_o A$

*<proof>*

**lemma** *VLambdaE[elim!]*:

**assumes**  $x \in_o (\lambda a \in_o A. f a)$

**obtains**  $a$  **where**  $a \in_o A$  **and**  $x = \langle a, f a \rangle$

*<proof>*

**lemma** *VLambda-iff1*:  $x \in_o (\lambda a \in_o A. f a) \longleftrightarrow (\exists a \in_o A. x = \langle a, f a \rangle)$  *<proof>*

**lemma** *VLambda-iff2*:  $\langle a, b \rangle \in_o (\lambda a \in_o A. f a) \longleftrightarrow b = f a \wedge a \in_o A$  *<proof>*

**lemma** *small-VLambda[simp]*:  $small \{ \langle a, f a \rangle \mid a. a \in_o A \}$  *<proof>*

**lemma** *VLambda-set-def*:  $(\lambda a \in_o A. f a) = set \{ \langle a, f a \rangle \mid a. a \in_o A \}$  *<proof>*

Set operations.

**lemma** *VLambda-vempty[simp]*:  $(\lambda a \in_o 0. f a) = 0$  *<proof>*

**lemma** *VLambda-vsingleton*:  $(\lambda a \in_o set \{a\}. f a) = set \{ \langle a, f a \rangle \}$

*<proof>*

**lemma** *VLambda-vdoubleton*:

$(\lambda a \in_o set \{a, b\}. f a) = set \{ \langle a, f a \rangle, \langle b, f b \rangle \}$

*<proof>*

**lemma** *VLambda-mono*:

**assumes**  $A \subseteq_o B$

**shows**  $(\lambda a \in_o A. f a) \subseteq_o (\lambda a \in_o B. f a)$

*<proof>*

**lemma** *VLambda-vinsert*:

$(\lambda a \in_o vinsert a A. f a) = (\lambda a \in_o set \{a\}. f a) \cup_o (\lambda a \in_o A. f a)$

*<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-vidiff*:  $(\lambda a \in_{\circ} A -_{\circ} B. f a) = (\lambda a \in_{\circ} A. f a) -_{\circ} (\lambda a \in_{\circ} B. f a)$  *<proof>*

Connections.

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

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

## Composition

**definition** *vcomp* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \circ_{\circ} \rangle$  75)

**where**  $r \circ_{\circ} s = \text{set } \{ \langle a, c \rangle \mid a \ c. \exists b. \langle a, b \rangle \in_{\circ} s \wedge \langle b, c \rangle \in_{\circ} r \}$

**notation** *vcomp* (**infixr**  $\langle \circ_{\circ} \rangle$  75)

**lemma** *vcomp-small[simp]*:  $\text{small } \{ \langle a, c \rangle \mid a \ c. \exists b. \langle a, b \rangle \in_{\circ} s \wedge \langle b, c \rangle \in_{\circ} r \}$   
*(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]*:

$(\text{vinsert } \langle a, b \rangle \ s) \circ_{\circ} r = (\text{set } \{ \langle a, b \rangle \} \circ_{\circ} r) \cup_{\circ} (s \circ_{\circ} r)$

*<proof>*

**lemma** *vcomp-vinsert-right[simp]*:

$r \circ_0 (\text{vinsert } \langle a, b \rangle s) = (r \circ_0 \text{set } \{\langle a, b \rangle\}) \cup_0 (r \circ_0 s)$   
 ⟨proof⟩

**lemma** *vcomp-vunion-left[simp]*:  $(s \cup_0 t) \circ_0 r = (s \circ_0 r) \cup_0 (t \circ_0 r)$  ⟨proof⟩

**lemma** *vcomp-vunion-right[simp]*:  $r \circ_0 (s \cup_0 t) = (r \circ_0 s) \cup_0 (r \circ_0 t)$  ⟨proof⟩

Connections.

**lemma** *vcomp-vid-on-idem[simp]*:  $\text{vid-on } A \circ_0 \text{vid-on } A = \text{vid-on } A$  ⟨proof⟩

**lemma** *vcomp-vid-on[simp]*:  $\text{vid-on } A \circ_0 \text{vid-on } B = \text{vid-on } (A \cap_0 B)$  ⟨proof⟩

**lemma** *vcomp-vconst-on-vid-on[simp]*:  $\text{vconst-on } A \ c \circ_0 \text{vid-on } A = \text{vconst-on } A \ c$   
 ⟨proof⟩

**lemma** *vcomp-VLambda-vid-on[simp]*:  $(\lambda a \in_0 A. f \ a) \circ_0 \text{vid-on } A = (\lambda a \in_0 A. f \ a)$   
 ⟨proof⟩

Special properties.

**lemma** *vcomp-vsubset-vtimes*:

**assumes**  $r \subseteq_0 B \times_0 C$  **and**  $s \subseteq_0 A \times_0 B$

**shows**  $r \circ_0 s \subseteq_0 A \times_0 C$

⟨proof⟩

**lemma** *vcomp-obtain-middle[elim]*:

**assumes**  $\langle a, c \rangle \in_0 r \circ_0 s$

**obtains**  $b$  **where**  $\langle a, b \rangle \in_0 s$  **and**  $\langle b, c \rangle \in_0 r$

⟨proof⟩

## Converse relation

**definition** *vconverse* ::  $V \Rightarrow V$

**where**  $\text{vconverse } A = (\lambda r \in_0 A. \text{set } \{\langle b, a \rangle \mid a \ b. \langle a, b \rangle \in_0 r\})$

**abbreviation** *app-vconverse*  $(\langle (-^1)_0 \rangle)$  [1000] 999

**where**  $r^{-1}_0 \equiv \text{vconverse } (\text{set } \{r\}) \ (\!|r)$

**lemma** *app-vconverse-def*:  $r^{-1}_0 = \text{set } \{\langle b, a \rangle \mid a \ b. \langle a, b \rangle \in_0 r\}$   
 ⟨proof⟩

**lemma** *vconverse-small[simp]*:  $\text{small } \{\langle b, a \rangle \mid a \ b. \langle a, b \rangle \in_0 r\}$   
 ⟨proof⟩

Rules.

**lemma** *vconverseI[intro!]*:

**assumes**  $r \in_0 A$

**shows**  $\langle r, r^{-1}_0 \rangle \in_0 \text{vconverse } A$

⟨proof⟩

**lemma** *vconverseD[dest]*:

**assumes**  $\langle r, s \rangle \in_0 \text{vconverse } A$

**shows**  $r \in_0 A$  **and**  $s = r^{-1}_0$

⟨proof⟩

**lemma** *vconverseE[elim]*:

**assumes**  $x \in_0 \text{vconverse } A$

**obtains**  $r$  **where**  $x = \langle r, r^{-1}_0 \rangle$  **and**  $r \in_0 A$

$\langle proof \rangle$

**lemma** *app-vconverseI*[*sym, intro!*]:

**assumes**  $\langle a, b \rangle \in_{\circ} r$

**shows**  $\langle b, a \rangle \in_{\circ} r^{-1}$ .

$\langle proof \rangle$

**lemma** *app-vconverseD*[*sym, dest*]:

**assumes**  $\langle a, b \rangle \in_{\circ} r^{-1}$ .

**shows**  $\langle b, a \rangle \in_{\circ} r$

$\langle proof \rangle$

**lemma** *app-vconverseE*[*elim!*]:

**assumes**  $x \in_{\circ} r^{-1}$ .

**obtains**  $a$  **and**  $b$  **where**  $x = \langle b, a \rangle$  **and**  $\langle a, b \rangle \in_{\circ} r$

$\langle proof \rangle$

**lemma** *vconverse-iff*:  $\langle b, a \rangle \in_{\circ} r^{-1} \longleftrightarrow \langle a, b \rangle \in_{\circ} r$   $\langle proof \rangle$

Set operations.

**lemma** *vconverse-vempty*[*simp*]:  $0^{-1} = 0$   $\langle proof \rangle$

**lemma** *vconverse-vsingleton*:  $(set \ \{ \langle a, b \rangle \})^{-1} = set \ \{ \langle b, a \rangle \}$   $\langle proof \rangle$

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

**lemma** *vconverse-vinsert*:  $(vinsert \ \langle a, b \rangle \ r)^{-1} = vinsert \ \langle b, a \rangle \ (r^{-1})$   $\langle proof \rangle$

**lemma** *vconverse-vintersection*:  $(r \ \cap_{\circ} \ s)^{-1} = r^{-1} \ \cap_{\circ} \ s^{-1}$   $\langle proof \rangle$

**lemma** *vconverse-vunion*:  $(r \ \cup_{\circ} \ s)^{-1} = r^{-1} \ \cup_{\circ} \ s^{-1}$   $\langle proof \rangle$

Connections.

**lemma** *vconverse-vid-on*[*simp*]:  $(vid\text{-on} \ A)^{-1} = vid\text{-on} \ A$   $\langle proof \rangle$

**lemma** *vconverse-vconst-on*[*simp*]:  $(vconst\text{-on} \ A \ c)^{-1} = set \ \{ c \} \times_{\circ} A$   $\langle proof \rangle$

**lemma** *vconverse-vcomp*:  $(r \ \circ_{\circ} \ s)^{-1} = s^{-1} \ \circ_{\circ} \ r^{-1}$   $\langle proof \rangle$

**lemma** *vconverse-vtimes*:  $(A \ \times_{\circ} \ B)^{-1} = (B \ \times_{\circ} \ A)$   $\langle proof \rangle$

### Left restriction

**definition** *vlrestriction* ::  $V \Rightarrow V$

**where** *vlrestriction*  $D =$

$VLambda \ D \ (\lambda \langle r, A \rangle. \ set \ \{ \langle a, b \rangle \mid a \ b. \ a \in_{\circ} A \ \wedge \ \langle a, b \rangle \in_{\circ} r \})$

**abbreviation** *app-vlrestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow^l_{\circ} \rangle$  80)

**where**  $r \ \uparrow^l_{\circ} \ A \equiv vlrestriction \ (set \ \{ \langle r, A \rangle \}) \ (\langle \uparrow^l_{\circ} \rangle)$

**lemma** *app-vlrestriction-def*:  $r \ \uparrow^l_{\circ} \ A = set \ \{ \langle a, b \rangle \mid a \ b. \ a \in_{\circ} A \ \wedge \ \langle a, b \rangle \in_{\circ} r \}$   
 $\langle proof \rangle$

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

Rules.

**lemma** *vlrestrictionI[intro!]*:

**assumes**  $\langle r, A \rangle \in_{\circ} D$

**shows**  $\langle \langle r, A \rangle, r \uparrow^l A \rangle \in_{\circ} \text{vlrestriction } D$

*\langle proof \rangle*

**lemma** *vlrestrictionD[dest]*:

**assumes**  $\langle \langle r, A \rangle, s \rangle \in_{\circ} \text{vlrestriction } D$

**shows**  $\langle r, A \rangle \in_{\circ} D$  **and**  $s = r \uparrow^l A$

*\langle proof \rangle*

**lemma** *vlrestrictionE[elim]*:

**assumes**  $x \in_{\circ} \text{vlrestriction } D$  **and**  $D \subseteq_{\circ} R \times_{\circ} X$

**obtains**  $r \ A$  **where**  $x = \langle \langle r, A \rangle, r \uparrow^l A \rangle$  **and**  $r \in_{\circ} R$  **and**  $A \in_{\circ} X$

*\langle proof \rangle*

**lemma** *app-vlrestrictionI[intro!]*:

**assumes**  $a \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$

**shows**  $\langle a, b \rangle \in_{\circ} r \uparrow^l A$

*\langle proof \rangle*

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

**assumes**  $\langle a, b \rangle \in_{\circ} r \uparrow^l A$

**shows**  $a \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$

*\langle proof \rangle*

**lemma** *app-vlrestrictionE[elim]*:

**assumes**  $x \in_{\circ} r \uparrow^l A$

**obtains**  $a \ b$  **where**  $x = \langle a, b \rangle$  **and**  $a \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$

*\langle proof \rangle*

Set operations.

**lemma** *vlrestriction-on-vempty[simp]*:  $r \uparrow^l 0 = 0$

*\langle proof \rangle*

**lemma** *vlrestriction-vempty[simp]*:  $0 \uparrow^l A = 0$  *\langle proof \rangle*

**lemma** *vlrestriction-vsingleton-in[simp]*:

**assumes**  $a \in_{\circ} A$

**shows**  $\text{set } \{ \langle a, b \rangle \} \uparrow^l A = \text{set } \{ \langle a, b \rangle \}$

*\langle proof \rangle*

**lemma** *vlrestriction-vsingleton-nin[simp]*:

**assumes**  $a \notin_{\circ} A$

**shows**  $\text{set } \{ \langle a, b \rangle \} \uparrow^l A = 0$

*\langle proof \rangle*

**lemma** *vlrestriction-mono*:

**assumes**  $A \subseteq_{\circ} B$

**shows**  $r \uparrow^l A \subseteq_{\circ} r \uparrow^l B$

*\langle proof \rangle*

**lemma** *vlrestriction-vinsert-nin[simp]*:

**assumes**  $a \notin_{\circ} A$

**shows**  $(\text{vinsert } \langle a, b \rangle r) \uparrow^l A = r \uparrow^l A$

*\langle proof \rangle*

**lemma** *vlrestriction-vinsert-in*:

**assumes**  $a \in_{\circ} A$

**shows**  $(\text{vinsert } \langle a, b \rangle r) \uparrow^l_\circ A = \text{vinsert } \langle a, b \rangle (r \uparrow^l_\circ A)$   
 ⟨proof⟩

**lemma** *vlrestriction-vintersection*:  $(r \cap_\circ s) \uparrow^l_\circ A = r \uparrow^l_\circ A \cap_\circ s \uparrow^l_\circ A$  ⟨proof⟩

**lemma** *vlrestriction-vunion*:  $(r \cup_\circ s) \uparrow^l_\circ A = r \uparrow^l_\circ A \cup_\circ s \uparrow^l_\circ A$  ⟨proof⟩

**lemma** *vlrestriction-vidiff*:  $(r -_\circ s) \uparrow^l_\circ A = r \uparrow^l_\circ A -_\circ s \uparrow^l_\circ A$  ⟨proof⟩

Connections.

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

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

**lemma** *vlrestriction-vconst-on-commute*:

**assumes**  $x \in_\circ \text{vconst-on } A c \uparrow^l_\circ B$

**shows**  $x \in_\circ \text{vconst-on } B c \uparrow^l_\circ A$

⟨proof⟩

**lemma** *vlrestriction-vcomp[simp]*:  $(r \circ_\circ s) \uparrow^l_\circ A = r \circ_\circ (s \uparrow^l_\circ A)$  ⟨proof⟩

Previous connections.

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

**lemma** *vcomp-vconst-on*:

$r \circ_\circ (\text{vconst-on } A c) = (r \uparrow^l_\circ \text{set } \{c\}) \circ_\circ (\text{vconst-on } A c)$

⟨proof⟩

Special properties.

**lemma** *vlrestriction-vsubset-vpairs*:  $r \uparrow^l_\circ A \subseteq_\circ \text{vpairs } r$   
 ⟨proof⟩

**lemma** *vlrestriction-vsubset-rel*:  $r \uparrow^l_\circ A \subseteq_\circ r$  ⟨proof⟩

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

## Right restriction

**definition** *vrrestriction* ::  $V \Rightarrow V$

**where** *vrrestriction*  $D =$

$\text{VLambda } D (\lambda \langle r, A \rangle. \text{set } \{\langle a, b \rangle \mid a b. b \in_\circ A \wedge \langle a, b \rangle \in_\circ r\})$

**abbreviation** *app-vrrestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow^r_\circ \rangle$  80)

**where**  $r \uparrow^r_\circ A \equiv \text{vrrestriction } (\text{set } \{\langle r, A \rangle\}) (\langle r, A \rangle)$

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

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

Rules.

**lemma** *vrrestrictionI[intro!]*:

**assumes**  $\langle r, A \rangle \in_\circ D$

**shows**  $\langle \langle r, A \rangle, r \uparrow^r_\circ A \rangle \in_\circ \text{vrrestriction } D$

⟨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**  $s = r \uparrow^r_{\circ} A$   
 $\langle proof \rangle$

**lemma** *vrrestrictionE[elim]*:

**assumes**  $x \in_{\circ} vrrestriction\ D$  **and**  $D \subseteq_{\circ} R \times_{\circ} X$   
**obtains**  $r\ A$  **where**  $x = \langle\langle r, A \rangle, r \uparrow^r_{\circ} A\rangle$  **and**  $r \in_{\circ} R$  **and**  $A \in_{\circ} X$   
 $\langle proof \rangle$

**lemma** *app-vrrestrictionI[intro!]*:

**assumes**  $b \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$   
**shows**  $\langle a, b \rangle \in_{\circ} r \uparrow^r_{\circ} A$   
 $\langle proof \rangle$

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

**assumes**  $\langle a, b \rangle \in_{\circ} r \uparrow^r_{\circ} A$   
**shows**  $b \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$   
 $\langle proof \rangle$

**lemma** *app-vrrestrictionE[elim]*:

**assumes**  $x \in_{\circ} r \uparrow^r_{\circ} A$   
**obtains**  $a\ b$  **where**  $x = \langle a, b \rangle$  **and**  $b \in_{\circ} A$  **and**  $\langle a, b \rangle \in_{\circ} r$   
 $\langle proof \rangle$

Set operations.

**lemma** *vrrestriction-on-vempty[simp]*:  $r \uparrow^r_{\circ} 0 = 0$

$\langle proof \rangle$

**lemma** *vrrestriction-vempty[simp]*:  $0 \uparrow^r_{\circ} A = 0$   $\langle proof \rangle$

**lemma** *vrrestriction-vsingleton-in[simp]*:

**assumes**  $b \in_{\circ} A$   
**shows**  $set\ \{\langle a, b \rangle\} \uparrow^r_{\circ} A = set\ \{\langle a, b \rangle\}$   
 $\langle proof \rangle$

**lemma** *vrrestriction-vsingleton-nin[simp]*:

**assumes**  $b \notin_{\circ} A$   
**shows**  $set\ \{\langle a, b \rangle\} \uparrow^r_{\circ} A = 0$   
 $\langle proof \rangle$

**lemma** *vrrestriction-mono*:

**assumes**  $A \subseteq_{\circ} B$   
**shows**  $r \uparrow^r_{\circ} A \subseteq_{\circ} r \uparrow^r_{\circ} B$   
 $\langle proof \rangle$

**lemma** *vrrestriction-vinsert-nin[simp]*:

**assumes**  $b \notin_{\circ} A$   
**shows**  $(vinsert\ \langle a, b \rangle\ r) \uparrow^r_{\circ} A = r \uparrow^r_{\circ} A$   
 $\langle proof \rangle$

**lemma** *vrrestriction-vinsert-in*:

**assumes**  $b \in_{\circ} A$   
**shows**  $(vinsert\ \langle a, b \rangle\ r) \uparrow^r_{\circ} A = vinsert\ \langle a, b \rangle\ (r \uparrow^r_{\circ} A)$   
 $\langle proof \rangle$

**lemma** *vrrestriction-vintersection*:  $(r \cap_{\circ} s) \uparrow^r_{\circ} A = r \uparrow^r_{\circ} A \cap_{\circ} s \uparrow^r_{\circ} A$   $\langle proof \rangle$

**lemma** *vrrestriction-vunion*:  $(r \cup_0 s) \uparrow^r_0 A = r \uparrow^r_0 A \cup_0 s \uparrow^r_0 A$  *<proof>*

**lemma** *vrrestriction-vidiff*:  $(r -_0 s) \uparrow^r_0 A = r \uparrow^r_0 A -_0 s \uparrow^r_0 A$  *<proof>*

Connections.

**lemma** *vrrestriction-vid-on[simp]*:  $(\text{vid-on } A) \uparrow^r_0 B = \text{vid-on } (A \cap_0 B)$  *<proof>*

**lemma** *vrrestriction-vconst-on*:

**assumes**  $c \in_0 B$

**shows**  $(\text{vconst-on } A c) \uparrow^r_0 B = \text{vconst-on } A c$   
*<proof>*

**lemma** *vrrestriction-vcomp[simp]*:  $(r \circ_0 s) \uparrow^r_0 A = (r \uparrow^r_0 A) \circ_0 s$  *<proof>*

Previous connections.

**lemma** *vcomp-vid-on-rel[simp]*:  $\text{vid-on } A \circ_0 r = r \uparrow^r_0 A$   
*<proof>*

**lemma** *vcomp-vconst-on-rel*:  $(\text{vconst-on } A c) \circ_0 r = (\text{vconst-on } A c) \circ_0 (r \uparrow^r_0 A)$   
*<proof>*

**lemma** *vlrestriction-vconverse*:  $r^{-1}_0 \uparrow^l_0 A = (r \uparrow^r_0 A)^{-1}_0$  *<proof>*

**lemma** *vrrestriction-vconverse*:  $r^{-1}_0 \uparrow^r_0 A = (r \uparrow^l_0 A)^{-1}_0$  *<proof>*

Special properties.

**lemma** *vrrestriction-vsubset-rel*:  $r \uparrow^r_0 A \subseteq_0 r$  *<proof>*

**lemma** *vrrestriction-vsubset-vpairs*:  $r \uparrow^r_0 A \subseteq_0 \text{vpairs } r$  *<proof>*

## Restriction

**definition** *vrestriction* ::  $V \Rightarrow V$

**where** *vrestriction*  $D =$

$V\text{Lambda } D (\lambda \langle r, A \rangle. \text{set } \{\langle a, b \rangle \mid a b. a \in_0 A \wedge b \in_0 A \wedge \langle a, b \rangle \in_0 r\})$

**abbreviation** *app-vrestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow_0 \rangle$  80)

**where**  $r \uparrow_0 A \equiv \text{vrestriction } (\text{set } \{\langle r, A \rangle\}) (\langle \uparrow_0 \rangle)$

**lemma** *app-vrestriction-def*:

$r \uparrow_0 A = \text{set } \{\langle a, b \rangle \mid a b. a \in_0 A \wedge b \in_0 A \wedge \langle a, b \rangle \in_0 r\}$   
*<proof>*

**lemma** *vrestriction-small[simp]*:

*small*  $\{\langle a, b \rangle \mid a b. a \in_0 A \wedge b \in_0 A \wedge \langle a, b \rangle \in_0 r\}$   
*<proof>*

Rules.

**lemma** *vrestrictionI[intro!]*:

**assumes**  $\langle r, A \rangle \in_0 D$

**shows**  $\langle \langle r, A \rangle, r \uparrow_0 A \rangle \in_0 \text{vrestriction } D$   
*<proof>*

**lemma** *vrestrictionD[dest]*:

**assumes**  $\langle \langle r, A \rangle, s \rangle \in_0 \text{vrestriction } D$

**shows**  $\langle r, A \rangle \in_0 D$  **and**  $s = r \uparrow_0 A$

$\langle proof \rangle$

**lemma** *vrestrictionE[elim]*:

**assumes**  $x \in_0 \text{vrestriction } D$  **and**  $D \subseteq_0 R \times_0 X$

**obtains**  $r \vdash_0 A$  **where**  $x = \langle \langle r, A \rangle, r \vdash_0 A \rangle$  **and**  $r \in_0 R$  **and**  $A \in_0 X$

$\langle proof \rangle$

**lemma** *app-vrestrictionI[intro!]*:

**assumes**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $\langle a, b \rangle \in_0 r$

**shows**  $\langle a, b \rangle \in_0 r \vdash_0 A$

$\langle proof \rangle$

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

**assumes**  $\langle a, b \rangle \in_0 r \vdash_0 A$

**shows**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $\langle a, b \rangle \in_0 r$

$\langle proof \rangle$

**lemma** *app-vrestrictionE[elim]*:

**assumes**  $x \in_0 r \vdash_0 A$

**obtains**  $a \ b$  **where**  $x = \langle a, b \rangle$  **and**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $\langle a, b \rangle \in_0 r$

$\langle proof \rangle$

Set operations.

**lemma** *vrestriction-on-vempty[simp]*:  $r \vdash_0 0 = 0$

$\langle proof \rangle$

**lemma** *vrestriction-vempty[simp]*:  $0 \vdash_0 A = 0$   $\langle proof \rangle$

**lemma** *vrestriction-vsingleton-in[simp]*:

**assumes**  $a \in_0 A$  **and**  $b \in_0 A$

**shows**  $\text{set } \{ \langle a, b \rangle \} \vdash_0 A = \text{set } \{ \langle a, b \rangle \}$

$\langle proof \rangle$

**lemma** *vrestriction-vsingleton-nin-left[simp]*:

**assumes**  $a \notin_0 A$

**shows**  $\text{set } \{ \langle a, b \rangle \} \vdash_0 A = 0$

$\langle proof \rangle$

**lemma** *vrestriction-vsingleton-nin-right[simp]*:

**assumes**  $b \notin_0 A$

**shows**  $\text{set } \{ \langle a, b \rangle \} \vdash_0 A = 0$

$\langle proof \rangle$

**lemma** *vrestriction-mono*:

**assumes**  $A \subseteq_0 B$

**shows**  $r \vdash_0 A \subseteq_0 r \vdash_0 B$

$\langle proof \rangle$

**lemma** *vrestriction-vinsert-nin[simp]*:

**assumes**  $a \notin_0 A$  **and**  $b \notin_0 A$

**shows**  $(\text{vinsert } \langle a, b \rangle r) \vdash_0 A = r \vdash_0 A$

$\langle proof \rangle$

**lemma** *vrestriction-vinsert-in*:

**assumes**  $a \in_0 A$  **and**  $b \in_0 A$

**shows**  $(\text{vinsert } \langle a, b \rangle r) \vdash_0 A = \text{vinsert } \langle a, b \rangle (r \vdash_0 A)$

$\langle proof \rangle$

**lemma** *restriction-vintersection*:  $(r \cap_0 s) \uparrow_0 A = r \uparrow_0 A \cap_0 s \uparrow_0 A$  *<proof>*

**lemma** *restriction-vunion*:  $(r \cup_0 s) \uparrow_0 A = r \uparrow_0 A \cup_0 s \uparrow_0 A$  *<proof>*

**lemma** *restriction-vidiff*:  $(r -_0 s) \uparrow_0 A = r \uparrow_0 A -_0 s \uparrow_0 A$  *<proof>*

Connections.

**lemma** *restriction-vid-on[simp]*:  $(\text{vid-on } A) \uparrow_0 B = \text{vid-on } (A \cap_0 B)$  *<proof>*

**lemma** *restriction-vconst-on-ex*:

**assumes**  $c \in_0 B$

**shows**  $(\text{vconst-on } A \ c) \uparrow_0 B = \text{vconst-on } (A \cap_0 B) \ c$

*<proof>*

**lemma** *restriction-vconst-on-nex*:

**assumes**  $c \notin_0 B$

**shows**  $(\text{vconst-on } A \ c) \uparrow_0 B = 0$

*<proof>*

**lemma** *restriction-vcomp[simp]*:  $(r \circ_0 s) \uparrow_0 A = (r \uparrow^r_0 A) \circ_0 (s \uparrow^l_0 A)$  *<proof>*

**lemma** *restriction-vconverse*:  $r^{-1}_0 \uparrow_0 A = (r \uparrow_0 A)^{-1}_0$  *<proof>*

Previous connections.

**lemma** *vrrestriction-vrestriction[simp]*:  $(r \uparrow^r_0 A) \uparrow^l_0 A = r \uparrow_0 A$  *<proof>*

**lemma** *vlrestriction-vrrestriction[simp]*:  $(r \uparrow^l_0 A) \uparrow^r_0 A = r \uparrow_0 A$  *<proof>*

**lemma** *restriction-vlrestriction[simp]*:  $(r \uparrow_0 A) \uparrow^l_0 A = r \uparrow_0 A$  *<proof>*

**lemma** *restriction-vrrestriction[simp]*:  $(r \uparrow_0 A) \uparrow^r_0 A = r \uparrow_0 A$  *<proof>*

Special properties.

**lemma** *restriction-vsubset-vpairs*:  $r \uparrow_0 A \subseteq_0 \text{vpairs } r$  *<proof>*

**lemma** *restriction-vsubset-vtimes*:  $r \uparrow_0 A \subseteq_0 A \times_0 A$  *<proof>*

**lemma** *restriction-vsubset-rel*:  $r \uparrow_0 A \subseteq_0 r$  *<proof>*

### 2.4.3 Properties

#### Domain

**definition** *vdomain* ::  $V \Rightarrow V$

**where**  $\text{vdomain } D = (\lambda r \in_0 D. \text{set } \{a. \exists b. \langle a, b \rangle \in_0 r\})$

**abbreviation** *app-vdomain* ::  $V \Rightarrow V$  ( $\langle \mathcal{D}_0 \rangle$ )

**where**  $\mathcal{D}_0 \ r \equiv \text{vdomain } (\text{set } \{r\}) \ (\uparrow r)$

**lemma** *app-vdomain-def*:  $\mathcal{D}_0 \ r = \text{set } \{a. \exists b. \langle a, b \rangle \in_0 r\}$   
*<proof>*

**lemma** *vdomain-small[simp]*:  $\text{small } \{a. \exists b. \langle a, b \rangle \in_0 r\}$   
*<proof>*

Rules.

**lemma** *vdomainI[intro!]*:

**assumes**  $r \in_0 A$

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

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

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

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

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

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

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

Set operations.

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

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

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

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

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

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

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

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

Connections.

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

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

*<proof>*

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

*<proof>*

**lemma** *vdomain-vrestriction*:  $\mathcal{D}_\circ (r \uparrow^l A) = \mathcal{D}_\circ r \cap_\circ A$  *<proof>*

**lemma** *vdomain-vrestriction-ussubset*:

**assumes**  $A \subseteq_\circ \mathcal{D}_\circ r$

**shows**  $\mathcal{D}_\circ (r \uparrow^l A) = A$

*<proof>*

Special properties.

**lemma** *vdomain-ussubset-vtimes*:

**assumes**  $r \subseteq_\circ x \times_\circ y$

**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**  $\mathcal{R}_\circ r \equiv vrange (set \{r\}) (r)$

**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**  $\langle a, b \rangle \in_{\circ} r$   
 $\langle \text{proof} \rangle$

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

Set operations.

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

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

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

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

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

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

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

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

Connections.

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

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

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

**lemma** *vrange-VLambda*:  $\mathcal{R}_{\circ} (\lambda a \in_{\circ} A. f a) = \text{set } (f \text{ ' elts } A)$   
 $\langle \text{proof} \rangle$

**lemma** *vrange-vrestriction*:  $\mathcal{R}_{\circ} (r \upharpoonright_{\circ} A) = \mathcal{R}_{\circ} r \cap_{\circ} A$   $\langle \text{proof} \rangle$

Previous connections

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

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

Special properties.

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

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

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

**lemma** *vrange-VLambda-vsubset*:  
**assumes**  $\bigwedge x. x \in_\circ A \implies f x \in_\circ B$   
**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**  $\bigwedge x y. \langle x, y \rangle \in_\circ r \implies y \in_\circ A$   
**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**  $\mathcal{F}_\circ r \equiv \text{vfield } (\text{set } \{r\}) (r)$

**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**  $\langle r, \mathcal{F}_\circ r \rangle \in_\circ \text{vfield } A$   
 ⟨proof⟩

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

**lemma** *vfieldE[elim]*:  
**assumes**  $x \in_\circ \text{vfield } A$   
**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**  $a \in_{\circ} \mathcal{D}_{\circ} r \cup_{\circ} \mathcal{R}_{\circ} r \implies P$   
**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*:  $a \in_{\circ} \mathcal{F}_{\circ} r \iff (\exists b. \langle a, b \rangle \in_{\circ} r \vee \langle b, a \rangle \in_{\circ} r)$  ⟨proof⟩

Set operations.

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

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

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

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

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

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

## Image

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

**abbreviation** *app-vimage* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle ' \circ \rangle$  90)  
**where**  $r \text{ ' } \circ A \equiv vimage (\text{set } \{\langle r, A \rangle\}) (\langle r, A \rangle)$

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

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

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

Rules.

**lemma** *vimageI[intro!]*:  
**assumes**  $\langle r, A \rangle \in_o D$   
**shows**  $\langle \langle r, A \rangle, r \circ A \rangle \in_o \text{vimage } D$   
 ⟨*proof*⟩

**lemma** *vimageD[dest]*:  
**assumes**  $\langle \langle r, A \rangle, s \rangle \in_o \text{vimage } D$   
**shows**  $\langle r, A \rangle \in_o D$  **and**  $s = r \circ A$   
 ⟨*proof*⟩

**lemma** *vimageE[elim]*:  
**assumes**  $x \in_o \text{vimage } (R \times_o X)$   
**obtains**  $r \ A$  **where**  $x = \langle \langle r, A \rangle, r \circ A \rangle$  **and**  $r \in_o R$  **and**  $A \in_o X$   
 ⟨*proof*⟩

**lemma** *app-vimageI1*:  
**assumes**  $x \in_o \mathcal{R}_o (r \uparrow_o A)$   
**shows**  $x \in_o r \circ A$   
 ⟨*proof*⟩

**lemma** *app-vimageI2[intro]*:  
**assumes**  $\langle a, b \rangle \in_o r$  **and**  $a \in_o A$   
**shows**  $b \in_o r \circ A$   
 ⟨*proof*⟩

**lemma** *app-vimageD[dest]*:  
**assumes**  $x \in_o r \circ A$   
**shows**  $x \in_o \mathcal{R}_o (r \uparrow_o A)$   
 ⟨*proof*⟩

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

**lemma** *app-vimage-iff*:  $b \in_o r \circ A \longleftrightarrow (\exists a \in_o A. \langle a, b \rangle \in_o r)$  ⟨*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*:  $r \circ \text{set } \{a\} = \text{set } \{b. \langle a, b \rangle \in_o r\}$   
 ⟨*proof*⟩

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

**lemma** *vimage-vsingleton-nin*[*intro, simp*]:

**assumes**  $a \notin A$   
**shows**  $\text{set } \{a, b\} \circ A = 0$   
 ⟨*proof*⟩

**lemma** *vimage-vsingleton-vinsert*[*simp*]:  $\text{set } \{a, b\} \circ \text{vinsert } a \ A = \text{set } \{b\}$

⟨*proof*⟩

**lemma** *vimage-mono*:

**assumes**  $r' \subseteq r$  **and**  $A' \subseteq A$   
**shows**  $(r' \circ A') \subseteq (r \circ A)$   
 ⟨*proof*⟩

**lemma** *vimage-vinsert*:  $r \circ (\text{vinsert } a \ A) = r \circ \text{set } \{a\} \cup r \circ A$

⟨*proof*⟩

**lemma** *vimage-vunion-left*:  $(r \cup s) \circ A = r \circ A \cup s \circ A$

⟨*proof*⟩

**lemma** *vimage-vunion-right*:  $r \circ (A \cup B) = r \circ A \cup r \circ B$

⟨*proof*⟩

**lemma** *vimage-vintersection*:  $r \circ (A \cap B) \subseteq r \circ A \cap r \circ B$  ⟨*proof*⟩

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

Previous set operations.

**lemma** *VPow-vinsert*:

$VPow (\text{vinsert } a \ A) = VPow \ A \cup ((\lambda x \in VPow \ A. \text{vinsert } a \ x) \circ VPow \ A)$   
 ⟨*proof*⟩

Special properties.

**lemma** *vimage-vsingleton-iff*[*iff*]:  $b \in \text{set } \{a\} \iff \langle a, b \rangle \in r$  ⟨*proof*⟩

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

**lemma** *vcomp-vimage-vtimes-right*:

**assumes**  $r \circ Y = Z$   
**shows**  $r \circ (X \times Y) = X \times Z$   
 ⟨*proof*⟩

Connections.

**lemma** *vid-on-vimage*[*simp*]:  $\text{vid-on } A \circ B = A \cap B$

⟨*proof*⟩

**lemma** *vimage-vconst-on-ne*[*simp*]:

**assumes**  $B \cap A \neq 0$   
**shows**  $\text{vconst-on } A \ c \circ B = \text{set } \{c\}$   
 ⟨*proof*⟩

**lemma** *vimage-vconst-on-vempty*[*simp*]:

**assumes**  $\text{vdisjnt } A \ B$   
**shows**  $\text{vconst-on } A \ c \circ B = 0$   
 ⟨*proof*⟩

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

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

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

**lemma** *vcomp-vimage*:  $(r \circ_{\circ} s) \text{ ' } \circ A = r \text{ ' } \circ (s \text{ ' } \circ A)$   
 ⟨proof⟩

**lemma** *vimage-vrestriction[simp]*:  $(r \uparrow^l \circ A) \text{ ' } \circ B = r \text{ ' } \circ (A \cap_{\circ} B)$   
 ⟨proof⟩

**lemma** *vimage-vrestriction[simp]*:  $(r \uparrow^r \circ A) \text{ ' } \circ B = A \cap_{\circ} r \text{ ' } \circ B$  ⟨proof⟩

**lemma** *vimage-vrestriction[simp]*:  $(r \uparrow \circ A) \text{ ' } \circ B = A \cap_{\circ} (r \text{ ' } \circ (A \cap_{\circ} B))$  ⟨proof⟩

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

**lemma** *vimage-eq-imp-vcomp*:

**assumes**  $r \text{ ' } \circ A = s \text{ ' } \circ B$

**shows**  $(t \circ_{\circ} r) \text{ ' } \circ A = (t \circ_{\circ} s) \text{ ' } \circ B$

⟨proof⟩

Previous connections.

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

**lemma** *vcomp-VLambda*:

$(\lambda b \in_{\circ} ((\lambda a \in_{\circ} A. g a) \text{ ' } \circ A). f b) \circ_{\circ} (\lambda a \in_{\circ} A. g a) = (\lambda a \in_{\circ} A. (f \circ g) a)$

⟨proof⟩

Further special properties.

**lemma** *vimage-vsubset*:

**assumes**  $r \subseteq_{\circ} A \times_{\circ} B$

**shows**  $r \text{ ' } \circ C \subseteq_{\circ} B$

⟨proof⟩

**lemma** *vimage-vdomain-vsubset*:  $r \text{ ' } \circ A \subseteq_{\circ} r \text{ ' } \circ \mathcal{D}_{\circ} r$  ⟨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** *invimage*  $D = VLambda \ D \ (\lambda \langle r, A \rangle. r^{-1} \text{ ' } \circ A)$

**abbreviation** *app-invimage* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \text{ ' } \circ \rangle$  90)

**where**  $r \text{ ' } \circ A \equiv \text{invimage } (\text{set } \{\langle r, A \rangle\}) \ (\langle r, A \rangle)$

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

**lemma** *invimage-small[simp]*: *small*  $\{a. \exists b \in_0 A. \langle a, b \rangle \in_0 r\}$   
 ⟨proof⟩

Rules.

**lemma** *invimageI[intro!]*:  
 assumes  $\langle r, A \rangle \in_0 D$   
 shows  $\langle \langle r, A \rangle, r -' A \rangle \in_0 \text{invimage } D$   
 ⟨proof⟩

**lemma** *invimageD[dest]*:  
 assumes  $\langle \langle r, A \rangle, s \rangle \in_0 \text{invimage } D$   
 shows  $\langle r, A \rangle \in_0 D$  and  $s = r -' A$   
 ⟨proof⟩

**lemma** *invimageE[elim]*:  
 assumes  $x \in_0 \text{invimage } D$  and  $D \subseteq_0 R \times_0 X$   
 obtains  $r A$  where  $x = \langle \langle r, A \rangle, r -' A \rangle$  and  $r \in_0 R$  and  $A \in_0 X$   
 ⟨proof⟩

**lemma** *app-invimageI[intro]*:  
 assumes  $\langle a, b \rangle \in_0 r$  and  $b \in_0 A$   
 shows  $a \in_0 r -' A$   
 ⟨proof⟩

**lemma** *app-invimageD[dest]*:  
 assumes  $a \in_0 r -' A$   
 shows  $a \in_0 \mathcal{D}_0 (r \uparrow^r A)$   
 ⟨proof⟩

**lemma** *app-invimageE[elim]*:  
 assumes  $a \in_0 r -' A$   
 obtains  $b$  where  $\langle a, b \rangle \in_0 r$  and  $b \in_0 A$   
 ⟨proof⟩

**lemma** *app-invimageI1*:  
 assumes  $a \in_0 \mathcal{D}_0 (r \uparrow^r A)$   
 shows  $a \in_0 r -' A$   
 ⟨proof⟩

**lemma** *app-invimageD1*:  
 assumes  $a \in_0 r -' A$   
 shows  $a \in_0 \mathcal{D}_0 (r \uparrow^r A)$   
 ⟨proof⟩

**lemma** *app-invimageE1*:  
 assumes  $a \in_0 r -' A$  and  $a \in_0 \mathcal{D}_0 (r \uparrow^r A) \implies P$   
 shows  $P$   
 ⟨proof⟩

**lemma** *app-invimageI2*:  
 assumes  $a \in_0 r^{-1} \circ ' A$   
 shows  $a \in_0 r -' A$   
 ⟨proof⟩

**lemma** *app-invimageD2*:  
 assumes  $a \in_0 r -' A$   
 shows  $a \in_0 r^{-1} \circ ' A$   
 ⟨proof⟩

**lemma** *app-invimageE2*:

**assumes**  $a \in_o r - \cdot A$  **and**  $a \in_o r^{-1} \cdot A \implies P$

**shows**  $P$

*<proof>*

**lemma** *invimage-iff*:  $a \in_o r - \cdot A \iff (\exists b \in_o A. \langle a, b \rangle \in_o r)$  *<proof>*

**lemma** *invimage-iff1*:  $a \in_o r - \cdot A \iff a \in_o \mathcal{D}_o (r \uparrow^r_o A)$  *<proof>*

**lemma** *invimage-iff2*:  $a \in_o r - \cdot A \iff a \in_o r^{-1} \cdot A$  *<proof>*

Set operations.

**lemma** *invimage-vempty[simp]*:  $0 - \cdot A = 0$  *<proof>*

**lemma** *invimage-of-vempty[simp]*:  $r - \cdot 0 = 0$  *<proof>*

**lemma** *invimage-vsingleton-in[intro, simp]*:

**assumes**  $b \in_o A$

**shows**  $\text{set } \{\langle a, b \rangle\} - \cdot A = \text{set } \{a\}$

*<proof>*

**lemma** *invimage-vsingleton-nin[intro, simp]*:

**assumes**  $b \notin_o A$

**shows**  $\text{set } \{\langle a, b \rangle\} - \cdot A = 0$

*<proof>*

**lemma** *invimage-vsingleton-vinsert[intro, simp]*:

$\text{set } \{\langle a, b \rangle\} - \cdot \text{vinsert } b A = \text{set } \{a\}$

*<proof>*

**lemma** *invimage-mono*:

**assumes**  $r' \subseteq_o r$  **and**  $A' \subseteq_o A$

**shows**  $(r' - \cdot A') \subseteq_o (r - \cdot A)$

*<proof>*

**lemma** *invimage-vinsert*:  $r - \cdot (\text{vinsert } a A) = r - \cdot \text{set } \{a\} \cup_o r - \cdot A$

*<proof>*

**lemma** *invimage-vunion-left*:  $(r \cup_o s) - \cdot A = r - \cdot A \cup_o s - \cdot A$

*<proof>*

**lemma** *invimage-vunion-right*:  $r - \cdot (A \cup_o B) = r - \cdot A \cup_o r - \cdot B$

*<proof>*

**lemma** *invimage-vintersection*:  $r - \cdot (A \cap_o B) \subseteq_o r - \cdot A \cap_o r - \cdot B$  *<proof>*

**lemma** *invimage-vdiff*:  $r - \cdot A -_o r - \cdot B \subseteq_o r - \cdot (A -_o B)$  *<proof>*

Special properties.

**lemma** *invimage-set-def*:  $r - \cdot A = \text{set } \{a. \exists b \in_o A. \langle a, b \rangle \in_o r\}$  *<proof>*

**lemma** *invimage-eq-vdomain-vrestriction*:  $r - \cdot A = \mathcal{D}_o (r \uparrow^r_o A)$  *<proof>*

**lemma** *invimage-vrange[simp]*:  $r - \cdot \mathcal{R}_o r = \mathcal{D}_o r$

*<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]*:  $\text{vid-on } A -'\circ B = A \cap_\circ B$   
 ⟨proof⟩

**lemma** *invimage-vconst-on-vsubset-vdomain[simp]*:  $\text{vconst-on } A c -'\circ B \subseteq_\circ A$   
 ⟨proof⟩

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

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

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

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

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

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

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

Previous connections.

**lemma** *vcomp-vconst-on-rel-vtimes*:  $\text{vconst-on } A c \circ_\circ r = (r -'\circ A) \times_\circ \text{set } \{c\}$   
 ⟨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*:  $\text{vpairs } r = r$

Rules.

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

**assumes**  $\bigwedge x. x \in_{\circ} r \implies \exists a b. x = \langle a, b \rangle$   
**shows**  $v\text{pairs } r = r$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{pairs-eqD}[dest]$ :  
**assumes**  $v\text{pairs } r = r$   
**shows**  $\bigwedge x. x \in_{\circ} r \implies \exists a b. x = \langle a, b \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{pairs-eqE}[elim!]$ :  
**assumes**  $v\text{pairs } r = r$  **and**  $(\bigwedge x. x \in_{\circ} r \implies \exists a b. x = \langle a, b \rangle) \implies P$   
**shows**  $P$   
 $\langle \text{proof} \rangle$

**lemmas**  $v\text{relationI}[intro!] = v\text{relation.intro}$   
**lemmas**  $v\text{relationD}[dest!] = v\text{relation.vrelation}$

**lemma**  $v\text{relationE}[elim!]$ :  
**assumes**  $v\text{relation } r$  **and**  $(v\text{pairs } r = r) \implies P$   
**shows**  $P$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{relationE1}[elim]$ :  
**assumes**  $v\text{relation } r$  **and**  $x \in_{\circ} r$   
**obtains**  $a b$  **where**  $x = \langle a, b \rangle$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{relationD1}[dest]$ :  
**assumes**  $v\text{relation } r$  **and**  $x \in_{\circ} r$   
**shows**  $\exists a b. x = \langle a, b \rangle$   
 $\langle \text{proof} \rangle$

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

Set operations.

**lemma**  $v\text{relation-vsubset}$ :  
**assumes**  $v\text{relation } s$  **and**  $r \subseteq_{\circ} s$   
**shows**  $v\text{relation } r$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{relation-vinsert}[simp]$ :  $v\text{relation } (v\text{insert } \langle a, b \rangle r) \longleftrightarrow v\text{relation } r$   
 $\langle \text{proof} \rangle$

**lemma** (**in**  $v\text{relation}$ )  $v\text{relation-vinsertI}[intro, simp]$ :  
 $v\text{relation } (v\text{insert } \langle a, b \rangle r)$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{relation-vinsertD}[dest]$ :  
**assumes**  $v\text{relation } (v\text{insert } \langle a, b \rangle r)$   
**shows**  $v\text{relation } r$   
 $\langle \text{proof} \rangle$

**lemma**  $v\text{relation-vunion}$ :  $v\text{relation } (r \cup_{\circ} s) \longleftrightarrow v\text{relation } r \wedge v\text{relation } s$   
 $\langle \text{proof} \rangle$

**lemma** *vbrelation-vunionI*:  
**assumes** *vbrelation r and vbrelation s*  
**shows** *vbrelation (r  $\cup$  s)*  
 $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vunionD[dest]*:  
**assumes** *vbrelation (r  $\cup$  s)*  
**shows** *vbrelation r and vbrelation s*  
 $\langle$ *proof* $\rangle$

**lemma** (**in** *vbrelation*) *vbrelation-vintersectionI*: *vbrelation (r  $\cap$  s)*  
 $\langle$ *proof* $\rangle$

**lemma** (**in** *vbrelation*) *vbrelation-vidiffI*: *vbrelation (r  $-$  s)*  
 $\langle$ *proof* $\rangle$

Connections.

**lemma** *vbrelation-vempty*: *vbrelation 0*  $\langle$ *proof* $\rangle$

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

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

**lemma** *vbrelation-vid-on[simp]*: *vbrelation (vid-on A)*  $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vconst-on[simp]*: *vbrelation (vconst-on A c)*  $\langle$ *proof* $\rangle$

**lemma** *vbrelation-VLambda[simp]*: *vbrelation (VLambda A f)*  
 $\langle$ *proof* $\rangle$

**global-interpretation** *rel-VLambda*: *vbrelation  $\langle$ VLambda U f $\rangle$*   
 $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vcomp*:  
**assumes** *vbrelation r and vbrelation s*  
**shows** *vbrelation (r  $\circ$  s)*  
 $\langle$ *proof* $\rangle$

**lemma** (**in** *vbrelation*) *vbrelation-vconverse*: *vbrelation (r<sup>-1</sup>  $\circ$ )*  
 $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vrestriction[intro, simp]*: *vbrelation (r  $\upharpoonright^l$  A)*  $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vrestriction[intro, simp]*: *vbrelation (r  $\upharpoonright^r$  A)*  $\langle$ *proof* $\rangle$

**lemma** *vbrelation-vrestriction[intro, simp]*: *vbrelation (r  $\upharpoonright$  A)*  $\langle$ *proof* $\rangle$

Previous connections.

**lemma** (**in** *vbrelation*) *vconverse-vconverse[simp]*: *(r<sup>-1</sup>  $\circ$ )<sup>-1</sup>  $\circ$  = r*  
 $\langle$ *proof* $\rangle$

**lemma** *vconverse-mono[simp]*:  
**assumes** *vbrelation r and vbrelation s*  
**shows** *r<sup>-1</sup>  $\circ$   $\subseteq$  s<sup>-1</sup>  $\circ$   $\longleftrightarrow$  r  $\subseteq$  s*  
 $\langle$ *proof* $\rangle$

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

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

**lemma** (in *vbrelation*) *vconverse-uset-swap-2*:

**assumes**  $r^{-1}_\circ \subseteq_\circ s$

**shows**  $r \subseteq_\circ s^{-1}_\circ$

⟨proof⟩

**lemma** (in *vbrelation*) *vrestriction-vdomain[simp]*:  $r \uparrow^l_\circ \mathcal{D}_\circ r = r$

⟨proof⟩

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

⟨proof⟩

Special properties.

**lemma** *brel-vsubset-vtimes*:

*vbrelation*  $r \iff r \subseteq_\circ \text{set } (vfst \text{ ' elts } r) \times_\circ \text{set } (vsnd \text{ ' elts } r)$

⟨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*) *vrestriction-uset-vrange[intro, simp]*:

**assumes**  $\mathcal{D}_\circ r \subseteq_\circ A$

**shows**  $r \uparrow^l_\circ A = r$

⟨proof⟩

**lemma** (in *vbrelation*) *vrrestriction-uset-vrange[intro, simp]*:

**assumes**  $\mathcal{R}_\circ r \subseteq_\circ B$

**shows**  $r \uparrow^r_\circ B = r$

⟨proof⟩

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

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

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

Alternative forms of existing results.

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

### Simple single-valued relation

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

Rules.

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

**mk-ide rf** *vsu-def*[*unfolded vsu-axioms-def*]  
 $| \text{intro } vsuI[\text{intro}] |$   
 $| \text{dest } vsuD[\text{dest}] |$   
 $| \text{elim } vsuE[\text{elim}] |$

Set operations.

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

**lemma** *vsu-vinsertD*:  
**assumes** *vsu* (*vinsert*  $x r$ )  
**shows** *vsu*  $r$   
 $\langle \text{proof} \rangle$

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

**lemma** (**in** *vsu*) *vsu-vintersection*[*intro*, *simp*]: *vsu*  $(r \cap_\circ s)$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *vsu*) *vsu-vdiff*[*intro*, *simp*]: *vsu*  $(r -_\circ s)$   $\langle \text{proof} \rangle$

Connections.

**lemma** *vsu-vempty*[*simp*]: *vsu*  $0$   $\langle \text{proof} \rangle$

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

**global-interpretation** *rel-vsingleton*: *vsu*  $\langle \text{set } \{\langle a, b \rangle\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *vsu-vdoubleton*:

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

**lemma**  $vsv\text{-}vid\text{-}on[simp]$ :  $vsv$  (vid-on  $A$ )  $\langle proof \rangle$

**lemma**  $vsv\text{-}vconst\text{-}on[simp]$ :  $vsv$  (vconst-on  $A$   $c$ )  $\langle proof \rangle$

**lemma**  $vsv\text{-}VLambda[simp]$ :  $vsv$  ( $\lambda a \in_o A. f a$ )  $\langle proof \rangle$

**global-interpretation**  $rel\text{-}VLambda$ :  $vsv$   $\langle (\lambda a \in_o A. f a) \rangle$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-}vcomp$ :  
**assumes**  $vsv$   $r$  **and**  $vsv$   $s$   
**shows**  $vsv$  ( $r \circ_o s$ )  
 $\langle proof \rangle$

**lemma** (**in**  $vsv$ )  $vsv\text{-}vlrestriction[intro, simp]$ :  $vsv$  ( $r \uparrow^l \circ_o A$ )  
 $\langle proof \rangle$

**lemma** (**in**  $vsv$ )  $vsv\text{-}vrrestriction[intro, simp]$ :  $vsv$  ( $r \uparrow^r \circ_o A$ )  
 $\langle proof \rangle$

**lemma** (**in**  $vsv$ )  $vsv\text{-}vrestriction[intro, simp]$ :  $vsv$  ( $r \uparrow \circ_o A$ )  
 $\langle proof \rangle$

Special properties.

**lemma**  $small\text{-}vsv[simp]$ :  $small$   $\{f. vsv f \wedge \mathcal{D}_o f = A \wedge \mathcal{R}_o f \subseteq_o B\}$   
 $\langle proof \rangle$

**context**  $vsv$   
**begin**

**lemma**  $vsv\text{-}ex1$ :  
**assumes**  $a \in_o \mathcal{D}_o r$   
**shows**  $\exists! b. \langle a, b \rangle \in_o r$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-}ex1\text{-}app1$ :  
**assumes**  $a \in_o \mathcal{D}_o r$   
**shows**  $b = r(\downarrow a) \longleftrightarrow \langle a, b \rangle \in_o r$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-}ex1\text{-}app2[iff]$ :  
**assumes**  $a \in_o \mathcal{D}_o r$   
**shows**  $r(\downarrow a) = b \longleftrightarrow \langle a, b \rangle \in_o r$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-}appI[intro, simp]$ :  
**assumes**  $\langle a, b \rangle \in_o r$   
**shows**  $r(\downarrow a) = b$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-}appE$ :  
**assumes**  $r(\downarrow a) = b$  **and**  $a \in_o \mathcal{D}_o r$  **and**  $\langle a, b \rangle \in_o r \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma** *vdomain-vrange-is-veempty*:  $\mathcal{D}_\circ r = 0 \leftrightarrow \mathcal{R}_\circ r = 0$  *<proof>*

**lemma** *vsv-vrange-veempty*:

**assumes**  $\mathcal{R}_\circ r = 0$

**shows**  $r = 0$

*<proof>*

**lemma** *vsv-vdomain-veempty-vrange-veempty*:

**assumes**  $\mathcal{D}_\circ r \neq 0$

**shows**  $\mathcal{R}_\circ r \neq 0$

*<proof>*

**lemma** *vsv-vdomain-vsingleton-vrange-vsingleton*:

**assumes**  $\mathcal{D}_\circ r = \text{set } \{a\}$

**obtains**  $b$  **where**  $\mathcal{R}_\circ r = \text{set } \{b\}$

*<proof>*

**lemma** *vsv-vsubset-vimageE*:

**assumes**  $B \subseteq_\circ r \text{ ' } A$

**obtains**  $C$  **where**  $C \subseteq_\circ A$  **and**  $B = r \text{ ' } C$

*<proof>*

**lemma** *vsv-vimage-eqI[intro]*:

**assumes**  $a \in_\circ \mathcal{D}_\circ r$  **and**  $r(|a) = b$  **and**  $a \in_\circ A$

**shows**  $b \in_\circ r \text{ ' } A$

*<proof>*

**lemma** *vsv-vimageI1*:

**assumes**  $a \in_\circ \mathcal{D}_\circ r$  **and**  $a \in_\circ A$

**shows**  $r(|a) \in_\circ r \text{ ' } A$

*<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**  $b \in_\circ r \text{ ' } A$

**obtains**  $x$  **where**  $r(|x) = b$  **and**  $x \in_\circ A$

*<proof>*

**lemma** *vsv-vimage-iff*:  $b \in_\circ r \text{ ' } A \leftrightarrow (\exists a. a \in_\circ A \wedge a \in_\circ \mathcal{D}_\circ r \wedge r(|a) = b)$

*<proof>*

**lemma** *vsv-vimage-vsingleton*:

**assumes**  $a \in_\circ \mathcal{D}_\circ r$

**shows**  $r \circ \text{set } \{a\} = \text{set } \{r(a)\}$   
 ⟨proof⟩

**lemma** *vsv-vimage-vsubsetI*:

**assumes**  $\bigwedge a. [\![ a \in_{\circ} A; a \in_{\circ} \mathcal{D}_{\circ} r ]\!] \implies r(a) \in_{\circ} B$   
**shows**  $r \circ A \subseteq_{\circ} B$   
 ⟨proof⟩

**lemma** *vsv-image-vsubset-iff*:

$r \circ A \subseteq_{\circ} B \iff (\forall a \in_{\circ} A. a \in_{\circ} \mathcal{D}_{\circ} r \longrightarrow r(a) \in_{\circ} B)$   
 ⟨proof⟩

**lemma** *vsv-vimage-vinsert*:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$   
**shows**  $r \circ \text{vinsert } a A = \text{vinsert } (r(a)) (r \circ A)$   
 ⟨proof⟩

**lemma** *vsv-vinsert-vimage[intro, simp]*:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $a \in_{\circ} A$   
**shows**  $\text{vinsert } (r(a)) (r \circ A) = r \circ A$   
 ⟨proof⟩

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

⟨proof⟩

**lemma** *vsv-is-VLambda-on-vrestriction[intro, simp]*:

**assumes**  $A \subseteq_{\circ} \mathcal{D}_{\circ} r$   
**shows**  $(\lambda x \in_{\circ} A. r(x)) = r \upharpoonright_{\circ} A$   
 ⟨proof⟩

**lemma** *pairwise-vimageI*:

**assumes**  $\bigwedge x y. [\![ x \in_{\circ} \mathcal{D}_{\circ} r; y \in_{\circ} \mathcal{D}_{\circ} r; x \neq y; r(x) \neq r(y) ]\!] \implies P (r(x)) (r(y))$   
**shows** *vpairwise*  $P (\mathcal{R}_{\circ} r)$   
 ⟨proof⟩

**lemma** *vsv-vrange-vsubset*:

**assumes**  $\bigwedge x. x \in_{\circ} \mathcal{D}_{\circ} r \implies r(x) \in_{\circ} A$   
**shows**  $\mathcal{R}_{\circ} r \subseteq_{\circ} A$   
 ⟨proof⟩

**lemma** *vsv-vrestriction-vinsert*:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$   
**shows**  $r \upharpoonright_{\circ} \text{vinsert } a A = \text{vinsert } (a, r(a)) (r \upharpoonright_{\circ} A)$   
 ⟨proof⟩

**end**

**lemma** *vsv-eqI*:

**assumes** *vsv*  $r$   
**and** *vsv*  $s$   
**and**  $\mathcal{D}_{\circ} r = \mathcal{D}_{\circ} s$   
**and**  $\bigwedge a. a \in_{\circ} \mathcal{D}_{\circ} r \implies r(a) = s(a)$   
**shows**  $r = s$   
 ⟨proof⟩

**lemma** (in *vsv*) *vsv-VLambda-cong*:

**assumes**  $\bigwedge a. a \in_{\circ} \mathcal{D}_{\circ} r \implies r(a) = f a$

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

**lemma** *Axiom-of-Choice*:

**obtains**  $f$  **where**  $\bigwedge x. x \in_{\circ} A \implies x \neq 0 \implies f(|x) \in_{\circ} x$  **and**  $vsv f$   
 ⟨proof⟩

**lemma** *VLambda-eqI*:

**assumes**  $X = Y$  **and**  $\bigwedge x. x \in_{\circ} X \implies f x = g x$   
**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**  $vid-on A (|a) = b \iff a = b$   
 ⟨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**  $s(\downarrow b) = c$   
**and**  $r(\downarrow a) = b$   
**shows**  $(s \circ_{\circ} r)(\downarrow a) = c$   
 $\langle proof \rangle$

**lemma** *vcomp-atD[dest]*:  
**assumes**  $(s \circ_{\circ} r)(\downarrow a) = c$   
**and**  $vsv r$   
**and**  $vsv s$   
**and**  $a \in_{\circ} \mathcal{D}_{\circ} r$   
**and**  $r(\downarrow a) \in_{\circ} \mathcal{D}_{\circ} s$   
**shows**  $\exists b. s(\downarrow b) = c \wedge r(\downarrow a) = b$   
 $\langle proof \rangle$

**lemma** *vcomp-atE1*:  
**assumes**  $(s \circ_{\circ} r)(\downarrow a) = c$   
**and**  $vsv r$   
**and**  $vsv s$   
**and**  $a \in_{\circ} \mathcal{D}_{\circ} r$   
**and**  $r(\downarrow a) \in_{\circ} \mathcal{D}_{\circ} s$   
**and**  $\exists b. s(\downarrow b) = c \wedge r(\downarrow a) = b \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma** *vcomp-atE[elim]*:  
**assumes**  $(s \circ_{\circ} r)(\downarrow a) = c$   
**and**  $vsv r$   
**and**  $vsv s$   
**and**  $a \in_{\circ} \mathcal{D}_{\circ} r$   
**and**  $r(\downarrow a) \in_{\circ} \mathcal{D}_{\circ} s$   
**obtains**  $b$  **where**  $r(\downarrow a) = b$  **and**  $s(\downarrow b) = c$   
 $\langle proof \rangle$

**lemma** *vsv-vcomp-at[simp]*:  
**assumes**  $vsv r$  **and**  $vsv s$  **and**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $r(\downarrow a) \in_{\circ} \mathcal{D}_{\circ} s$   
**shows**  $(s \circ_{\circ} r)(\downarrow a) = s(\downarrow r(\downarrow a))$   
 $\langle proof \rangle$

**context**  $vsv$   
**begin**

Converse relation.

**lemma** *vconverse-atI[intro]*:  
**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $r(\downarrow a) = b$   
**shows**  $\langle b, a \rangle \in_{\circ} r^{-1}_{\circ}$   
 $\langle proof \rangle$

**lemma** *vconverse-atD[dest]*:  
**assumes**  $\langle b, a \rangle \in_{\circ} r^{-1}_{\circ}$   
**shows**  $r(\downarrow a) = b$   
 $\langle proof \rangle$

**lemma** *vconverse-atE[elim]*:  
**assumes**  $\langle b, a \rangle \in_{\circ} r^{-1}_{\circ}$  **and**  $r(\downarrow a) = b \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma** *vconverse-iff*:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$

**shows**  $\langle b, a \rangle \in_{\circ} r^{-1}_{\circ} \longleftrightarrow r(\downarrow a) = b$

*<proof>*

Left restriction.

**interpretation** *vlrestriction*:  $vsu \langle r \uparrow^l_{\circ} A \rangle$  *<proof>*

**lemma** *vlrestriction-atI*[*intro, simp*]:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $a \in_{\circ} A$  **and**  $r(\downarrow a) = b$

**shows**  $(r \uparrow^l_{\circ} A)(\downarrow a) = b$

*<proof>*

**lemma** *vlrestriction-atD*[*dest*]:

**assumes**  $(r \uparrow^l_{\circ} A)(\downarrow a) = b$  **and**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $a \in_{\circ} A$

**shows**  $r(\downarrow a) = b$

*<proof>*

**lemma** *vlrestriction-atE1*[*elim*]:

**assumes**  $(r \uparrow^l_{\circ} A)(\downarrow a) = b$

**and**  $a \in_{\circ} \mathcal{D}_{\circ} r$

**and**  $a \in_{\circ} A$

**and**  $r(\downarrow a) = b \implies P$

**shows**  $P$

*<proof>*

**lemma** *vlrestriction-atE2*[*elim*]:

**assumes**  $x \in_{\circ} r \uparrow^l_{\circ} A$

**obtains**  $a b$  **where**  $x = \langle a, b \rangle$  **and**  $a \in_{\circ} A$  **and**  $r(\downarrow a) = b$

*<proof>*

Right restriction.

**interpretation** *vrrestriction*:  $vsu \langle r \uparrow^r_{\circ} A \rangle$  *<proof>*

**lemma** *vrrestriction-atI*[*intro, simp*]:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $b \in_{\circ} A$  **and**  $r(\downarrow a) = b$

**shows**  $(r \uparrow^r_{\circ} A)(\downarrow a) = b$

*<proof>*

**lemma** *vrrestriction-atD*[*dest*]:

**assumes**  $(r \uparrow^r_{\circ} A)(\downarrow a) = b$  **and**  $a \in_{\circ} r^{-\prime}_{\circ} A$

**shows**  $b \in_{\circ} A$  **and**  $r(\downarrow a) = b$

*<proof>*

**lemma** *vrrestriction-atE1*[*elim*]:

**assumes**  $(r \uparrow^r_{\circ} A)(\downarrow a) = b$  **and**  $a \in_{\circ} r^{-\prime}_{\circ} A$  **and**  $r(\downarrow a) = b \implies P$

**shows**  $P$

*<proof>*

**lemma** *vrrestriction-atE2*[*elim*]:

**assumes**  $x \in_{\circ} r \uparrow^r_{\circ} A$

**obtains**  $a b$  **where**  $x = \langle a, b \rangle$  **and**  $b \in_{\circ} A$  **and**  $r(\downarrow a) = b$

*<proof>*

Restriction.

**interpretation** *vrestriction*:  $vsu \langle r \uparrow_{\circ} A \rangle$  *<proof>*

**lemma** *vrestriction-app*[*intro, simp*]:

**assumes**  $a \in_{\circ} \mathcal{D}_{\circ} r$  **and**  $a \in_{\circ} A$   
**shows**  $(r \upharpoonright_{\circ} A)(a) = r(a)$   
*<proof>*

**lemma** *vrestriction-atD*[*dest*]:

**assumes**  $(r \upharpoonright_{\circ} A)(a) = b$  **and**  $a \in_{\circ} r^{-\circ} A$  **and**  $a \in_{\circ} A$   
**shows**  $b \in_{\circ} A$  **and**  $r(a) = b$   
*<proof>*

**lemma** *vrestriction-atE1*[*elim*]:

**assumes**  $(r \upharpoonright_{\circ} A)(a) = b$   
**and**  $a \in_{\circ} r^{-\circ} A$   
**and**  $a \in_{\circ} A$   
**and**  $r(a) = b \implies P$   
**shows**  $P$   
*<proof>*

**lemma** *vrestriction-atE2*[*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**  $r(a) = b$   
*<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*:  $r^{-\circ} A = \text{set } \{b. \exists a \in_{\circ} A \cap_{\circ} \mathcal{D}_{\circ} r. r(a) = b\}$

*<proof>*

**lemma** *vimage-set-iff*:  $b \in_{\circ} r^{-\circ} A \iff (\exists a \in_{\circ} A \cap_{\circ} \mathcal{D}_{\circ} r. r(a) = b)$

*<proof>*

Further derived results.

**lemma** *vimage-image*:

**assumes**  $A \subseteq_o \mathcal{D}_o r$

**shows**  $\text{elts } (r \circ A) = (\lambda x. r(x)) \circ (\text{elts } A)$

*<proof>*

**lemma** *vsu-vinsert-match-appI*[*intro, simp*]:

**assumes**  $a \notin_o \mathcal{D}_o r$

**shows**  $\text{vinsert } \langle a, b \rangle r \downarrow a = b$

*<proof>*

**lemma** *vsu-vinsert-no-match-appI*:

**assumes**  $a \notin_o \mathcal{D}_o r$  **and**  $c \in_o \mathcal{D}_o r$  **and**  $r \downarrow c = d$

**shows**  $\text{vinsert } \langle a, b \rangle r \downarrow c = d$

*<proof>*

**lemma** *vsu-is-vconst-onI*:

**assumes**  $\mathcal{D}_o r = A$  **and**  $\mathcal{R}_o r = \text{set } \{a\}$

**shows**  $r = \text{vconst-on } A a$

*<proof>*

**lemma** *vsu-vdomain-vrange-vsingleton*:

**assumes**  $\mathcal{D}_o r = \text{set } \{a\}$  **and**  $\mathcal{R}_o r = \text{set } \{b\}$

**shows**  $r = \text{set } \{\langle a, b \rangle\}$

*<proof>*

end

Alternative forms of existing results.

**lemmas** [*intro*] = *vsu.vconverse-atI*

**and** *vsu.vconverse-atD*[*dest*] = *vsu.vconverse-atD*[*rotated*]

**and** *vsu.vconverse-atE*[*elim*] = *vsu.vconverse-atE*[*rotated*]

**and** [*intro, simp*] = *vsu.vlrestriction-atI*

**and** *vsu.vlrestriction-atD*[*dest*] = *vsu.vlrestriction-atD*[*rotated*]

**and** *vsu.vlrestriction-atE1*[*elim*] = *vsu.vlrestriction-atE1*[*rotated*]

**and** *vsu.vlrestriction-atE2*[*elim*] = *vsu.vlrestriction-atE2*[*rotated*]

**and** [*intro, simp*] = *vsu.vrrestriction-atI*

**and** *vsu.vrrestriction-atD*[*dest*] = *vsu.vrrestriction-atD*[*rotated*]

**and** *vsu.vrrestriction-atE1*[*elim*] = *vsu.vrrestriction-atE1*[*rotated*]

**and** *vsu.vrrestriction-atE2*[*elim*] = *vsu.vrrestriction-atE2*[*rotated*]

**and** [*intro, simp*] = *vsu.vlrestriction-app*

**and** *vsu.vrestriction-atD*[*dest*] = *vsu.vrestriction-atD*[*rotated*]

**and** *vsu.vrestriction-atE1*[*elim*] = *vsu.vrestriction-atE1*[*rotated*]

**and** *vsu.vrestriction-atE2*[*elim*] = *vsu.vrestriction-atE2*[*rotated*]

**and** *vsu.vdomain-atD* = *vsu.vdomain-atD*[*rotated*]

**and** *vsu.vdomain-atE* = *vsu.vdomain-atE*[*rotated*]

**and** *vrange-atD* = *vsu.vrange-atD*[*rotated*]

**and** *vrange-atE* = *vsu.vrange-atE*[*rotated*]

**and** *vsu.vinsert-match-appI*[*intro, simp*] = *vsu.vsu-vinsert-match-appI*

**and** *vsu.vinsert-no-match-appI*[*intro, simp*] =

*vsu.vsu-vinsert-no-match-appI*[*rotated 3*]

Corollaries of the alternative forms of existing results.

**lemma** *vsu-vlrestriction-vrange*:

**assumes** *vsu s* **and**  $vsu (r \downarrow_o \mathcal{R}_o s)$

**shows**  $vsv (r \circ_0 s)$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-vunion-app-right}[simp]$ :  
**assumes**  $vsv r$  **and**  $vsv s$  **and**  $vdisjnt (\mathcal{D}_0 r) (\mathcal{D}_0 s)$  **and**  $x \in_0 \mathcal{D}_0 s$   
**shows**  $(r \cup_0 s)(x) = s(x)$   
 $\langle proof \rangle$

**lemma**  $vsv\text{-vunion-app-left}[simp]$ :  
**assumes**  $vsv r$  **and**  $vsv s$  **and**  $vdisjnt (\mathcal{D}_0 r) (\mathcal{D}_0 s)$  **and**  $x \in_0 \mathcal{D}_0 r$   
**shows**  $(r \cup_0 s)(x) = r(x)$   
 $\langle proof \rangle$

### One-to-one relation

**locale**  $v11 = vsv r$  **for**  $r +$   
**assumes**  $vsv\text{-vconverse}: vsv (r^{-1}_0)$

Rules.

**lemmas** **(in**  $v11$ )  $[intro] = v11\text{-axioms}$

**mk-ide rf**  $v11\text{-def}[unfolded v11\text{-axioms-def}]$   
 $|intro v11I[intro]|$   
 $|dest v11D[dest]|$   
 $|elim v11E[elim]|$

Set operations.

**lemma** **(in**  $v11$ )  $v11\text{-vinsert}[intro, simp]$ :  
**assumes**  $a \notin_0 \mathcal{D}_0 r$  **and**  $b \notin_0 \mathcal{R}_0 r$   
**shows**  $v11 (vinsert \langle a, b \rangle r)$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vinsertD}$ :  
**assumes**  $v11 (vinsert x r)$   
**shows**  $v11 r$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vunion}$ :  
**assumes**  $v11 r$   
**and**  $v11 s$   
**and**  $vdisjnt (\mathcal{D}_0 r) (\mathcal{D}_0 s)$   
**and**  $vdisjnt (\mathcal{R}_0 r) (\mathcal{R}_0 s)$   
**shows**  $v11 (r \cup_0 s)$   
 $\langle proof \rangle$

**lemma** **(in**  $v11$ )  $v11\text{-vintersection}[intro, simp]$ :  $v11 (r \cap_0 s)$   
 $\langle proof \rangle$

**lemma** **(in**  $v11$ )  $v11\text{-vdiff}[intro, simp]$ :  $v11 (r -_0 s)$   
 $\langle proof \rangle$

Special properties.

**lemma** **(in**  $vsv$ )  $vsv\text{-valneq-v11I}$ :  
**assumes**  $\bigwedge x y. [[ x \in_0 \mathcal{D}_0 r; y \in_0 \mathcal{D}_0 r; x \neq y ]] \implies r(x) \neq r(y)$   
**shows**  $v11 r$   
 $\langle proof \rangle$

**lemma** **(in**  $vsv$ )  $vsv\text{-valeq-v11I}$ :

**assumes**  $\wedge x y. [[ x \in_0 \mathcal{D}_0 r; y \in_0 \mathcal{D}_0 r; r(x) = r(y) ]] \implies x = y$   
**shows**  $v11\ r$   
 $\langle proof \rangle$

Connections.

**lemma**  $v11\text{-vempty}[simp]: v11\ 0\ \langle proof \rangle$

**lemma**  $v11\text{-vsingleton}[simp]: v11\ (\text{set } \{\langle a, b \rangle\})\ \langle proof \rangle$

**lemma**  $v11\text{-vdoubleton}$ :

**assumes**  $a \neq c$  **and**  $b \neq d$   
**shows**  $v11\ (\text{set } \{\langle a, b \rangle, \langle c, d \rangle\})$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vid-on}[simp]: v11\ (\text{vid-on } A)\ \langle proof \rangle$

**lemma**  $v11\text{-VLambda}[intro]$ :

**assumes**  $\text{inj-on } f\ (\text{elts } A)$   
**shows**  $v11\ (\lambda a \in_0 A. f\ a)$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vcomp}$ :

**assumes**  $v11\ r$  **and**  $v11\ s$   
**shows**  $v11\ (r \circ_0 s)$   
 $\langle proof \rangle$

**context**  $v11$

**begin**

**lemma**  $v11\text{-vconverse}: v11\ (r^{-1} \circ_0)\ \langle proof \rangle$

**interpretation**  $v11\ \langle r^{-1} \circ_0 \rangle\ \langle proof \rangle$

**lemma**  $v11\text{-vlrestriction}[intro, simp]: v11\ (r \upharpoonright^l_0 A)$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vrrestriction}[intro, simp]: v11\ (r \upharpoonright^r_0 A)$   
 $\langle proof \rangle$

**lemma**  $v11\text{-vrestriction}[intro, simp]: v11\ (r \upharpoonright_0 A)$   
 $\langle proof \rangle$

**end**

Further Special properties.

**context**  $v11$

**begin**

**lemma**  $v11\text{-injective}$ :

**assumes**  $a \in_0 \mathcal{D}_0 r$  **and**  $b \in_0 \mathcal{D}_0 r$  **and**  $r(a) = r(b)$   
**shows**  $a = b$   
 $\langle proof \rangle$

**lemma**  $v11\text{-double-pair}$ :

**assumes**  $a \in_0 \mathcal{D}_0 r$  **and**  $a' \in_0 \mathcal{D}_0 r$  **and**  $r(a) = b$  **and**  $r(a') = b'$   
**shows**  $a = a' \iff b = b'$   
 $\langle proof \rangle$

**lemma** *v11-vrange-ex1-eq*:  $b \in_{\circ} \mathcal{R}_{\circ} r \leftrightarrow (\exists ! a \in_{\circ} \mathcal{D}_{\circ} r. r(|a|) = b)$   
 ⟨proof⟩

**lemma** *v11-VLambda-iff*: *inj-on*  $f$  (*elts*  $A$ )  $\leftrightarrow$  *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  $r \circ_{\circ} A \neq r \circ_{\circ} B$   
 ⟨proof⟩

**lemma** *v11-eq-iff[simp]*:  
 assumes  $a \in_{\circ} \mathcal{D}_{\circ} r$  and  $b \in_{\circ} \mathcal{D}_{\circ} r$   
 shows  $r(|a|) = r(|b|) \leftrightarrow a = b$   
 ⟨proof⟩

**lemma** *v11-vcomp-vconverse*:  $r^{-1} \circ_{\circ} r = \text{vid-on } (\mathcal{D}_{\circ} r)$   
 ⟨proof⟩

**lemma** *v11-vcomp-vconverse'*:  $r \circ_{\circ} r^{-1} = \text{vid-on } (\mathcal{R}_{\circ} r)$   
 ⟨proof⟩

**lemma** *v11-vconverse-app[simp]*:  
 assumes  $r(|a|) = b$  and  $a \in_{\circ} \mathcal{D}_{\circ} r$   
 shows  $r^{-1} \circ_{\circ} (|b|) = a$   
 ⟨proof⟩

**lemma** *v11-vconverse-app-in-vdomain*:  
 assumes  $y \in_{\circ} \mathcal{R}_{\circ} r$   
 shows  $r^{-1} \circ_{\circ} (|y|) \in_{\circ} \mathcal{D}_{\circ} r$   
 ⟨proof⟩

**lemma** *v11-app-if-vconverse-app*:  
 assumes  $y \in_{\circ} \mathcal{R}_{\circ} r$  and  $r^{-1} \circ_{\circ} (|y|) = x$   
 shows  $r(|x|) = y$   
 ⟨proof⟩

**lemma** *v11-app-vconverse-app*:  
 assumes  $a \in_{\circ} \mathcal{R}_{\circ} r$   
 shows  $r(|r^{-1} \circ_{\circ} (|a|)|) = a$   
 ⟨proof⟩

**lemma** *v11-vconverse-app-app*:  
 assumes  $a \in_{\circ} \mathcal{D}_{\circ} r$   
 shows  $r^{-1} \circ_{\circ} (|r(|a|)|) = a$   
 ⟨proof⟩

**end**

**lemma** *v11-vrestriction-vsubset*:  
 assumes *v11*  $(f \uparrow_{\circ} A)$  and  $B \subseteq_{\circ} A$   
 shows *v11*  $(f \uparrow_{\circ} B)$   
 ⟨proof⟩

**lemma** *v11-vrestriction-vrange*:  
 assumes *v11*  $s$  and *v11*  $(r \uparrow_{\circ} \mathcal{R}_{\circ} s)$   
 shows *v11*  $(r \circ_{\circ} s)$   
 ⟨proof⟩

```

lemma v11-v1-restriction-vcomp:
  assumes v11 (f  $\vdash^l$  A) and v11 (g  $\vdash^l$  (f  $\circ$  A))
  shows v11 ((g  $\circ$  f)  $\vdash^l$  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-v1-restriction
  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_VLambda_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_VLambda_thm vsv_VLambda_match vsv_VLambda_thm;
val mk_VLambda_vdomain =
  mk_VLambda_thm vdomain_VLambda_match vdomain_VLambda_thm;
val mk_VLambda_app =
  mk_VLambda_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**  $imVLambda A f \equiv (\lambda a \in_o A. f a) 'o A$

### 2.5.2 Intersection of an indexed family of sets

**syntax**  $-VIFINTER :: pttm \Rightarrow V \Rightarrow V \Rightarrow V \langle (3 \cap_o \cdot \epsilon_o \cdot / \cdot) \rangle [0, 0, 10] 10)$

**syntax-consts**  $-VIFINTER \rightleftharpoons VInter$

**translations**  $\cap_o x \in_o A. f \rightleftharpoons CONST VInter (CONST imVLambda A (\lambda x. f))$

Rules.

**lemma**  $vifintersectionI[intro]$ :  
**assumes**  $I \neq 0$  **and**  $\bigwedge i. i \in_o I \implies a \in_o f i$   
**shows**  $a \in_o (\bigcap_o i \in_o I. f i)$   
 $\langle proof \rangle$

**lemma**  $vifintersectionD[dest]$ :  
**assumes**  $a \in_o (\bigcap_o i \in_o I. f i)$  **and**  $i \in_o I$   
**shows**  $a \in_o f i$   
 $\langle proof \rangle$

**lemma**  $vifintersectionE1[elim]$ :  
**assumes**  $a \in_o (\bigcap_o i \in_o I. f i)$  **and**  $a \in_o f i \implies P$  **and**  $i \notin_o I \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma**  $vifintersectionE3[elim]$ :  
**assumes**  $a \in_o (\bigcap_o i \in_o I. f i)$   
**obtains**  $\bigwedge i. i \in_o I \implies a \in_o f i$   
 $\langle proof \rangle$

**lemma**  $vifintersectionE2[elim]$ :  
**assumes**  $a \in_o (\bigcap_o i \in_o I. f i)$   
**obtains**  $i$  **where**  $i \in_o I$  **and**  $a \in_o f i$   
 $\langle proof \rangle$

Set operations.

**lemma**  $vifintersection-vempty-is[simp]$ :  $(\bigcap_o i \in_o 0. f i) = 0$   $\langle proof \rangle$

**lemma**  $vifintersection-vsingleton-is[simp]$ :  $(\bigcap_o i \in_o set\{i\}. f i) = f i$   
 $\langle proof \rangle$

**lemma**  $vifintersection-vdoubleton-is[simp]$ :  $(\bigcap_o i \in_o set\{i, j\}. f i) = f i \cap_o f j$   
 $\langle proof \rangle$

**lemma**  $vifintersection-antimonol$ :

**assumes**  $I \neq 0$  and  $I \subseteq_o J$   
**shows**  $(\bigcap_{j \in_o J} f j) \subseteq_o (\bigcap_{i \in_o I} f i)$   
 ⟨proof⟩

**lemma** *vfintersection-antimono2*:

**assumes**  $I \neq 0$  and  $I \subseteq_o J$  and  $\bigwedge i. i \in_o I \implies f i \subseteq_o g i$   
**shows**  $(\bigcap_{j \in_o J} f j) \subseteq_o (\bigcap_{i \in_o I} g i)$   
 ⟨proof⟩

**lemma** *vfintersection-vintersection*:

**assumes**  $I \neq 0$  and  $J \neq 0$   
**shows**  $(\bigcap_{i \in_o I} f i) \cap_o (\bigcap_{i \in_o J} f i) = (\bigcap_{i \in_o I \cup_o J} f i)$   
 ⟨proof⟩

**lemma** *vfintersection-vintersection-family*:

**assumes**  $I \neq 0$   
**shows**  $(\bigcap_{i \in_o I} A i) \cap_o (\bigcap_{i \in_o I} B i) = (\bigcap_{i \in_o I} A i \cap_o B i)$   
 ⟨proof⟩

**lemma** *vfintersection-vunion*:

**assumes**  $I \neq 0$  and  $J \neq 0$   
**shows**  $(\bigcap_{i \in_o I} f i) \cup_o (\bigcap_{j \in_o J} g j) = (\bigcap_{i \in_o I \cup_o J} f i \cup_o g j)$   
 ⟨proof⟩

**lemma** *vfintersection-vinsert-is[intro, simp]*:

**assumes**  $I \neq 0$   
**shows**  $(\bigcap_{i \in_o I} \text{vinsert } j \text{ } f i) = f j \cap_o (\bigcap_{i \in_o I} f i)$   
 ⟨proof⟩

**lemma** *vfintersection-VPow*:

**assumes**  $I \neq 0$   
**shows**  $VPow (\bigcap_{i \in_o I} f i) = (\bigcap_{i \in_o I} VPow (f i))$   
 ⟨proof⟩

Elementary properties.

**lemma** *vfintersection-constant[intro, simp]*:

**assumes**  $I \neq 0$   
**shows**  $(\bigcap_{y \in_o I} c) = c$   
 ⟨proof⟩

**lemma** *vfintersection-vsubset-iff*:

**assumes**  $I \neq 0$   
**shows**  $A \subseteq_o (\bigcap_{i \in_o I} f i) \iff (\forall i \in_o I. A \subseteq_o f i)$   
 ⟨proof⟩

**lemma** *vfintersection-vsubset-lower*:

**assumes**  $i \in_o I$   
**shows**  $(\bigcap_{i \in_o I} f i) \subseteq_o f i$   
 ⟨proof⟩

**lemma** *vfintersection-vsubset-greatest*:

**assumes**  $I \neq 0$  and  $\bigwedge i. i \in_o I \implies A \subseteq_o f i$   
**shows**  $A \subseteq_o (\bigcap_{i \in_o I} f i)$   
 ⟨proof⟩

**lemma** *vfintersection-vintersection-value*:

**assumes**  $i \in_o I$   
**shows**  $f i \cap_o (\bigcap_{i \in_o I} f i) = (\bigcap_{i \in_o I} f i)$

*<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*  $\Rightarrow V \Rightarrow V \Rightarrow V$  ( $\langle (3 \cup_{\circ} \text{-}\epsilon_{\circ} \text{-}/ \text{-}) \rangle [0, 0, 10] 10$ )

**syntax-consts** -VIFUNION  $\hat{=} VUnion$

**translations**  $\bigcup_{\circ} x \in_{\circ} A. f \hat{=} CONST VUnion (CONST imVLambda A (\lambda x. f))$

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]*:  $(\bigcup_{\circ} i \in_{\circ} \text{set } \{i\}. f i) = f i$  *<proof>*

**lemma** *vifunion-vsingleton-family[simp]*:  $(\bigcup_{\circ} i \in_{\circ} I. \text{set } \{i\}) = I$  *<proof>*

**lemma** *vifunion-vdoubleton*:  $(\bigcup_{\circ} i \in_{\circ} \text{set } \{i, j\}. f i) = f i \cup_{\circ} f j$

*<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*:

$$(\bigcup_{i \in_o I} f i) \cap_o (\bigcup_{j \in_o J} g j) = (\bigcup_{i \in_o I} \bigcup_{j \in_o J} f i \cap_o g j)$$

*<proof>*

**lemma** *vifunion-vinsert-is*:

$$(\bigcup_{i \in_o I} \text{vinsert } j \ I. f i) = f j \cup_o (\bigcup_{i \in_o I} f i)$$

*<proof>*

**lemma** *vifunion-VPow*:  $(\bigcup_{i \in_o I} \text{VPow } (f i)) \subseteq_o \text{VPow } (\bigcup_{i \in_o I} f i)$  *<proof>*

Elementary properties.

**lemma** *vifunion-vempty-conv*:

$$0 = (\bigcup_{i \in_o I} f i) \iff (\forall i \in_o I. f i = 0)$$

$$(\bigcup_{i \in_o I} f i) = 0 \iff (\forall i \in_o I. f i = 0)$$

*<proof>*

**lemma** *vifunion-constant[simp]*:  $(\bigcup_{i \in_o I} c) = (\text{if } I = 0 \text{ then } 0 \text{ else } c)$

*<proof>*

**lemma** *vifunion-upper*:

**assumes**  $i \in_o I$   
**shows**  $f i \subseteq_o (\bigcup_{i \in_o I} f i)$   
*<proof>*

**lemma** *vifunion-least*:

**assumes**  $\bigwedge i. i \in_o I \implies f i \subseteq_o C$   
**shows**  $(\bigcup_{i \in_o I} f i) \subseteq_o C$   
*<proof>*

**lemma** *vifunion-absorb*:

**assumes**  $j \in_o I$   
**shows**  $f j \cup_o (\bigcup_{i \in_o I} f i) = (\bigcup_{i \in_o I} f i)$   
*<proof>*

**lemma** *vifunion-vifunion-flatten*:

$$(\bigcup_{j \in_o I} (\bigcup_{i \in_o I} f i). g j) = (\bigcup_{i \in_o I} \bigcup_{j \in_o I} f i. g j)$$

*<proof>*

**lemma** *vifunion-vsubset-iff*:  $((\bigcup_{i \in_o I} f i) \subseteq_o B) = (\forall i \in_o I. f i \subseteq_o B)$  *<proof>*

**lemma** *vifunion-vsingleton-eq-vrange*:  $(\bigcup_{i \in_o I} \text{set } \{f i\}) = \mathcal{R}_o (\lambda a \in_o I. f a)$

*<proof>*

**lemma** *vball-vifunion[simp]*:  $(\forall z \in_o (\bigcup_{i \in_o I} f i). P z) \iff (\forall x \in_o I. \forall z \in_o f x. P z)$

*<proof>*

**lemma** *vbox-vifunion[simp]*:  $(\exists z \in_o (\bigcup_{i \in_o I} f i). P z) \iff (\exists x \in_o I. \exists z \in_o f x. P z)$

*<proof>*

**lemma** *vifunion-vintersection-index-right[simp]*:  $(\bigcup_{C \in_o B} A \cap_o C) = A \cap_o \bigcup_o B$

*<proof>*

**lemma** *vifunion-vintersection-index-left[simp]*:  $(\bigcup_{C \in_o B} C \cap_o A) = \bigcup_o B \cap_o A$

*<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*:  $(\bigcup_{\circ} C \cap_{\circ} A = 0) \leftrightarrow (\forall B \in_{\circ} C. vdisjnt B A)$   
 ⟨proof⟩

**lemma** *vdisjnt-vifunion-iff*:  
 $vdisjnt A (\bigcup_{\circ} i \in_{\circ} I. f i) \leftrightarrow (\forall i \in_{\circ} I. vdisjnt A (f i))$   
 ⟨proof⟩

**lemma** *vifunion-VLambda*:  $(\bigcup_{\circ} i \in_{\circ} A. set \{\{i, f i\}\}) = (\lambda a \in_{\circ} A. f a)$   
 ⟨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 *vsu*) *vsu-vrange-vsubset-vifunion-app*:  
**assumes**  $\mathcal{D}_{\circ} r = I$  **and**  $\bigwedge i. i \in_{\circ} I \implies r(i) \in_{\circ} A$   
**shows**  $\mathcal{R}_{\circ} r \subseteq_{\circ} (\bigcup_{\circ} i \in_{\circ} I. A i)$   
 ⟨proof⟩

**lemma** *v11-vrestriction-vifintersection*:  
**assumes**  $I \neq 0$  **and**  $\bigwedge i. i \in_{\circ} I \implies v11 (f \uparrow_{\circ} (A i))$   
**shows**  $v11 (f \uparrow_{\circ} (\bigcap_{\circ} i \in_{\circ} I. A i))$   
 ⟨proof⟩

## 2.5.4 Additional simplification rules for indexed families of sets.

Union.

**lemma** *vifunion-simps[simp]*:  
 $\bigwedge a B I. (\bigcup_{\circ} i \in_{\circ} I. vinsert a (B i)) =$   
 (if  $I=0$  then 0 else  $vinsert a (\bigcup_{\circ} i \in_{\circ} I. B i)$ )  
 $\bigwedge A B I. (\bigcup_{\circ} i \in_{\circ} I. A i \cup_{\circ} B) = ((if I=0 then 0 else (\bigcup_{\circ} i \in_{\circ} I. A i) \cup_{\circ} B))$   
 $\bigwedge A B I. (\bigcup_{\circ} i \in_{\circ} I. A i \cup_{\circ} B i) = ((if I=0 then 0 else A \cup_{\circ} (\bigcup_{\circ} i \in_{\circ} I. B i)))$   
 $\bigwedge A B I. (\bigcup_{\circ} i \in_{\circ} I. A i \cap_{\circ} B) = ((\bigcup_{\circ} i \in_{\circ} I. A i) \cap_{\circ} B)$   
 $\bigwedge A B I. (\bigcup_{\circ} i \in_{\circ} I. A i \cap_{\circ} B i) = (A \cap_{\circ} (\bigcup_{\circ} i \in_{\circ} I. B i))$   
 $\bigwedge A B I. (\bigcup_{\circ} i \in_{\circ} I. A i -_{\circ} B) = ((\bigcup_{\circ} i \in_{\circ} I. A i) -_{\circ} B)$   
 $\bigwedge A B. (\bigcup_{\circ} i \in_{\circ} \bigcup_{\circ} A. B i) = (\bigcup_{\circ} y \in_{\circ} A. \bigcup_{\circ} i \in_{\circ} y. B i)$   
 ⟨proof⟩

**lemma** *vifunion-simps-ext*:  
 $\bigwedge a B I. vinsert a (\bigcup_{\circ} i \in_{\circ} I. B i) =$   
 (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*]:

$$\begin{aligned} \wedge A P. (\forall a \in_o \cup_o A. P a) &\longleftrightarrow (\forall y \in_o A. \forall a \in_o y. P a) \\ \wedge A P. (\exists a \in_o \cup_o A. P a) &\longleftrightarrow (\exists y \in_o A. \exists a \in_o y. P a) \\ \langle \text{proof} \rangle \end{aligned}$$

Intersection.

**lemma** *vifintersection-simps*[*simp*]:

$$\begin{aligned} \wedge I A B. (\cap_o i \in_o I. A i \cap_o B) &= (\text{if } I = 0 \text{ then } 0 \text{ else } (\cap_o i \in_o I. A i) \cap_o B) \\ \wedge I A B. (\cap_o i \in_o I. A \cap_o B i) &= (\text{if } I = 0 \text{ then } 0 \text{ else } A \cap_o (\cap_o i \in_o I. B i)) \\ \wedge I A B. (\cap_o i \in_o I. A i \neg_o B) &= (\text{if } I = 0 \text{ then } 0 \text{ else } (\cap_o i \in_o I. A i) \neg_o B) \\ \wedge I A B. (\cap_o i \in_o I. A \neg_o B i) &= (\text{if } I = 0 \text{ then } 0 \text{ else } A \neg_o (\cup_o i \in_o I. B i)) \\ \wedge I a B. \\ (\cap_o i \in_o I. \text{vinsert } a (B i)) &= (\text{if } I = 0 \text{ then } 0 \text{ else } \text{vinsert } a (\cap_o i \in_o I. B i)) \\ \wedge I A B. (\cap_o i \in_o I. A i \cup_o B) &= (\text{if } I = 0 \text{ then } 0 \text{ else } ((\cap_o i \in_o I. A i) \cup_o B)) \\ \wedge I A B. (\cap_o i \in_o I. A \cup_o B i) &= (\text{if } I = 0 \text{ then } 0 \text{ else } (A \cup_o (\cap_o i \in_o I. B i))) \\ \langle \text{proof} \rangle \end{aligned}$$

**lemma** *vifintersection-simps-ext*:

$$\begin{aligned} \wedge A B I. (\cap_o i \in_o I. A i) \cap_o B &= (\text{if } I = 0 \text{ then } 0 \text{ else } (\cap_o i \in_o I. A i \cap_o B)) \\ \wedge A B I. A \cap_o (\cap_o i \in_o I. B i) &= (\text{if } I = 0 \text{ then } 0 \text{ else } (\cap_o i \in_o I. A \cap_o B i)) \\ \wedge A B I. (\cap_o i \in_o I. A i) \neg_o B &= (\text{if } I = 0 \text{ then } 0 \text{ else } (\cap_o i \in_o I. A i \neg_o B)) \\ \wedge A B I. A \neg_o (\cup_o i \in_o I. B i) &= (\text{if } I = 0 \text{ then } A \text{ else } (\cap_o i \in_o I. A \neg_o B i)) \\ \wedge a B I. \text{vinsert } a (\cap_o i \in_o I. B i) &= \\ (\text{if } I = 0 \text{ then set } \{a\} \text{ else } &(\cap_o i \in_o I. \text{vinsert } a (B i))) \\ \wedge A B I. ((\cap_o i \in_o I. A i) \cup_o B) &= (\text{if } I = 0 \text{ then } B \text{ else } (\cap_o i \in_o I. A i \cup_o B)) \\ \wedge A B I. A \cup_o (\cap_o i \in_o I. B i) &= (\text{if } I = 0 \text{ then } A \text{ else } (\cap_o i \in_o I. A \cup_o B i)) \\ \langle \text{proof} \rangle \end{aligned}$$

### 2.5.5 Knowledge transfer: union and intersection of indexed families

**lemma** *SUP-vifunion*:  $(\text{SUP } \xi \in \text{elts } \alpha. A \xi) = (\cup_o \xi \in_o \alpha. A \xi)$

$\langle \text{proof} \rangle$

**lemma** *INF-vifintersection*:  $(\text{INF } \xi \in \text{elts } \alpha. A \xi) = (\cap_o \xi \in_o \alpha. A \xi)$

$\langle \text{proof} \rangle$

**lemmas** *Ord-induct3'*[*consumes 1, case-names 0 succ Limit, induct type: V*] = *Ord-induct3*[*unfolded SUP-vifunion*]

**lemma** *Limit-vifunion-def*[*simp*]:

$$\begin{aligned} \text{assumes } & \text{Limit } \alpha \\ \text{shows } & (\cup_o \xi \in_o \alpha. \xi) = \alpha \\ \langle \text{proof} \rangle \end{aligned}$$

**lemmas-with**[*unfolded SUP-vifunion INF-vifintersection*]:

$$\begin{aligned} \text{TC} &= \text{ZFC-Cardinals.TC} \\ \text{and rank-Sup} &= \text{ZFC-Cardinals.rank-Sup} \\ \text{and TC-def} &= \text{ZFC-Cardinals.TC-def} \\ \text{and Ord-equality} &= \text{ZFC-in-HOL.Ord-equality} \\ \text{and Aleph-Limit} &= \text{ZFC-Cardinals.Aleph-Limit} \\ \text{and rank} &= \text{ZFC-Cardinals.rank} \\ \text{and Vset} &= \text{ZFC-in-HOL.Vset} \\ \text{and mult} &= \text{Kirby.mult} \\ \text{and Aleph-def} &= \text{ZFC-Cardinals.Aleph-def} \\ \text{and times-V-def} &= \text{Kirby.times-V-def} \\ \text{and mult-Limit} &= \text{Kirby.mult-Limit} \\ \text{and Vfrom} &= \text{ZFC-in-HOL.Vfrom} \end{aligned}$$

**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**  $-VSIGMA :: ptrn \Rightarrow V \Rightarrow V \Rightarrow V \langle (3 \prod \circ -\epsilon \circ - / -) \rangle [0, 0, 10] 10$

**syntax-consts**  $-VSIGMA \Rightarrow VSigma$

**translations**  $\prod \circ i \in \circ I. A \Rightarrow CONST VSigma I (\lambda i. A)$

Further rules.

**lemma**  $vdunion-def: (\prod \circ i \in \circ I. A i) = (\bigcup \circ i \in \circ I. \bigcup \circ x \in \circ A i. set \{ \langle i, x \rangle \})$   
 $\langle proof \rangle$

**lemma**  $vdunionI:$

**assumes**  $ix = \langle i, x \rangle$  **and**  $i \in \circ I$  **and**  $x \in \circ A i$   
**shows**  $ix \in \circ (\prod \circ i \in \circ I. A i)$   
 $\langle proof \rangle$

**lemmas**  $vdunionD = VSigmaD1 VSigmaD2$   
**and**  $vdunionE = VSigmaE$

Set operations.

**lemma**  $vdunion-vsingleton: (\prod \circ i \in \circ set \{ c \}. A i) = set \{ c \} \times \circ A c \langle proof \rangle$

**lemma**  $vdunion-vdoubleton:$

$(\prod \circ i \in \circ set \{ a, b \}. A i) = set \{ a \} \times \circ A a \cup \circ set \{ b \} \times \circ A b$   
 $\langle proof \rangle$

Connections.

**lemma**  $vdunion-vsum: (\prod \circ i \in \circ set \{ 0, 1 \}. if i=0 then A else B) = A \uplus B$   
 $\langle proof \rangle$

### 2.5.7 Canonical injection

**definition**  $vcinjection :: (V \Rightarrow V) \Rightarrow V \Rightarrow V$   
**where**  $vcinjection A i = (\lambda x \in \circ A i. \langle i, x \rangle)$

Rules.

**mk-VLambda**  $vcinjection-def$   
 $| vsv vcinjection-vsuv[intro] |$   
 $| vdomain vcinjection-vdomain[simp] |$   
 $| app vcinjection-app[simp, intro] |$

Elementary results.

**lemma**  $vcinjection-vrange-uset:$   
**assumes**  $i \in \circ I$

**shows**  $\mathcal{R}_\circ (vcinjection\ A\ i) \subseteq_\circ (\prod_\circ i \in_\circ I. A\ i)$   
 ⟨proof⟩

**lemma** *vcinjection-vrange*:

**assumes**  $i \in_\circ I$  **and**  $\bigwedge j. j \in_\circ I \implies A\ j \neq 0$

**shows**  $\mathcal{R}_\circ (vcinjection\ A\ i) = (\bigcup_\circ x \in_\circ A\ i. set\ \{\langle i, x \rangle\})$

⟨proof⟩

## 2.5.8 Infinite Cartesian product

**definition** *vproduct* ::  $V \Rightarrow (V \Rightarrow V) \Rightarrow V$

**where**  $vproduct\ I\ A = set\ \{f. vsv\ f \wedge \mathcal{D}_\circ f = I \wedge (\forall i \in_\circ I. f(i) \in_\circ A\ i)\}$

**syntax** *-VPRODUCT* ::  $pttrn \Rightarrow V \Rightarrow V \Rightarrow V$  (⟨(3 $\prod_\circ$ - $\epsilon_\circ$ -./- )⟩ [0, 0, 10] 10)

**syntax-consts** *-VPRODUCT*  $\Rightarrow$  *vproduct*

**translations**  $\prod_\circ i \in_\circ I. A \Rightarrow CONST\ vproduct\ I\ (\lambda i. A)$

**lemma** *small-vproduct[simp]*:

*small*  $\{f. vsv\ f \wedge \mathcal{D}_\circ f = I \wedge (\forall i \in_\circ I. f(i) \in_\circ A\ i)\}$

(**is** ⟨small ?A⟩)

⟨proof⟩

Rules.

**lemma** *vproductI[intro!]*:

**assumes**  $vsv\ f$  **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**  $vsv\ f$

**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**  $vsv\ f$  **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]*:  $(\prod_\circ i \in_\circ 0. A\ i) = set\ \{0\}$

⟨proof⟩

**lemma** *vproduct-vsingletonI*:

**assumes**  $f(c) \in_\circ A\ c$  **and**  $f = set\ \{\langle c, f(c) \rangle\}$

**shows**  $f \in_\circ (\prod_\circ i \in_\circ set\ \{c\}. A\ i)$

⟨proof⟩

**lemma** *vproduct-vsingletonD*:

**assumes**  $f \in_\circ (\prod_\circ i \in_\circ set\ \{c\}. A\ i)$

**shows**  $vsv\ f$  **and**  $f(c) \in_\circ A\ c$  **and**  $f = set\ \{\langle c, f(c) \rangle\}$

⟨proof⟩

**lemma** *vproduct-vsingletonE*:

**assumes**  $f \in_\circ (\prod_\circ i \in_\circ set\ \{c\}. A\ i)$

**obtains**  $vsv f$  and  $f(\langle c \rangle) \in_0 A c$  and  $f = set \{ \langle c, f(\langle c \rangle) \rangle \}$   
 ⟨proof⟩

**lemma** *vproduct-vsingleton-iff*:

$f \in_0 (\prod_{i \in_0 set \{c\}} A i) \longleftrightarrow f(\langle c \rangle) \in_0 A c \wedge f = set \{ \langle c, f(\langle c \rangle) \rangle \}$   
 ⟨proof⟩

**lemma** *vproduct-vdoubletonI[intro]*:

**assumes**  $vsv f$   
 and  $f(\langle a \rangle) \in_0 A a$   
 and  $f(\langle b \rangle) \in_0 A b$   
 and  $\mathcal{D}_0 f = set \{a, b\}$   
 and  $\mathcal{R}_0 f \subseteq_0 A a \cup_0 A b$   
**shows**  $f \in_0 (\prod_{i \in_0 set \{a, b\}} A i)$   
 ⟨proof⟩

**lemma** *vproduct-vdoubletonD[dest]*:

**assumes**  $f \in_0 (\prod_{i \in_0 set \{a, b\}} A i)$   
**shows**  $vsv f$   
 and  $f(\langle a \rangle) \in_0 A a$   
 and  $f(\langle b \rangle) \in_0 A b$   
 and  $\mathcal{D}_0 f = set \{a, b\}$   
 and  $f = set \{ \langle a, f(\langle a \rangle) \rangle, \langle b, f(\langle b \rangle) \rangle \}$   
 ⟨proof⟩

**lemma** *vproduct-vdoubletonE*:

**assumes**  $f \in_0 (\prod_{i \in_0 set \{a, b\}} A i)$   
**obtains**  $vsv f$   
 and  $f(\langle a \rangle) \in_0 A a$   
 and  $f(\langle b \rangle) \in_0 A b$   
 and  $\mathcal{D}_0 f = set \{a, b\}$   
 and  $f = set \{ \langle a, f(\langle a \rangle) \rangle, \langle b, f(\langle b \rangle) \rangle \}$   
 ⟨proof⟩

**lemma** *vproduct-vdoubleton-iff*:

$f \in_0 (\prod_{i \in_0 set \{a, b\}} A i) \longleftrightarrow$   
 $vsv f \wedge$   
 $f(\langle a \rangle) \in_0 A a \wedge$   
 $f(\langle b \rangle) \in_0 A b \wedge$   
 $\mathcal{D}_0 f = set \{a, b\} \wedge$   
 $f = set \{ \langle a, f(\langle a \rangle) \rangle, \langle b, f(\langle b \rangle) \rangle \}$   
 ⟨proof⟩

Elementary properties.

**lemma** *vproduct-eq-vemptyI*:

**assumes**  $i \in_0 I$  and  $A i = 0$   
**shows**  $(\prod_{i \in_0 I} A i) = 0$   
 ⟨proof⟩

**lemma** *vproduct-eq-vemptyD*:

**assumes**  $(\prod_{i \in_0 I} A i) \neq 0$   
**shows**  $\bigwedge i. i \in_0 I \implies A i \neq 0$   
 ⟨proof⟩

**lemma** *vproduct-vrange*:

**assumes**  $f \in_0 (\prod_{i \in_0 I} A i)$   
**shows**  $\mathcal{R}_0 f \subseteq_0 (\bigcup_{i \in_0 I} A i)$   
 ⟨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**  $\bigwedge i. i \in_{\circ} I \implies f \ i \in_{\circ} A \ i$

**shows**  $(\lambda i \in_{\circ} I. f \ i) \in_{\circ} (\prod_{\circ} i \in_{\circ} I. A \ i)$

⟨proof⟩

**lemma** *vproduct-cong*:

**assumes**  $\bigwedge i. i \in_{\circ} I \implies f \ i = g \ i$

**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**  $\bigwedge i. i \in_{\circ} I \implies A \ i \neq 0$

**obtains**  $X$  **where**  $X \in_{\circ} (\prod_{\circ} i \in_{\circ} I. A \ i)$  **and**  $x = X \uparrow_{\circ}^l J$

⟨proof⟩

**lemma** *vproduct-usingleton-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**  $f(\!|i\!) \in_{\circ} \bigcup_{\circ} (\bigcup_{\circ} (A))$

⟨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*

|*vsu vprojection-vsuv[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**  $\bigwedge j. j \in_{\circ} I \implies A \ j \neq 0$

**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**  $\langle \hat{\ }_{\times} \rangle$  80)

**where**  $A \ \hat{\ }_{\times} \ n = (\prod_{\circ} i \in_{\circ} n. A)$

Rules.

**lemma** *vcpowerI[intro]*:

**assumes**  $f \in_{\circ} (\prod_{\circ} i \in_{\circ} n. A)$

**shows**  $f \in_{\circ} (A \ \hat{\ }_{\times} \ n)$

*<proof>*

**lemma** *vcpowerD[dest]*:  
**assumes**  $f \in_{\circ} (A \hat{\times} n)$   
**shows**  $f \in_{\circ} (\prod_{\circ i \in_{\circ} n. A})$   
*<proof>*

**lemma** *vcpowerE[elim]*:  
**assumes**  $f \in_{\circ} (A \hat{\times} n)$  **and**  $f \in_{\circ} (\prod_{\circ i \in_{\circ} n. A) \implies P$   
**shows**  $P$   
*<proof>*

Set operations.

**lemma** *vcpower-index-vempty[simp]*:  $A \hat{\times} 0 = \text{set } \{0\}$   
*<proof>*

**lemma** *vcpower-of-vempty*:  
**assumes**  $n \neq 0$   
**shows**  $0 \hat{\times} n = 0$   
*<proof>*

**lemma** *vcpower-vsubset-mono*:  
**assumes**  $A \subseteq_{\circ} B$   
**shows**  $A \hat{\times} n \subseteq_{\circ} B \hat{\times} n$   
*<proof>*

Connections.

**lemma** *vcpower-vdomain*:  
**assumes**  $f \in_{\circ} (A \hat{\times} n)$   
**shows**  $\mathcal{D}_{\circ} f = n$   
*<proof>*

**lemma** *vcpower-vrange*:  
**assumes**  $f \in_{\circ} (A \hat{\times} n)$   
**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**  $\langle \approx_\circ \rangle$  50)  
**where**  $A \approx_\circ B \equiv elts A \approx elts B$

Rules

**lemma** (**in**  $v11$ )  $v11\text{-veqpollI}$ [*intro*]:  
**assumes**  $\mathcal{D}_\circ r = A$  **and**  $\mathcal{R}_\circ r = B$   
**shows**  $A \approx_\circ B$   
 $\langle proof \rangle$

**lemmas**  $v11\text{-veqpollI}$ [*intro*] =  $v11.v11\text{-veqpollI}$

**lemma**  $v11\text{-veqpollE}$ [*elim*]:  
**assumes**  $A \approx_\circ B$   
**obtains**  $f$  **where**  $v11 f$  **and**  $\mathcal{D}_\circ f = A$  **and**  $\mathcal{R}_\circ f = B$   
 $\langle proof \rangle$

Set operations.

**lemma**  $veqpoll\text{-vsingleton}$ :  $set \{x\} \approx_\circ set \{y\}$   
 $\langle proof \rangle$

**lemma**  $veqpoll\text{-vinsert}$ :  
**assumes**  $A \approx_\circ B$  **and**  $a \notin_\circ A$  **and**  $b \notin_\circ B$   
**shows**  $vinsert a A \approx_\circ vinsert b B$   
 $\langle proof \rangle$

**lemma**  $veqpoll\text{-pair}$ :  
**assumes**  $a \neq b$  **and**  $c \neq d$   
**shows**  $set \{a, b\} \approx_\circ set \{c, d\}$   
 $\langle proof \rangle$

**lemma**  $veqpoll\text{-vpair}$ :  
**assumes**  $a \neq b$  **and**  $c \neq d$   
**shows**  $\langle a, b \rangle \approx_\circ \langle c, d \rangle$   
 $\langle proof \rangle$

### 2.6.3 *vlepoll*

**abbreviation**  $vlepoll :: V \Rightarrow V \Rightarrow bool$  (**infixl**  $\langle \lesssim_\circ \rangle$  50)  
**where**  $A \lesssim_\circ B \equiv elts A \lesssim elts B$

Set operations.

**lemma**  $vlepoll\text{-vsubset}$ :  
**assumes**  $A \subseteq_\circ B$   
**shows**  $A \lesssim_\circ B$   
 $\langle proof \rangle$

Special properties.

**lemma** *vlepoll-singleton-vinsert*:  $set \{x\} \lesssim_o vinsert\ y\ A$   
 ⟨*proof*⟩

**lemma** *vlepoll-vempty-iff[simp]*:  $A \lesssim_o 0 \longleftrightarrow A = 0$  ⟨*proof*⟩

#### 2.6.4 *vlespoll*

**abbreviation** *vlesspoll* ::  $V \Rightarrow V \Rightarrow bool$  (**infixl**  $\langle <_o \rangle$  50)  
 where  $A <_o B \equiv elts\ A < elts\ B$

**lemma** *vlesspoll-def*:  $A <_o B = (A \lesssim_o B \wedge \sim(A \approx_o B))$  ⟨*proof*⟩

Rules.

**lemmas** *vlesspollI[intro]* = *vlesspoll-def[THEN iffD2]*

**lemmas** *vlesspollD[dest]* = *vlesspoll-def[THEN iffD1]*

**lemma** *vlesspollE[elim]*:  
 assumes  $A <_o B$  and  $A \lesssim_o B \implies \sim(A \approx_o B) \implies P$   
 shows  $P$   
 ⟨*proof*⟩

**lemma** (**in** *v11*) *v11-vlepollI[intro]*:  
 assumes  $\mathcal{D}_o\ r = A$  and  $\mathcal{R}_o\ r \subseteq_o B$   
 shows  $A \lesssim_o B$   
 ⟨*proof*⟩

**lemmas** *v11-vlepollI[intro]* = *v11.v11-vlepollI*

**lemma** *v11-vlepollE[elim]*:  
 assumes  $A \lesssim_o B$   
 obtains  $f$  where  $v11\ f$  and  $\mathcal{D}_o\ f = A$  and  $\mathcal{R}_o\ f \subseteq_o B$   
 ⟨*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 iffD1*]  
 and *vcard-vemptyI* = *vcard-vempty*[*THEN iffD2*]

**lemma** *vcard-neq-vempty*:  $vcard\ A \neq 0_{\mathbb{N}} \longleftrightarrow A \neq 0_{\mathbb{N}}$   
 ⟨*proof*⟩

**lemmas** *vcard-neq-vemptyD* = *vcard-neq-vempty*[*THEN iffD1*]  
 and *vcard-neq-vemptyI* = *vcard-neq-vempty*[*THEN iffD2*]

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*:  $vcard (D_{\circ} r) = vcard r$

*<proof>*

Special properties.

**lemma** *vcard-union-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**  $\exists n \in_{\circ} \omega. n \approx_{\circ} A \implies P$

**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-vleqpoll-trans*:

**assumes** *vfinite*  $A$  and  $B \lesssim_0 A$   
**shows** *vfinite*  $B$   
 ⟨*proof*⟩

**lemma** *vfinite-vlesspoll-trans*:  
**assumes** *vfinite*  $A$  and  $B <_0 A$   
**shows** *vfinite*  $B$   
 ⟨*proof*⟩

Induction.

**lemma** *vfinite-induct*[*consumes* 1, *case-names* *vempty* *vinfosert*]:  
**assumes** *vfinite*  $F$   
 and  $P\ 0$   
 and  $\wedge x\ F. \llbracket \text{vfinite } F; x \notin_0 F; P\ F \rrbracket \implies P\ (\text{vinfosert } x\ F)$   
**shows**  $P\ F$   
 ⟨*proof*⟩

Set operations.

**lemma** *vfinite-vempty*[*simp*]: *vfinite*  $(0_{\mathbb{N}})$  ⟨*proof*⟩

**lemma** *vfinite-vsingleton*[*simp*]: *vfinite*  $(\text{set } \{x\})$  ⟨*proof*⟩

**lemma** *vfinite-vdoubleton*[*simp*]: *vfinite*  $(\text{set } \{x, y\})$  ⟨*proof*⟩

**lemma** *vfinite-vinsert*:  
**assumes** *vfinite*  $F$   
**shows** *vfinite*  $(\text{vinfosert } x\ F)$   
 ⟨*proof*⟩

**lemma** *vfinite-vinsertD*:  
**assumes** *vfinite*  $(\text{vinfosert } x\ F)$   
**shows** *vfinite*  $F$   
 ⟨*proof*⟩

**lemma** *vfinite-vsubset*:  
**assumes** *vfinite*  $B$  and  $A \subseteq_0 B$   
**shows** *vfinite*  $A$   
 ⟨*proof*⟩

**lemma** *vfinite-vunion*: *vfinite*  $(A \cup_0 B) \iff \text{vfinite } A \wedge \text{vfinite } B$   
 ⟨*proof*⟩

**lemma** *vfinite-vunionI*:  
**assumes** *vfinite*  $A$  and *vfinite*  $B$   
**shows** *vfinite*  $(A \cup_0 B)$   
 ⟨*proof*⟩

**lemma** *vfinite-vunionD*:  
**assumes** *vfinite*  $(A \cup_0 B)$   
**shows** *vfinite*  $A$  and *vfinite*  $B$   
 ⟨*proof*⟩

**lemma** *vfinite-vintersectionI*:  
**assumes** *vfinite*  $A$  and *vfinite*  $B$   
**shows** *vfinite*  $(A \cap_0 B)$   
 ⟨*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*:  $vfinite\ A \leftrightarrow vcard\ A \in_o \omega$   
 ⟨*proof*⟩

**lemmas** *vcard-vfinite-omega* = *vfinite-vcard-omega-iff*[*THEN iffD2*]  
**and** *vfinite-vcard-omega* = *vfinite-vcard-omega-iff*[*THEN iffD1*]

**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*]:  $vcard\ (set\ \{a\}) = 1_N$  ⟨*proof*⟩

**lemma** *vfinite-vcard-vinsert-nin*[*simp*]:  
**assumes** *vfinite*  $A$  **and**  $a \notin_o A$   
**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*  $\beta$  and  $\alpha \in_o \beta$

shows *succ*  $\alpha \in_o$  *succ*  $\beta$

*<proof>*

lemma *Limit-right-Limit-mult*:

— Based on Theorem 8.23 in [59].

assumes *Ord*  $\alpha$  and *Limit*  $\beta$  and  $0 \in_o \alpha$

shows *Limit*  $(\alpha * \beta)$

*<proof>*

lemma *Limit-left-Limit-mult*:

assumes *Limit*  $\alpha$  and *Ord*  $\beta$  and  $0 \in_o \beta$

shows *Limit*  $(\alpha * \beta)$

*<proof>*

lemma *zero-if-Limit-eq-Limit-plus-vnat*:

assumes *Limit*  $\alpha$  and *Limit*  $\beta$  and  $\alpha = \beta + n$  and  $n \in_o \omega$

shows  $n = 0$

*<proof>*

lemma *Ord-vsubset-closed*:

assumes *Ord*  $\alpha$  and *Ord*  $\gamma$  and  $\alpha \subseteq_o \beta$  and  $\beta \in_o \gamma$

shows  $\alpha \in_o \gamma$

*<proof>*

lemma

— Based on Exercise 1, page 53 in [59].

assumes *Ord*  $\alpha$  and *Ord*  $\gamma$  and  $\alpha + \beta \in_o \gamma$

shows *Ord-plus-Ord-closed-augend*:  $\alpha \in_o \gamma$

and *Ord-plus-Ord-closed-addend*:  $\beta \in_o \gamma$

*<proof>*

lemma *Ord-ex1-Limit-plus-in-omega*:

— Based on Theorem 8.13 in [59].

assumes *Ord*  $\alpha$  and  $\omega \subseteq_o \alpha$

shows  $\exists! \beta. \exists! n. n \in_o \omega \wedge$  *Limit*  $\beta \wedge \alpha = \beta + n$

*<proof>*

lemma *not-Limit-if-in-Limit-plus-omega*:

assumes *Limit*  $\alpha$  and  $\alpha \in_o \beta$  and  $\beta \in_o \alpha + \omega$

shows  $\sim$  *Limit*  $\beta$

*<proof>*

lemma *Limit-plus-omega-vsubset-Limit*:

assumes *Limit*  $\alpha$  and *Limit*  $\beta$  and  $\alpha \in_o \beta$

shows  $\alpha + \omega \subseteq_o \beta$

*<proof>*

**lemma** *Limit-plus-nat-in-Limit:*

**assumes** *Limit*  $\alpha$  **and** *Limit*  $\beta$  **and**  $\alpha \in_o \beta$

**shows**  $\alpha + a_{\mathbb{N}} \in_o \beta$

*<proof>*

**lemma** *omega2-vsubset-Limit:*

**assumes** *Limit*  $\alpha$  **and**  $\omega \in_o \alpha$

**shows**  $\omega + \omega \subseteq_o \alpha$

*<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*:  $\mathcal{D}_o \text{ } xs \in_o \omega$

**locale** *vfsequence-pair* =  $r_1$ : *vfsequence xs*<sub>1</sub> +  $r_2$ : *vfsequence xs*<sub>2</sub> **for** *xs*<sub>1</sub> *xs*<sub>2</sub>

Rules.

**lemmas** [*intro*] = *vfsequence.axioms*(1)

**lemma** *vfsequenceI*[*intro*]:  
**assumes** *vsv xs* **and**  $\mathcal{D}_o \text{ } xs \in_o \omega$   
**shows** *vfsequence xs*  
 ⟨*proof*⟩

**lemma** *vfsequenceD*[*dest*]:  
**assumes** *vfsequence xs*  
**shows**  $\mathcal{D}_o \text{ } xs \in_o \omega$   
 ⟨*proof*⟩

**lemma** *vfsequenceE*[*elim*]:  
**assumes** *vfsequence xs* **and**  $\mathcal{D}_o \text{ } xs \in_o \omega \implies P$   
**shows**  $P$   
 ⟨*proof*⟩

**lemma** *vfsequence-iff*: *vfsequence xs*  $\longleftrightarrow vsv \text{ } xs \wedge \mathcal{D}_o \text{ } xs \in_o \omega$   
 ⟨*proof*⟩

Elementary properties.

**lemma** (**in** *vfsequence*) *vfsequence-vdomain*:  $\mathcal{D}_o \text{ } xs = vcard \text{ } xs$   
 ⟨*proof*⟩

**lemma** (**in** *vfsequence*) *vfsequence-vcard-in-omega*[*simp*]:  $vcard \text{ } xs \in_o \omega$   
 ⟨*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-vrestriction[intro, simp]*:  
**assumes**  $k \in_\circ \omega$   
**shows** *vfsequence* ( $xs \upharpoonright_\circ^l k$ )  
 ⟨*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**  $xs \in_\circ A \hat{\ }_x n$   
**shows** *vfsequence xs*  
 ⟨*proof*⟩

**lemma** *vfsequence-vcomp-vs-vfsequence*:  
**assumes** *vs v 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-vrestriction[intro, simp]*:  
**assumes**  $k \in_\circ \text{vcard } xs$   
**shows**  $\mathcal{D}_\circ (xs \upharpoonright_\circ^l k) = k$   
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-vrestriction-vcard[simp]*:  
 $xs \upharpoonright_\circ^l (\text{vcard } xs) = xs$   
 ⟨*proof*⟩

**lemma** *vfsequence-vfinite-vcardI*:  
**assumes** *vs v xs* **and** *vfinite xs* **and**  $\mathcal{D}_\circ xs = \text{vcard } xs$   
**shows** *vfsequence xs*  
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-vrangeE*:  
**assumes**  $a \in_\circ \mathcal{R}_\circ xs$   
**obtains**  $n$  **where**  $n \in_\circ \text{vcard } xs$  **and**  $xs \upharpoonright_\circ^l n = a$   
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-vrange-vproduct*:  
**assumes**  $\bigwedge i. i \in_\circ \text{vcard } xs \implies xs \upharpoonright_\circ^l i \in_\circ A i$   
**shows**  $xs \in_\circ (\prod_{\circ} i \in_\circ \text{vcard } xs. A i)$   
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-vrange-vcpower*:  
**assumes**  $\mathcal{R}_\circ xs \subseteq_\circ A$   
**shows**  $xs \in_\circ A \hat{\ }_x (\text{vcard } xs)$   
 ⟨*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-vrestriction*  
**and** [*intro, simp*] = *vfsequence.vfsequence-vdomain-vrestriction*

and  $[intro, simp] = vfsequence.vfsequence-vrestriction-vcard$

### 2.9.3 Appending an element to a finite sequence: $vcons$

#### Definition and common properties

**definition**  $vcons :: V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \#_o \rangle$  65)  
**where**  $xs \#_o x = vinsert \langle vcard\ xs, x \rangle\ xs$

Syntax.

**abbreviation**  $vempty-vfsequence \langle [ ]_o \rangle$  **where**  
 $vempty-vfsequence \equiv 0 :: V$

**notation**  $vempty-vfsequence \langle [ ]_o \rangle$

**nonterminal**  $fsfields$

**nonterminal**  $vlist$

**syntax**

$:: V \Rightarrow fsfields \langle \langle - \rangle \rangle$   
 $-fsfields :: fsfields \Rightarrow V \Rightarrow fsfields \langle \langle -, / - \rangle \rangle$   
 $-vlist :: fsfields \Rightarrow V \langle \langle [ ]_o \rangle \rangle$   
 $-vapp :: V \Rightarrow fsfields \Rightarrow V \langle \langle - \langle [ ]_o \rangle \bullet \rangle [100, 100] 100 \rangle$

**syntax-consts**

$-vlist == vcons$  **and**  
 $-vapp == app$

**translations**

$[xs, x]_o == [xs]_o \#_o x$   
 $[x]_o == []_o \#_o x$

**translations**

$f \langle [xs, x]_o \bullet \rangle == f \langle [ [xs, x]_o \rangle \rangle$   
 $f \langle [x]_o \bullet \rangle == f \langle [ [x]_o \rangle \rangle$

Rules.

**lemma**  $vconsI[intro!]$ :

**assumes**  $a \in_o vinsert \langle vcard\ xs, x \rangle\ xs$   
**shows**  $a \in_o xs \#_o x$   
 $\langle proof \rangle$

**lemma**  $vconsD[dest!]$ :

**assumes**  $a \in_o xs \#_o x$   
**shows**  $a \in_o vinsert \langle vcard\ xs, x \rangle\ xs$   
 $\langle proof \rangle$

**lemma**  $vconsE[elim!]$ :

**assumes**  $a \in_o xs \#_o x$   
**obtains**  $a$  **where**  $a \in_o vinsert \langle vcard\ xs, x \rangle\ xs$   
 $\langle proof \rangle$

Elementary properties.

**lemma**  $vcons-neq-vempty[simp]$ :  $ys \#_o y \neq []_o$   $\langle proof \rangle$

Set operations.

**lemma**  $vcons-vsingleton$ :  $[a]_o = set \{ \langle 0_N, a \rangle \}$   $\langle proof \rangle$

**lemma** *vcons-vdoubleton*:  $[a, b]_{\circ} = \text{set } \{\langle 0_{\mathbf{N}}, a \rangle, \langle 1_{\mathbf{N}}, b \rangle\}$   
 ⟨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*]:  
 $\mathcal{D}_{\circ} (xs \#_{\circ} x) = \text{succ } (vcard \ xs)$   
 ⟨proof⟩

**lemma** (in *vfsequence*) *vfsequence-vcons-vrange*[*simp*]:  
 $\mathcal{R}_{\circ} (xs \#_{\circ} x) = \text{vinsert } x \ (\mathcal{R}_{\circ} \ xs)$   
 ⟨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-vrestriction-succ*:  
**assumes** [*simp*]:  $k \in_{\circ} vcard \ xs$   
**shows**  $xs \uparrow^l_{\circ} \text{succ } k = xs \uparrow^l_{\circ} k \#_{\circ} (xs(|k|))$   
 ⟨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**  $\bigwedge xs \ x. [[i\text{vfsequence } xs; P \ xs]] \implies P \ (xs \#_{\circ} x)$   
**shows**  $P \ xs$   
 ⟨proof⟩

**lemma** *vfsequence-cases*[*consumes* 1, *case-names* 0 *vcons*]:  
**assumes** *vfsequence xs*  
**and**  $xs = []_{\circ} \implies P$   
**and**  $\bigwedge xs' x. [[xs = xs' \#_{\circ} x; vfsequence xs']] \implies P$   
**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**  $vfsequence-of-vlist :: V\ list \Rightarrow V$

**where**

$vfsequence-of-vlist\ [] = 0$

$| vfsequence-of-vlist\ (x\ \# \ xs) = vfsequence-of-vlist\ xs\ \#_o\ x$

**definition**  $vlist-of-vfsequence :: V \Rightarrow V\ list$

**where**  $vlist-of-vfsequence = inv-into\ UNIV\ vfsequence-of-vlist$

**lemma**  $vfsequence-vfsequence-of-vlist: vfsequence\ (vfsequence-of-vlist\ xs)$

$\langle proof \rangle$

**lemma**  $inj-vfsequence-of-vlist: inj\ vfsequence-of-vlist$

$\langle proof \rangle$

**lemma**  $range-vfsequence-of-vlist:$

$range\ vfsequence-of-vlist = \{xs.\ vfsequence\ xs\}$

$\langle proof \rangle$

**lemma**  $vlist-of-vfsequence-vfsequence-of-vlist[simp]:$

$vlist-of-vfsequence\ (vfsequence-of-vlist\ xs) = xs$

$\langle proof \rangle$

**lemma**  $(in\ vfsequence)\ vfsequence-of-vlist-vlist-of-vfsequence[simp]:$

$vfsequence-of-vlist\ (vlist-of-vfsequence\ xs) = xs$

$\langle proof \rangle$

**lemmas**  $vfsequence-of-vlist-vlist-of-vfsequence[intro, simp] =$

$vfsequence.vfsequence-of-vlist-vlist-of-vfsequence$

**lemma**  $vlist-of-vfsequence-vempty[simp]: vlist-of-vfsequence\ []_o = []$

$\langle proof \rangle$

Transfer relation 1.

**definition**  $cr-vfsequence :: V \Rightarrow V\ list \Rightarrow bool$

**where**  $cr-vfsequence\ a\ b \longleftrightarrow (a = vfsequence-of-vlist\ b)$

**lemma**  $cr-vfsequence-right-total[transfer-rule]: right-total\ cr-vfsequence$

$\langle proof \rangle$

**lemma**  $cr-vfsequence-bi-unqie[transfer-rule]: bi-unique\ cr-vfsequence$

*<proof>*

**lemma** *cr-vfsequence-transfer-domain-rule*[*transfer-domain-rule*]:

*Domainp cr-vfsequence = (λxs. vfsequence xs)*

*<proof>*

**lemma** *cr-vfsequence-vconsD*:

**assumes** *cr-vfsequence (xs #<sub>o</sub> x) (y # ys)*

**shows** *cr-vfsequence xs ys and x = y*

*<proof>*

Transfer relation 2.

**definition** *cr-cr-vfsequence* ::  $V \Rightarrow V \text{ list list} \Rightarrow \text{bool}$

**where** *cr-cr-vfsequence a b*  $\longleftrightarrow$

$(a = \text{vfsequence-of-vlist } (\text{map } \text{vfsequence-of-vlist } b))$

**lemma** *cr-cr-vfsequence-right-total*[*transfer-rule*]:

*right-total cr-cr-vfsequence*

*<proof>*

**lemma** *cr-cr-vfsequence-bi-unique*[*transfer-rule*]: *bi-unique cr-cr-vfsequence*

*<proof>*

Transfer relation for scalars.

**definition** *cr-scalar* ::  $(V \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow V \Rightarrow 'a \Rightarrow \text{bool}$

**where** *cr-scalar R x y* =  $(\exists a. x = [a]_o \wedge R a y)$

**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) = (λx. ∃ a. x = [a]\_o ∧ Domainp R a)*

*<proof>*

## Transfer rules for previously defined entities

**context**

**includes** *lifting-syntax*

**begin**

**lemma** *vfsequence-vempty-transfer*[*transfer-rule*]: *cr-vfsequence []<sub>o</sub> []*

*<proof>*

**lemma** *vfsequence-vempty-ll-transfer*[*transfer-rule*]:

*cr-cr-vfsequence [[]]<sub>o</sub> [[]]*

*<proof>*

**lemma** *vcons-transfer*[*transfer-rule*]:

$((=) \implies \text{cr-vfsequence} \implies \text{cr-vfsequence}) (\lambda x xs. xs \#_o x) (\lambda x xs. x \# xs)$

*<proof>*

**lemma** *vcons-ll-transfer*[*transfer-rule*]:  
 (*cr-vfsequence* ==> *cr-cr-vfsequence* ==> *cr-cr-vfsequence*)  
 ( $\lambda x xs. xs \#_o x$ ) ( $\lambda x xs. x \# xs$ )  
 <proof>

**lemma** *vfsequence-vrange-transfer*[*transfer-rule*]:  
 (*cr-vfsequence* ==> (=)) ( $\lambda xs. elts (\mathcal{R}_o xs)$ ) *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*:  $\mathcal{D}_o$  (*vfsequence-of-vlist* *xs*) = *length xs*  
 <proof>

**lemma** *vrange-vfsequence-of-vlist*:  
 $\mathcal{R}_o$  (*vfsequence-of-vlist* *xs*) = *set (list.set xs)*  
 <proof>

**lemma** *cr-cr-vfsequence-transfer-domain-rule*[*transfer-domain-rule*]:  
*Domainp cr-cr-vfsequence* =  
 ( $\lambda xss. vfsequence xss \wedge (\forall xs \in_o \mathcal{R}_o xss. vfsequence xs)$ )  
 <proof>

## Appending elements

**definition** *vappend* ::  $V \Rightarrow V \Rightarrow V$  (**infixr** <@<sub>o</sub>> 65)  
**where**  $xs @_o ys =$   
*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*)  
 ( $\lambda xs ys. vappend ys xs$ ) *append*  
 <proof>

**lemma** *vappend-ll-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*cr-cr-vfsequence* ==> *cr-cr-vfsequence* ==> *cr-cr-vfsequence*)  
 ( $\lambda xs ys. vappend ys xs$ ) *append*  
 <proof>

Elementary properties.

**lemma** (**in** *vfsequence*) *vfsequence-vappend-vempty-vfsequence*[*simp*]:  
 $[\ ]_o @_o xs = xs$   
 <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* ::  $V \Rightarrow bool$   
**where** *vdistinct xs* = *distinct (vlist-of-vfsequence xs)*

Transfer.

**lemma** *vdistinct-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows**  $(cr\text{-}vfsequence \implies (=)) \text{ } vdistinct \text{ } distinct$   
 ⟨*proof*⟩

**lemma** *vdistinct-ll-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows**  $(cr\text{-}cr\text{-}vfsequence \implies (=)) \text{ } vdistinct \text{ } distinct$   
 ⟨*proof*⟩

Elementary properties.

**lemma** (in *vfsequence*) *vfsequence-vdistinct-if-vcard-vrange-eq-vcard*:  
**assumes**  $vcard (\mathcal{R}_{\circ} xs) = vcard \text{ } xs$   
**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**  $x \notin_{\circ} \mathcal{R}_{\circ} xs$   
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-v11I*[*intro*]:  
**assumes** *vdistinct xs*  
**shows** *v11 xs*  
 ⟨*proof*⟩

**lemma** (in *vfsequence*) *vfsequence-vcons-vdistinctI*:  
**assumes** *vdistinct xs* **and**  $x \notin_{\circ} \mathcal{R}_{\circ} xs$   
**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* []<sub>o</sub> = []<sub>o</sub>  
*<proof>*

**lemma** *vconcat-append*[*simp*]:  
**assumes** *vfsequence* *xss*  
**and**  $\forall xs \in_{\circ} \mathcal{R}_{\circ} xss. \text{vfsequence } xs$   
**and** *vfsequence* *yss*  
**and**  $\forall xs \in_{\circ} \mathcal{R}_{\circ} yss. \text{vfsequence } xs$   
**shows** *vconcat* (*xss* @<sub>o</sub> *yss*) = *vconcat* *xss* @<sub>o</sub> *vconcat* *yss*  
*<proof>*

**lemma** *vconcat-vcons*[*simp*]:  
**assumes** *vfsequence* *xs* **and** *vfsequence* *xss* **and**  $\forall xs \in_{\circ} \mathcal{R}_{\circ} xss. \text{vfsequence } xs$   
**shows** *vconcat* (*xss* #<sub>o</sub> *xs*) = *vconcat* *xss* @<sub>o</sub> *xs*  
*<proof>*

**lemma** (**in** *vfsequence*) *vfsequence-vconcat-fsingleton*[*simp*]: *vconcat* [*xs*]<sub>o</sub> = *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**  $xs \in_{\circ} (\prod_{\circ} i \in_{\circ} \text{vcard } xs. A \ i)$   
**and**  $x \in_{\circ} A \ (\text{vcard } xs)$   
**and**  $n = \text{vcard } (xs \#_{\circ} x)$   
**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**  $xs \in_{\circ} (\prod_{\circ} i \in_{\circ} \text{vcard } xs. A \ i)$   
**and**  $x \in_{\circ} A \ (\text{vcard } xs)$   
**and**  $n = \text{vcard } (xs \#_{\circ} x)$   
*<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**  $xs \in_{\circ} (\prod_{\circ} i \in_{\circ} \text{vcard } xs. A \ i)$   
**and**  $x \in_{\circ} A \ (\text{vcard } xs)$   
**and**  $n = \text{vcard } (xs \#_{\circ} x)$   
*<proof>*

## 2.9.6 Binary Cartesian product based on finite sequences: *ftimes*

**definition** *ftimes* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \times_{\bullet} \rangle$  80)  
**where** *ftimes*  $a \ b \equiv (\prod_{\circ} i \in_{\circ} 2_{\mathbb{N}}. \text{if } i = 0 \text{ then } a \ \text{else } b)$

**lemma** *small-fpairs[simp]*: *small*  $\{[a, b]_{\circ} \mid a \ b. [a, b]_{\circ} \in_{\circ} r\}$   
*<proof>*

Rules.

**lemma** *ftimesI1[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** *ftimesI2[intro!]*:  
**assumes**  $a \in_{\circ} A$  **and**  $b \in_{\circ} B$   
**shows**  $[a, b]_{\circ} \in_{\circ} A \times_{\bullet} B$   
*<proof>*

**lemma** *fproductE1[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** *fproductE2[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*:  $[a, b]_o \in_o (\prod_{i \in_o 2_N} f i) \longleftrightarrow \langle a, b \rangle \in_o f (0_N) \times_o f (1_N)$   
 ⟨proof⟩

Connections.

**lemma** *vcpower-two-ftimes*:  $A \hat{\times}_x 2_N = A \times_\bullet A$   
 ⟨proof⟩

**lemma** *vcpower-two-ftimesI[intro]*:  
**assumes**  $x \in_o A \times_\bullet A$   
**shows**  $x \in_o A \hat{\times}_x 2_N$   
 ⟨proof⟩

**lemma** *vcpower-two-ftimesD[dest]*:  
**assumes**  $x \in_o A \hat{\times}_x 2_N$   
**shows**  $x \in_o A \times_\bullet A$   
 ⟨proof⟩

**lemma** *vcpower-two-ftimesE[elim]*:  
**assumes**  $x \in_o A \hat{\times}_x 2_N$  **and**  $x \in_o A \times_\bullet A \implies P$   
**shows**  $P$   
 ⟨proof⟩

**lemma** *vfsequence-vcpower-two-vpair*:  $[a, b]_o \in_o A \hat{\times}_x 2_N \longleftrightarrow \langle a, b \rangle \in_o A \times_o A$   
 ⟨proof⟩

**lemma** *vsv-vfsequence-two*:  
**assumes** *vsv gf* **and**  $\mathcal{D}_o gf = 2_N$   
**shows**  $[vpfst gf, vpsnd gf]_o = gf$   
 ⟨proof⟩

**lemma** *vsv-vfsequence-three*:  
**assumes** *vsv hgf* **and**  $\mathcal{D}_o hgf = 3_N$   
**shows**  $[vpfst hgf, vpsnd hgf, vpthrd hgf]_o = hgf$   
 ⟨proof⟩

### 2.9.7 Sequence as an element of a Cartesian power of a set

**lemma** *vcons-in-vcpowerI[intro!]*:  
**assumes**  $n \in_o \omega$   
**and**  $xs \in_o A \hat{\times}_x vcard xs$   
**and**  $x \in_o A$   
**and**  $n = vcard (xs \#_o x)$   
**shows**  $xs \#_o x \in_o A \hat{\times}_x n$   
 ⟨proof⟩

**lemma** *vcons-in-vcpowerD[dest]*:  
**assumes**  $xs \#_o x \in_o A \hat{\times}_x n$  **and**  $n \in_o \omega$   
**shows**  $xs \in_o A \hat{\times}_x vcard xs$   
**and**  $x \in_o A$   
**and**  $n = vcard (xs \#_o x)$   
 ⟨proof⟩

**lemma** *vcons-in-vcpowerE1[elim!]*:  
**assumes**  $xs \#_o x \in_o A \hat{\times}_x n$  **and**  $n \in_o \omega$   
**obtains**  $xs \in_o A \hat{\times}_x vcard xs$  **and**  $x \in_o A$  **and**  $n = vcard (xs \#_o x)$   
 ⟨proof⟩

**lemma** *vcons-in-vcpowerE2*:

**assumes**  $xs \in_o A \widehat{\times} n$  **and**  $n \in_o \omega$  **and**  $0 \in_o n$   
**obtains**  $x \ xs'$  **where**  $xs = xs' \#_o x$   
**and**  $xs' \in_o A \widehat{\times} \text{vcard } xs'$   
**and**  $x \in_o A$   
**and**  $n = \text{vcard } (xs' \#_o x)$

*<proof>*

**lemma** *vcons-vcpower1E*:

**assumes**  $xs \in_o A \widehat{\times} 1_{\mathbb{N}}$   
**obtains**  $x$  **where**  $xs = [x]_o$  **and**  $x \in_o A$

*<proof>*

## 2.9.8 The set of all finite sequences on a set

### Definition and elementary properties

**definition** *vfsequences-on* ::  $V \Rightarrow V$

**where**  $\text{vfsequences-on } X = \text{set } \{x. \text{vfsequence } x \wedge (\forall i \in_o \mathcal{D}_o x. x(i) \in_o X)\}$

**lemma** *vfsequences-on-subset- $\omega$ -set*:

$\{x. \text{vfsequence } x \wedge (\forall i \in \text{elts } (\mathcal{D}_o x). x(i) \in_o X)\} \subseteq \text{elts } (\text{VPow } (\omega \times_o X))$

*<proof>*

**lemma** *small-vfsequences-on[simp]*:

*small*  $\{x. \text{vfsequence } x \wedge (\forall i \in_o \mathcal{D}_o x. x(i) \in_o X)\}$

*<proof>*

Rules.

**lemma** *vfsequences-onI*:

**assumes** *vfsequence*  $xs$  **and**  $\wedge i. i \in_o \mathcal{D}_o xs \implies xs(i) \in_o X$

**shows**  $xs \in_o \text{vfsequences-on } X$

*<proof>*

**lemma** *vfsequences-onD[dest]*:

**assumes**  $xs \in_o \text{vfsequences-on } X$

**shows** *vfsequence*  $xs$  **and**  $\wedge i. i \in_o \mathcal{D}_o xs \implies xs(i) \in_o X$

*<proof>*

**lemma** *vfsequences-onE[elim]*:

**assumes**  $xs \in_o \text{vfsequences-on } X$

**obtains** *vfsequence*  $xs$  **and**  $\wedge i. i \in_o \mathcal{D}_o xs \implies xs(i) \in_o X$

*<proof>*

### Further properties

**lemma** *vfsequences-on-vsubset-mono*:

**assumes**  $A \subseteq_o B$

**shows**  $\text{vfsequences-on } A \subseteq_o \text{vfsequences-on } B$

*<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  $[a, b]_o$  (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]*:  $([a, b]_o = [a', b']_o) = (a = a' \wedge b = b')$  *(proof)*

**lemmas** *fpair-inject[elim!]* = *fpair-iff[THEN iffD1, THEN conjE]*

### 2.10.2 *fpairs*

**definition** *fpairs* ::  $V \Rightarrow V$  **where**

*fpairs*  $r = \text{set } \{x. x \in_o r \wedge (\exists a b. x = [a, b]_o)\}$

**lemma** *small-fpairs[simp]*: *small*  $\{x. x \in_o r \wedge (\exists a b. x = [a, b]_o)\}$   
*(proof)*

Rules.

**lemma** *fpairsI[intro]*:

**assumes**  $x \in_o r$  **and**  $x = [a, b]_o$

**shows**  $x \in_o \text{fpairs } r$

*(proof)*

**lemma** *fpairsD[dest]*:

**assumes**  $x \in_o \text{fpairs } r$

**shows**  $x \in_o r$  **and**  $\exists a b. x = [a, b]_o$

*(proof)*

**lemma** *fpairsE[elim]*:

**assumes**  $x \in_o \text{fpairs } r$

**obtains**  $a b$  **where**  $x = [a, b]_o$  **and**  $[a, b]_o \in_o r$

*(proof)*

**lemma** *fpairs-iff*:  $x \in_o \text{fpairs } r \longleftrightarrow x \in_o r \wedge (\exists a b. x = [a, b]_o)$  *(proof)*

Elementary properties.

**lemma** *fpairs-iff-elts*:  $[a, b]_o \in_o \text{fpairs } r \longleftrightarrow [a, b]_o \in_o r$  *(proof)*

Set operations.

**lemma** *fpairs-verify[simp]*: *fpairs*  $0 = 0$  *(proof)*

**lemma** *fpairs-vsingleton[simp]*: *fpairs*  $(\text{set } \{[a, b]_o\}) = \text{set } \{[a, b]_o\}$  *(proof)*

**lemma** *fpairs-vinsert*: *fpairs*  $(\text{vinsert } [a, b]_o A) = \text{set } \{[a, b]_o\} \cup_o \text{fpairs } A$   
*(proof)*

**lemma** *fpairs-mono*:

**assumes**  $r \subseteq_o s$

**shows** *fpairs*  $r \subseteq_o \text{fpairs } s$

*(proof)*

**lemma** *fpairs-vunion*: *fpairs*  $(A \cup_o B) = \text{fpairs } A \cup_o \text{fpairs } B$  *(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**  $\exists b. [x(0_{\mathbf{N}}), b]_{\circ} \in_{\circ} r$

*<proof>*

**lemma** *fpairs-ex-vsnd*:

**assumes**  $x \in_{\circ} fpairs r$

**shows**  $\exists a. [a, x(1_{\mathbf{N}})]_{\circ} \in_{\circ} r$

*<proof>*

**lemma** *fpair-vcpower2I[intro]*:

**assumes**  $a \in_{\circ} A \hat{\times}_{\times} 1_{\mathbf{N}}$  **and**  $b \in_{\circ} A \hat{\times}_{\times} 1_{\mathbf{N}}$

**shows**  $vconcat [a, b]_{\circ} \in_{\circ} A \hat{\times}_{\times} 2_{\mathbf{N}}$

*<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**  $\exists a \in_{\circ} A. x = [a, a]_{\circ} \implies P$

**shows**  $P$

*<proof>*

**lemma** *fid-on-iff*:  $[a, b]_{\circ} \in_{\circ} fid-on A \iff a = b \wedge a \in_{\circ} A$  *<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*:  $\text{fid-on } (\text{set } \{a, b\}) = \text{set } \{[a, a]_{\circ}, [b, b]_{\circ}\}$  *<proof>*

**lemma** *fid-on-mono*:

**assumes**  $A \subseteq_{\circ} B$

**shows**  $\text{fid-on } A \subseteq_{\circ} \text{fid-on } B$

*<proof>*

**lemma** *fid-on-vinsert*:  $\text{vinsert } [a, a]_{\circ} (\text{fid-on } A) = \text{fid-on } (\text{vinsert } a A)$

*<proof>*

**lemma** *fid-on-vintersection*:  $\text{fid-on } (A \cap_{\circ} B) = \text{fid-on } A \cap_{\circ} \text{fid-on } B$  *<proof>*

**lemma** *fid-on-vunion*:  $\text{fid-on } (A \cup_{\circ} B) = \text{fid-on } A \cup_{\circ} \text{fid-on } B$  *<proof>*

**lemma** *fid-on-vdiff*:  $\text{fid-on } (A -_{\circ} B) = \text{fid-on } A -_{\circ} \text{fid-on } B$  *<proof>*

Special properties.

**lemma** *fid-on-vsubset-vcpower*:  $\text{fid-on } A \subseteq_{\circ} A \hat{\times}_{\times} 2_{\mathbb{N}}$  *<proof>*

### Constant function

**definition** *fconst-on* ::  $V \Rightarrow V \Rightarrow V$

**where**  $\text{fconst-on } A c = \text{set } \{[a, c]_{\circ} \mid a. a \in_{\circ} A\}$

**lemma** *small-fconst-on[simp]*:  $\text{small } \{[a, c]_{\circ} \mid a. a \in_{\circ} A\}$

*<proof>*

Rules.

**lemma** *fconst-onI[intro!]*:

**assumes**  $a \in_{\circ} A$

**shows**  $[a, c]_{\circ} \in_{\circ} \text{fconst-on } A c$

*<proof>*

**lemma** *fconst-onD[dest!]*:

**assumes**  $[a, c]_{\circ} \in_{\circ} \text{fconst-on } A c$

**shows**  $a \in_{\circ} A$

*<proof>*

**lemma** *fconst-onE[elim!]*:

**assumes**  $x \in_{\circ} \text{fconst-on } A c$

**obtains**  $a$  **where**  $a \in_{\circ} A$  **and**  $x = [a, c]_{\circ}$

*<proof>*

**lemma** *fconst-on-iff*:  $[a, c]_{\circ} \in_{\circ} \text{fconst-on } A c \longleftrightarrow a \in_{\circ} A$  *<proof>*

Set operations.

**lemma** *fconst-on-vempty[simp]*:  $\text{fconst-on } 0 c = 0$

*<proof>*

**lemma** *fconst-on-vsingleton[simp]*:  $\text{fconst-on } (\text{set } \{a\}) c = \text{set } \{[a, c]_{\circ}\}$

*<proof>*

**lemma** *fconst-on-vdoubleton*:  $\text{fconst-on } (\text{set } \{a, b\}) c = \text{set } \{[a, c]_{\circ}, [b, c]_{\circ}\}$

*<proof>*

**lemma** *fconst-on-mono*:

**assumes**  $A \subseteq_{\circ} B$

**shows**  $fconst\text{-on } A \subseteq_0 fconst\text{-on } B \ c$   
 ⟨proof⟩

**lemma**  $fconst\text{-on-vinsert}$ :  
 $(vinsert [a, c]_0 (fconst\text{-on } A \ c)) = (fconst\text{-on } (vinsert \ a \ A) \ c)$   
 ⟨proof⟩

**lemma**  $fconst\text{-on-vintersection}$ :  
 $fconst\text{-on } (A \cap_0 B) \ c = fconst\text{-on } A \ c \cap_0 fconst\text{-on } B \ c$   
 ⟨proof⟩

**lemma**  $fconst\text{-on-vunion}$ :  $fconst\text{-on } (A \cup_0 B) \ c = fconst\text{-on } A \ c \cup_0 fconst\text{-on } B \ c$   
 ⟨proof⟩

**lemma**  $fconst\text{-on-vdiff}$ :  $fconst\text{-on } (A \ -_0 B) \ c = fconst\text{-on } A \ c \ -_0 fconst\text{-on } B \ c$   
 ⟨proof⟩

Special properties.

**lemma**  $fconst\text{-on-eq-ftimes}$ :  $fconst\text{-on } A \ c = A \times_{\bullet} set \ \{c\}$  ⟨proof⟩

## Composition

**definition**  $fcomp :: V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \circ_{\bullet} \rangle$  75)  
**where**  $r \circ_{\bullet} s = set \ \{[a, c]_0 \mid a \ c. \exists b. [a, b]_0 \in_0 s \wedge [b, c]_0 \in_0 r\}$   
**notation**  $fcomp$  (**infixr**  $\langle \circ_{\bullet} \rangle$  75)

**lemma**  $fcomp\text{-small}[simp]$ :  $small \ \{[a, c]_0 \mid a \ c. \exists b. [a, b]_0 \in_0 s \wedge [b, c]_0 \in_0 r\}$   
 (**is**  $\langle small \ ?s \rangle$ )  
 ⟨proof⟩

Rules.

**lemma**  $fcompI[intro]$ :  
**assumes**  $[b, c]_0 \in_0 r$  **and**  $[a, b]_0 \in_0 s$   
**shows**  $[a, c]_0 \in_0 r \circ_{\bullet} s$   
 ⟨proof⟩

**lemma**  $fcompD[dest]$ :  
**assumes**  $[a, c]_0 \in_0 r \circ_{\bullet} s$   
**shows**  $\exists b. [b, c]_0 \in_0 r \wedge [a, b]_0 \in_0 s$   
 ⟨proof⟩

**lemma**  $fcompE[elim]$ :  
**assumes**  $ac \in_0 r \circ_{\bullet} s$   
**obtains**  $a \ b \ c$  **where**  $ac = [a, c]_0$  **and**  $[a, b]_0 \in_0 s$  **and**  $[b, c]_0 \in_0 r$   
 ⟨proof⟩

Elementary properties.

**lemma**  $fcomp\text{-assoc}$ :  $(r \circ_{\bullet} s) \circ_{\bullet} t = r \circ_{\bullet} (s \circ_{\bullet} t)$  ⟨proof⟩

Set operations.

**lemma**  $fcomp\text{-vempty-left}[simp]$ :  $0 \circ_{\bullet} r = 0$  ⟨proof⟩

**lemma**  $fcomp\text{-vempty-right}[simp]$ :  $r \circ_{\bullet} 0 = 0$  ⟨proof⟩

**lemma**  $fcomp\text{-mono}$ :  
**assumes**  $r' \subseteq_0 r$  **and**  $s' \subseteq_0 s$   
**shows**  $r' \circ_{\bullet} s' \subseteq_0 r \circ_{\bullet} s$

*<proof>*

**lemma** *fcomp-vinsert-left[simp]*:

*vinsert*  $([a, b]_o) s \circ_\bullet r = (set \{[a, b]_o\} \circ_\bullet r) \cup_o (s \circ_\bullet r)$   
*<proof>*

**lemma** *fcomp-vinsert-right[simp]*:

$r \circ_\bullet vinsert [a, b]_o s = (r \circ_\bullet set \{[a, b]_o\}) \cup_o (r \circ_\bullet s)$   
*<proof>*

**lemma** *fcomp-vunion-left[simp]*:  $(s \cup_o t) \circ_\bullet r = (s \circ_\bullet r) \cup_o (t \circ_\bullet r)$  *<proof>*

**lemma** *fcomp-vunion-right[simp]*:  $r \circ_\bullet (s \cup_o t) = (r \circ_\bullet s) \cup_o (r \circ_\bullet t)$  *<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*  $A \circ_\bullet fid-on B = fid-on (A \cap_o B)$  *<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**  $r \subseteq_o B \times_\bullet C$  **and**  $s \subseteq_o A \times_\bullet B$   
**shows**  $r \circ_\bullet s \subseteq_o A \times_\bullet C$   
*<proof>*

**lemma** *fcomp-obtain-middle[elim]*:

**assumes**  $[a, c]_o \in_o f \circ_\bullet g$   
**obtains**  $b$  **where**  $[a, b]_o \in_o g$  **and**  $[b, c]_o \in_o f$   
*<proof>*

## Converse relation

**definition** *fconverse* ::  $V \Rightarrow V (\langle(-)^{-1}, \rangle)$  [1000] 999

**where**  $r^{-1}_\bullet = set \{[b, a]_o \mid a b. [a, b]_o \in_o r\}$

**lemma** *fconverse-small[simp]*: *small*  $\{[b, a]_o \mid a b. [a, b]_o \in_o r\}$   
*<proof>*

Rules.

**lemma** *fconverseI[sym, intro!]*:

**assumes**  $[a, b]_o \in_o r$   
**shows**  $[b, a]_o \in_o r^{-1}_\bullet$   
*<proof>*

**lemma** *fconverseD[sym, dest]*:

**assumes**  $[a, b]_o \in_o r^{-1}_\bullet$   
**shows**  $[b, a]_o \in_o r$   
*<proof>*

**lemma** *fconverseE[elim!]*:

**assumes**  $x \in_o r^{-1}_\bullet$   
**obtains**  $a b$  **where**  $x = [b, a]_o$  **and**  $[a, b]_o \in_o r$   
*<proof>*

**lemma** *fconverse-iff*:  $[b, a]_o \in_o r^{-1}_\bullet \iff [a, b]_o \in_o r$  *<proof>*

Set operations.

**lemma** *fconverse-vempty[simp]*:  $0^{-1} \bullet = 0$  *<proof>*

**lemma** *fconverse-vsingleton*:  $(\text{set } \{[a, b]_{\circ}\})^{-1} \bullet = \text{set } \{[b, a]_{\circ}\}$  *<proof>*

**lemma** *fconverse-vdoubleton*:  $(\text{set } \{[a, b]_{\circ}, [c, d]_{\circ}\})^{-1} \bullet = \text{set } \{[b, a]_{\circ}, [d, c]_{\circ}\}$  *<proof>*

**lemma** *fconverse-vinsert*:  $(\text{vinsert } [a, b]_{\circ} r)^{-1} \bullet = \text{vinsert } [b, a]_{\circ} (r^{-1} \bullet)$  *<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]*:  $(\text{fid-on } A)^{-1} \bullet = \text{fid-on } A$  *<proof>*

**lemma** *fconverse-fconst-on[simp]*:  $(\text{fconst-on } A c)^{-1} \bullet = \text{set } \{c\} \times_{\bullet} A$  *<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**  $(\text{set } \{[a, b]_{\circ} \mid a b. P a b\})^{-1} \bullet = \text{set } \{[b, a]_{\circ} \mid a b. P a b\}$

*<proof>*

**Left restriction**

**definition** *frestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow^{\bullet} \rangle$  80)

**where**  $r \uparrow^{\bullet} A = \text{set } \{[a, b]_{\circ} \mid a b. a \in_{\circ} A \wedge [a, b]_{\circ} \in_{\circ} r\}$

**lemma** *frestriction-small[simp]*: *small*  $\{[a, b]_{\circ} \mid a b. a \in_{\circ} A \wedge [a, b]_{\circ} \in_{\circ} r\}$  *<proof>*

Rules.

**lemma** *frestrictionI[intro!]*:

**assumes**  $a \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$

**shows**  $[a, b]_{\circ} \in_{\circ} r \uparrow^{\bullet} A$

*<proof>*

**lemma** *frestrictionD[dest]*:

**assumes**  $[a, b]_{\circ} \in_{\circ} r \uparrow^{\bullet} A$

**shows**  $a \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$

*<proof>*

**lemma** *frestrictionE[elim!]*:

**assumes**  $x \in_{\circ} r \uparrow^{\bullet} A$

**obtains**  $a b$  **where**  $x = [a, b]_{\circ}$  **and**  $a \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$

*<proof>*

Set operations.

**lemma** *frestriction-on-vempty[simp]*:  $r \uparrow^{\bullet} 0 = 0$  *<proof>*

**lemma** *frestriction-vempty[simp]*:  $0 \uparrow^{\bullet} A = 0$  *<proof>*

**lemma** *flrestriction-usingleton-in[simp]*:  
**assumes**  $a \in_o A$   
**shows**  $set \{[a, b]_o\} \uparrow^l. A = set \{[a, b]_o\}$   
 $\langle proof \rangle$

**lemma** *flrestriction-usingleton-nin[simp]*:  
**assumes**  $a \notin_o A$   
**shows**  $set \{[a, b]_o\} \uparrow^l. A = 0$   
 $\langle proof \rangle$

**lemma** *flrestriction-mono*:  
**assumes**  $A \subseteq_o B$   
**shows**  $r \uparrow^l. A \subseteq_o r \uparrow^l. B$   
 $\langle proof \rangle$

**lemma** *flrestriction-vinsert-nin[simp]*:  
**assumes**  $a \notin_o A$   
**shows**  $(vinsert [a, b]_o r) \uparrow^l. A = r \uparrow^l. A$   
 $\langle proof \rangle$

**lemma** *flrestriction-vinsert-in*:  
**assumes**  $a \in_o A$   
**shows**  $(vinsert [a, b]_o r) \uparrow^l. A = vinsert [a, b]_o (r \uparrow^l. A)$   
 $\langle proof \rangle$

**lemma** *flrestriction-vintersection*:  $(r \cap_o s) \uparrow^l. A = r \uparrow^l. A \cap_o s \uparrow^l. A$   $\langle proof \rangle$

**lemma** *flrestriction-vunion*:  $(r \cup_o s) \uparrow^l. A = r \uparrow^l. A \cup_o s \uparrow^l. A$   $\langle proof \rangle$

**lemma** *flrestriction-vdiff*:  $(r -_o s) \uparrow^l. A = r \uparrow^l. A -_o s \uparrow^l. A$   $\langle proof \rangle$

Connections.

**lemma** *flrestriction-fid-on[simp]*:  $(fid-on A) \uparrow^l. B = fid-on (A \cap_o B)$   $\langle proof \rangle$

**lemma** *flrestriction-fconst-on*:  $(fconst-on A c) \uparrow^l. B = (fconst-on B c) \uparrow^l. A$   
 $\langle proof \rangle$

**lemma** *flrestriction-fconst-on-commute*:  
**assumes**  $x \in_o fconst-on A c \uparrow^l. B$   
**shows**  $x \in_o fconst-on B c \uparrow^l. A$   
 $\langle proof \rangle$

**lemma** *flrestriction-fcomp[simp]*:  $(r \circ_o s) \uparrow^l. A = r \circ_o (s \uparrow^l. A)$   $\langle proof \rangle$

Previous connections.

**lemma** *fcomp-rel-fid-on[simp]*:  $r \circ_o fid-on A = r \uparrow^l. A$   $\langle proof \rangle$

**lemma** *fcomp-fconst-on*:  
 $r \circ_o (fconst-on A c) = (r \uparrow^l. set \{c\}) \circ_o (fconst-on A c)$   
 $\langle proof \rangle$

Special properties.

**lemma** *flrestriction-vsubset-fpairs*:  $r \uparrow^l. A \subseteq_o fpairs r$   
 $\langle proof \rangle$

**lemma** *flrestriction-vsubset-frel*:  $r \uparrow^l. A \subseteq_o r$   $\langle proof \rangle$

**Right restriction**

**definition** *frrestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow^r \bullet \rangle$  80)  
**where**  $r \uparrow^r \bullet A = \text{set } \{[a, b]_{\circ} \mid a \ b. \ b \in_{\circ} A \wedge [a, b]_{\circ} \in_{\circ} r\}$

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

Rules.

**lemma** *frrestrictionI[intro!]*:  
**assumes**  $b \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$   
**shows**  $[a, b]_{\circ} \in_{\circ} r \uparrow^r \bullet A$   
 $\langle \text{proof} \rangle$

**lemma** *frrestrictionD[dest]*:  
**assumes**  $[a, b]_{\circ} \in_{\circ} r \uparrow^r \bullet A$   
**shows**  $b \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$   
 $\langle \text{proof} \rangle$

**lemma** *frrestrictionE[elim!]*:  
**assumes**  $x \in_{\circ} r \uparrow^r \bullet A$   
**obtains**  $a \ b$  **where**  $x = [a, b]_{\circ}$  **and**  $b \in_{\circ} A$  **and**  $[a, b]_{\circ} \in_{\circ} r$   
 $\langle \text{proof} \rangle$

Set operations.

**lemma** *frrestriction-on-vempty[simp]*:  $r \uparrow^r \bullet 0 = 0$   $\langle \text{proof} \rangle$

**lemma** *frrestriction-vempty[simp]*:  $0 \uparrow^r \bullet A = 0$   $\langle \text{proof} \rangle$

**lemma** *frrestriction-usingleton-in[simp]*:  
**assumes**  $b \in_{\circ} A$   
**shows**  $\text{set } \{[a, b]_{\circ}\} \uparrow^r \bullet A = \text{set } \{[a, b]_{\circ}\}$   
 $\langle \text{proof} \rangle$

**lemma** *frrestriction-usingleton-nin[simp]*:  
**assumes**  $b \notin_{\circ} A$   
**shows**  $\text{set } \{[a, b]_{\circ}\} \uparrow^r \bullet A = 0$   
 $\langle \text{proof} \rangle$

**lemma** *frrestriction-mono*:  
**assumes**  $A \subseteq_{\circ} B$   
**shows**  $r \uparrow^r \bullet A \subseteq_{\circ} r \uparrow^r \bullet B$   
 $\langle \text{proof} \rangle$

**lemma** *frrestriction-vinsert-nin[simp]*:  
**assumes**  $b \notin_{\circ} A$   
**shows**  $(\text{vinsert } [a, b]_{\circ} r) \uparrow^r \bullet A = r \uparrow^r \bullet A$   
 $\langle \text{proof} \rangle$

**lemma** *frrestriction-vinsert-in*:  
**assumes**  $b \in_{\circ} A$   
**shows**  $(\text{vinsert } [a, b]_{\circ} r) \uparrow^r \bullet A = \text{vinsert } [a, b]_{\circ} (r \uparrow^r \bullet A)$   
 $\langle \text{proof} \rangle$

**lemma** *frrestriction-vintersection*:  $(r \cap_{\circ} s) \uparrow^r \bullet A = r \uparrow^r \bullet A \cap_{\circ} s \uparrow^r \bullet A$   $\langle \text{proof} \rangle$

**lemma** *frrestriction-vunion*:  $(r \cup_{\circ} s) \uparrow^r \bullet A = r \uparrow^r \bullet A \cup_{\circ} s \uparrow^r \bullet A$   $\langle \text{proof} \rangle$

**lemma** *frrestriction-vdiff*:  $(r \circ_0 s) \uparrow^r \bullet A = r \uparrow^r \bullet A \circ_0 s \uparrow^r \bullet A$  *<proof>*

Connections.

**lemma** *frrestriction-fid-on[simp]*:  $(fid\text{-}on\ A) \uparrow^r \bullet B = fid\text{-}on\ (A \cap_0 B)$  *<proof>*

**lemma** *frrestriction-fconst-on*:

**assumes**  $c \in_0 B$

**shows**  $(fconst\text{-}on\ A\ c) \uparrow^r \bullet B = fconst\text{-}on\ A\ c$   
*<proof>*

**lemma** *frrestriction-fcomp[simp]*:  $(r \circ_0 s) \uparrow^r \bullet A = (r \uparrow^r \bullet A) \circ_0 s$  *<proof>*

Previous connections.

**lemma** *fcomp-fid-on-rel[simp]*:  $fid\text{-}on\ A \circ_0 r = r \uparrow^r \bullet A$  *<proof>*

**lemma** *fcomp-fconst-on-rel*:  $(fconst\text{-}on\ A\ c) \circ_0 r = (fconst\text{-}on\ A\ c) \circ_0 (r \uparrow^r \bullet A)$   
*<proof>*

**lemma** *frrestriction-fconverse*:  $r^{-1} \bullet \uparrow^l \bullet A = (r \uparrow^r \bullet A)^{-1} \bullet$  *<proof>*

**lemma** *frrestriction-fconverse*:  $r^{-1} \bullet \uparrow^r \bullet A = (r \uparrow^l \bullet A)^{-1} \bullet$  *<proof>*

Special properties.

**lemma** *frrestriction-vsubset-rel*:  $r \uparrow^r \bullet A \subseteq_0 r$  *<proof>*

**lemma** *frrestriction-vsubset-vpairs*:  $r \uparrow^r \bullet A \subseteq_0 fpairs\ r$  *<proof>*

## Restriction

**definition** *frrestriction* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \uparrow_\bullet \rangle$  80)

**where**  $r \uparrow_\bullet A = set\ \{[a, b]_0 \mid a\ b.\ a \in_0 A \wedge b \in_0 A \wedge [a, b]_0 \in_0 r\}$

**lemma** *frrestriction-small[simp]*:

*small*  $\{[a, b]_0 \mid a\ b.\ a \in_0 A \wedge b \in_0 A \wedge [a, b]_0 \in_0 r\}$   
*<proof>*

Rules.

**lemma** *frrestrictionI[intro!]*:

**assumes**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $[a, b]_0 \in_0 r$   
**shows**  $[a, b]_0 \in_0 r \uparrow_\bullet A$   
*<proof>*

**lemma** *frrestrictionD[dest]*:

**assumes**  $[a, b]_0 \in_0 r \uparrow_\bullet A$   
**shows**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $[a, b]_0 \in_0 r$   
*<proof>*

**lemma** *frrestrictionE[elim!]*:

**assumes**  $x \in_0 r \uparrow_\bullet A$   
**obtains**  $a\ b$  **where**  $x = [a, b]_0$  **and**  $a \in_0 A$  **and**  $b \in_0 A$  **and**  $[a, b]_0 \in_0 r$   
*<proof>*

Set operations.

**lemma** *frrestriction-on-vempty[simp]*:  $r \uparrow_\bullet 0 = 0$  *<proof>*

**lemma** *frrestriction-vempty[simp]*:  $0 \uparrow_\bullet A = 0$  *<proof>*

**lemma** *frestriction-vsingleton-in[simp]*:  
**assumes**  $a \in_o A$  **and**  $b \in_o A$   
**shows**  $set \{[a, b]_o\} \vdash_o A = set \{[a, b]_o\}$   
 $\langle proof \rangle$

**lemma** *frestriction-vsingleton-nin-left[simp]*:  
**assumes**  $a \notin_o A$   
**shows**  $set \{[a, b]_o\} \vdash_o A = 0$   
 $\langle proof \rangle$

**lemma** *frestriction-vsingleton-nin-right[simp]*:  
**assumes**  $b \notin_o A$   
**shows**  $set \{[a, b]_o\} \vdash_o A = 0$   
 $\langle proof \rangle$

**lemma** *frestriction-mono*:  
**assumes**  $A \subseteq_o B$   
**shows**  $r \vdash_o A \subseteq_o r \vdash_o B$   
 $\langle proof \rangle$

**lemma** *frestriction-vinsert-nin[simp]*:  
**assumes**  $a \notin_o A$  **and**  $b \notin_o A$   
**shows**  $(vinsert [a, b]_o r) \vdash_o A = r \vdash_o A$   
 $\langle proof \rangle$

**lemma** *frestriction-vinsert-in*:  
**assumes**  $a \in_o A$  **and**  $b \in_o A$   
**shows**  $(vinsert [a, b]_o r) \vdash_o A = vinsert [a, b]_o (r \vdash_o A)$   
 $\langle proof \rangle$

**lemma** *frestriction-vintersection*:  $(r \cap_o s) \vdash_o A = r \vdash_o A \cap_o s \vdash_o A$   $\langle proof \rangle$

**lemma** *frestriction-vunion*:  $(r \cup_o s) \vdash_o A = r \vdash_o A \cup_o s \vdash_o A$   $\langle proof \rangle$

**lemma** *frestriction-vdiff*:  $(r -_o s) \vdash_o A = r \vdash_o A -_o s \vdash_o A$   $\langle proof \rangle$

Connections.

**lemma** *fid-on-frestriction[simp]*:  $(fid\text{-on } A) \vdash_o B = fid\text{-on } (A \cap_o B)$   $\langle proof \rangle$

**lemma** *frestriction-fconst-on-ex*:  
**assumes**  $c \in_o B$   
**shows**  $(fconst\text{-on } A c) \vdash_o B = fconst\text{-on } (A \cap_o B) c$   
 $\langle proof \rangle$

**lemma** *frestriction-fconst-on-nex*:  
**assumes**  $c \notin_o B$   
**shows**  $(fconst\text{-on } A c) \vdash_o B = 0$   
 $\langle proof \rangle$

**lemma** *frestriction-fcomp[simp]*:  $(r \circ_o s) \vdash_o A = (r \uparrow^r \cdot A) \circ_o (s \uparrow^l \cdot A)$   $\langle proof \rangle$

**lemma** *frestriction-fconverse*:  $r^{-1} \cdot \vdash_o A = (r \vdash_o A)^{-1} \cdot \langle proof \rangle$

Previous connections.

**lemma** *frrestriction-flrestriction[simp]*:  $(r \uparrow^r \cdot A) \uparrow^l \cdot A = r \vdash_o A$   $\langle proof \rangle$

**lemma** *flrestriction-frrestriction[simp]*:  $(r \uparrow^l \cdot A) \uparrow^r \cdot A = r \vdash_o A$   $\langle proof \rangle$

**lemma** *frestriction-frestriction*[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*:  $r \upharpoonright_{\bullet} A \subseteq_{\circ} \text{fpairs } r$  *<proof>*

**lemma** *frestriction-vsubset-ftimes*:  $r \upharpoonright_{\bullet} A \subseteq_{\circ} A \hat{\times}_{\mathbb{N}}$  *<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  $\mathcal{D}_{\bullet} r = \text{set } \{a. \exists b. [a, b]_{\circ} \in_{\circ} r\}$

**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]:  $\mathcal{D}_{\bullet} (\text{set } \{[a, b]_{\circ}\}) = \text{set } \{a\}$  *<proof>*

**lemma** *fdomain-vdoubleton*[simp]:  $\mathcal{D}_{\bullet} (\text{set } \{[a, b]_{\circ}, [c, d]_{\circ}\}) = \text{set } \{a, c\}$   
*<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]:  $\mathcal{D}_{\bullet} (\text{vinsert } [a, b]_{\circ} r) = \text{vinsert } a (\mathcal{D}_{\bullet} r)$   
*<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]*:  $\mathcal{D}_\bullet (\text{fid-on } A) = A$  *<proof>*

**lemma** *fdomain-fconst-on[simp]*:  $\mathcal{D}_\bullet (\text{fconst-on } A \ c) = A$  *<proof>*

**lemma** *fdomain-flrestriction*:  $\mathcal{D}_\bullet (r \upharpoonright_\bullet A) = \mathcal{D}_\bullet r \cap_\circ A$  *<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** *frange*  $r = \text{set } \{b. \exists a. [a, b]_\circ \in_\circ r\}$

**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*:  $b \in_\circ \mathcal{R}_\bullet r \iff (\exists a. [a, b]_\circ \in_\circ r)$  *<proof>*

Set operations.

**lemma** *frange-vempty[simp]*:  $\mathcal{R}_\bullet 0 = 0$  *<proof>*

**lemma** *frange-vsingleton[simp]*:  $\mathcal{R}_\bullet (\text{set } \{[a, b]_\circ\}) = \text{set } \{b\}$  *<proof>*

**lemma** *frange-vdoubleton[simp]*:  $\mathcal{R}_\bullet (\text{set } \{[a, b]_\circ, [c, d]_\circ\}) = \text{set } \{b, d\}$   
*<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]*:  $\mathcal{R}_\bullet$   $(vinsert [a, b]_\circ r) = vinsert b (\mathcal{R}_\bullet r)$  ⟨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]*:  $\mathcal{R}_\bullet$   $(fid-on A) = A$  ⟨proof⟩

**lemma** *frange-fconst-on-vempty[simp]*:  $\mathcal{R}_\bullet$   $(fconst-on 0 c) = 0$  ⟨proof⟩

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

**lemma** *frange-vrestriction*:  $\mathcal{R}_\bullet$   $(r \upharpoonright_\bullet A) = \mathcal{R}_\bullet r \cap_\circ A$  ⟨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*:  $b \in_\circ \mathcal{R}_\bullet r \iff (\exists a \in_\circ \mathcal{D}_\bullet r. [a, b]_\circ \in_\circ r)$  ⟨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** *ffield*  $r = \mathcal{D}_\bullet r \cup_\circ \mathcal{R}_\bullet r$

**abbreviation** *app-ffield* ::  $V \Rightarrow V$  ( $\langle \mathcal{F}_\bullet \rangle$ )  
**where**  $\mathcal{F}_\bullet r \equiv ffield r$

Rules.

**lemma** *ffieldI1[intro]*:  
**assumes**  $a \in_\circ \mathcal{D}_\bullet r \cup_\circ \mathcal{R}_\bullet r$   
**shows**  $a \in_\circ ffield r$   
 ⟨proof⟩

**lemma** *ffieldI2[intro]*:

**assumes**  $[a, b]_o \in_o r$   
**shows**  $a \in_o \text{ffield } r$   
 $\langle \text{proof} \rangle$

**lemma** *ffieldI3*[*intro*]:  
**assumes**  $[a, b]_o \in_o r$   
**shows**  $b \in_o \text{ffield } r$   
 $\langle \text{proof} \rangle$

**lemma** *ffieldD*[*intro*]:  
**assumes**  $a \in_o \text{ffield } r$   
**shows**  $a \in_o \mathcal{D}_\bullet r \cup_o \mathcal{R}_\bullet r$   
 $\langle \text{proof} \rangle$

**lemma** *ffieldE*[*elim*]:  
**assumes**  $a \in_o \text{ffield } r$  **and**  $a \in_o \mathcal{D}_\bullet r \cup_o \mathcal{R}_\bullet r \implies P$   
**shows**  $P$   
 $\langle \text{proof} \rangle$

**lemma** *ffield-pair*[*elim*]:  
**assumes**  $a \in_o \text{ffield } r$   
**obtains**  $b$  **where**  $[a, b]_o \in_o r \vee [b, a]_o \in_o r$   
 $\langle \text{proof} \rangle$

**lemma** *ffield-iff*:  $a \in_o \text{ffield } r \iff (\exists b. [a, b]_o \in_o r \vee [b, a]_o \in_o r)$   $\langle \text{proof} \rangle$

Set operations.

**lemma** *ffield-vempty*[*simp*]:  $\text{ffield } 0 = 0$   $\langle \text{proof} \rangle$

**lemma** *ffield-vsingleton*[*simp*]:  $\text{ffield } (\text{set } \{[a, b]_o\}) = \text{set } \{a, b\}$   $\langle \text{proof} \rangle$

**lemma** *ffield-vdoubleton*[*simp*]:  
 $\text{ffield } (\text{set } \{[a, b]_o, [c, d]_o\}) = \text{set } \{a, b, c, d\}$   
 $\langle \text{proof} \rangle$

**lemma** *ffield-mono*:  
**assumes**  $r \subseteq_o s$   
**shows**  $\text{ffield } r \subseteq_o \text{ffield } s$   
 $\langle \text{proof} \rangle$

**lemma** *ffield-vinsert*[*simp*]:  
 $\text{ffield } (\text{vinsert } [a, b]_o r) = \text{set } \{a, b\} \cup_o (\text{ffield } r)$   
 $\langle \text{proof} \rangle$

**lemma** *ffield-vunion*[*simp*]:  $\text{ffield } (r \cup_o s) = \text{ffield } r \cup_o \text{ffield } s$   
 $\langle \text{proof} \rangle$

Connections.

**lemma** *fid-on-ffield*[*simp*]:  $\text{ffield } (\text{fid-on } A) = A$   $\langle \text{proof} \rangle$

**lemma** *fconst-on-ffield-ne*[*intro, simp*]:  
**assumes**  $A \neq 0$   
**shows**  $\text{ffield } (\text{fconst-on } A c) = \text{vinsert } c A$   
 $\langle \text{proof} \rangle$

**lemma** *fconst-on-ffield-vempty*[*simp*]:  $\text{ffield } (\text{fconst-on } 0 c) = 0$   $\langle \text{proof} \rangle$

**lemma** *ffield-fconverse*[*simp*]:  $\text{ffield } (r^{-1}\bullet) = \text{ffield } r$   $\langle \text{proof} \rangle$

Special properties.

**lemma** *ffield-vsubset-VUnion2*:  $\mathcal{F}_\bullet \cdot r \subseteq_o \cup_o(\cup_o(\cup_o r))$   
 ⟨proof⟩

### Image

**definition** *fimage* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle \cdot, \cdot \rangle$  90)

where  $r \cdot A = \mathcal{R}_\bullet (r \uparrow^l \cdot A)$

**notation** *fimage* (**infixr**  $\langle \cdot, \cdot \rangle$  90)

**lemma** *fimage-small[simp]*: *small*  $\{b. \exists a \in_o A. [a, b]_o \in_o r\}$   
 ⟨proof⟩

Rules.

**lemma** *fimageI1*:

**assumes**  $x \in_o \mathcal{R}_\bullet (r \uparrow^l \cdot A)$

**shows**  $x \in_o r \cdot A$

⟨proof⟩

**lemma** *fimageI2[intro]*:

**assumes**  $[a, b]_o \in_o r$  **and**  $a \in_o A$

**shows**  $b \in_o r \cdot A$

⟨proof⟩

**lemma** *fimageD[dest]*:

**assumes**  $x \in_o r \cdot A$

**shows**  $x \in_o \mathcal{R}_\bullet (r \uparrow^l \cdot A)$

⟨proof⟩

**lemma** *fimageE[elim]*:

**assumes**  $b \in_o r \cdot A$

**obtains**  $a$  **where**  $[a, b]_o \in_o r$  **and**  $a \in_o A$

⟨proof⟩

**lemma** *fimage-iff*:  $b \in_o r \cdot A \longleftrightarrow (\exists a \in_o A. [a, b]_o \in_o r)$  ⟨proof⟩

Set operations.

**lemma** *fimage-vempty[simp]*:  $0 \cdot A = 0$  ⟨proof⟩

**lemma** *fimage-of-vempty[simp]*:  $r \cdot 0 = 0$  ⟨proof⟩

**lemma** *fimage-vsingleton-in[intro, simp]*:

**assumes**  $a \in_o A$

**shows**  $\text{set } \{[a, b]_o\} \cdot A = \text{set } \{b\}$

⟨proof⟩

**lemma** *fimage-vsingleton-nin[intro, simp]*:

**assumes**  $a \notin_o A$

**shows**  $\text{set } \{[a, b]_o\} \cdot A = 0$

⟨proof⟩

**lemma** *fimage-vsingleton-vinsert[intro, simp]*:

$\text{set } \{[a, b]_o\} \cdot \text{vinsert } a \ A = \text{set } \{b\}$

⟨proof⟩

**lemma** *fimage-mono*:

**assumes**  $r' \subseteq_o r$  **and**  $A' \subseteq_o A$

**shows**  $(r' \dot{\cdot} A') \subseteq_o (r \dot{\cdot} A)$   
 ⟨proof⟩

**lemma** *fimage-vinsert*:  $r \dot{\cdot} (\text{vinsert } a \ A) = r \dot{\cdot} \text{set } \{a\} \cup_o r \dot{\cdot} A$  ⟨proof⟩

**lemma** *fimage-vunion-left*:  $(r \cup_o s) \dot{\cdot} A = r \dot{\cdot} A \cup_o s \dot{\cdot} A$  ⟨proof⟩

**lemma** *fimage-vunion-right*:  $r \dot{\cdot} (A \cup_o B) = r \dot{\cdot} A \cup_o r \dot{\cdot} B$  ⟨proof⟩

**lemma** *fimage-vintersection*:  $r \dot{\cdot} (A \cap_o B) \subseteq_o r \dot{\cdot} A \cap_o r \dot{\cdot} B$  ⟨proof⟩

**lemma** *fimage-vdiff*:  $r \dot{\cdot} A -_o r \dot{\cdot} B \subseteq_o r \dot{\cdot} (A -_o B)$  ⟨proof⟩

Special properties.

**lemma** *fimage-vsingleton-iff*[iff]:  $b \in_o r \dot{\cdot} \text{set } \{a\} \leftrightarrow [a, b]_o \in_o r$  ⟨proof⟩

**lemma** *fimage-is-vempty*[iff]:  $r \dot{\cdot} A = 0 \leftrightarrow \text{vdisjnt } (\mathcal{D} \bullet r) \ A$  ⟨proof⟩

Connections.

**lemma** *fid-on-fimage*[simp]:  $(\text{fid-on } A) \dot{\cdot} B = A \cap_o B$  ⟨proof⟩

**lemma** *fimage-fconst-on-ne*[simp]:  
**assumes**  $B \cap_o A \neq 0$   
**shows**  $(\text{fconst-on } A \ c) \dot{\cdot} B = \text{set } \{c\}$   
 ⟨proof⟩

**lemma** *fimage-fconst-on-vempty*[simp]:  
**assumes**  $\text{vdisjnt } A \ B$   
**shows**  $(\text{fconst-on } A \ c) \dot{\cdot} B = 0$   
 ⟨proof⟩

**lemma** *fimage-fconst-on-vsubset-const*[simp]:  $(\text{fconst-on } A \ c) \dot{\cdot} B \subseteq_o \text{set } \{c\}$   
 ⟨proof⟩

**lemma** *fcomp-frange*:  $\mathcal{R} \bullet (r \circ_\bullet s) = r \dot{\cdot} (\mathcal{R} \bullet s)$  ⟨proof⟩

**lemma** *fcomp-fimage*:  $(r \circ_\bullet s) \dot{\cdot} A = r \dot{\cdot} (s \dot{\cdot} A)$  ⟨proof⟩

**lemma** *fimage-ftrestriction*[simp]:  $(r \uparrow^l \bullet A) \dot{\cdot} B = r \dot{\cdot} (A \cap_o B)$  ⟨proof⟩

**lemma** *fimage-frrestriction*[simp]:  $(r \uparrow^r \bullet A) \dot{\cdot} B = A \cap_o r \dot{\cdot} B$  ⟨proof⟩

**lemma** *fimage-frestriction*[simp]:  $(r \uparrow \bullet A) \dot{\cdot} B = A \cap_o (r \dot{\cdot} (A \cap_o B))$  ⟨proof⟩

**lemma** *fimage-fdomain*:  $r \dot{\cdot} \mathcal{D} \bullet r = \mathcal{R} \bullet r$  ⟨proof⟩

**lemma** *fimage-eq-imp-fcomp*:  
**assumes**  $f \dot{\cdot} A = g \dot{\cdot} B$   
**shows**  $(h \circ_\bullet f) \dot{\cdot} A = (h \circ_\bullet g) \dot{\cdot} B$   
 ⟨proof⟩

Previous connections.

**lemma** *fcomp-rel-fconst-on-ftimes*:  $r \circ_\bullet (\text{fconst-on } A \ c) = A \times_\bullet (r \dot{\cdot} \text{set } \{c\})$   
 ⟨proof⟩

Further special properties.

**lemma** *fimage-vsubset*:

**assumes**  $r \subseteq_0 A \times_0 B$   
**shows**  $r \dot{\subseteq}_0 C \subseteq_0 B$   
 ⟨proof⟩

**lemma** *image-set-def*:  $r \dot{\subseteq}_0 A = \text{set } \{b. \exists a \in_0 A. [a, b]_0 \in_0 r\}$   
 ⟨proof⟩

**lemma** *image-singleton*:  $r \dot{\subseteq}_0 \text{set } \{a\} = \text{set } \{b. [a, b]_0 \in_0 r\}$   
 ⟨proof⟩

**lemma** *image-strict-vsubset*:  $f \dot{\subseteq}_0 A \subseteq_0 f \dot{\subseteq}_0 D \bullet f$  ⟨proof⟩

### Inverse image

**definition** *finvimage* ::  $V \Rightarrow V \Rightarrow V$  (**infixr**  $\langle - \dot{\subseteq}_0 \rangle$  90)  
 where  $r - \dot{\subseteq}_0 A = r^{-1} \dot{\subseteq}_0 A$

**lemma** *finvimage-small[simp]*: *small*  $\{a. \exists b \in_0 A. [a, b]_0 \in_0 r\}$   
 ⟨proof⟩

Rules.

**lemma** *finvimageI[intro]*:  
**assumes**  $[a, b]_0 \in_0 r$  **and**  $b \in_0 A$   
**shows**  $a \in_0 r - \dot{\subseteq}_0 A$   
 ⟨proof⟩

**lemma** *finvimageD[dest]*:  
**assumes**  $a \in_0 r - \dot{\subseteq}_0 A$   
**shows**  $a \in_0 D \bullet (r \uparrow^r \bullet A)$   
 ⟨proof⟩

**lemma** *finvimageE[elim]*:  
**assumes**  $a \in_0 r - \dot{\subseteq}_0 A$   
**obtains**  $b$  **where**  $[a, b]_0 \in_0 r$  **and**  $b \in_0 A$   
 ⟨proof⟩

**lemma** *finvimageI1*:  
**assumes**  $a \in_0 D \bullet (r \uparrow^r \bullet A)$   
**shows**  $a \in_0 r - \dot{\subseteq}_0 A$   
 ⟨proof⟩

**lemma** *finvimageD1*:  
**assumes**  $a \in_0 r - \dot{\subseteq}_0 A$   
**shows**  $a \in_0 D \bullet (r \uparrow^r \bullet A)$   
 ⟨proof⟩

**lemma** *finvimageE1*:  
**assumes**  $a \in_0 r - \dot{\subseteq}_0 A$  **and**  $a \in_0 D \bullet (r \uparrow^r \bullet A) \implies P$   
**shows**  $P$   
 ⟨proof⟩

**lemma** *finvimageI2*:  
**assumes**  $a \in_0 r^{-1} \dot{\subseteq}_0 A$   
**shows**  $a \in_0 r - \dot{\subseteq}_0 A$   
 ⟨proof⟩

**lemma** *finvimageD2*:  
**assumes**  $a \in_0 r - \dot{\subseteq}_0 A$

**shows**  $a \in_{\circ} r^{-1} \bullet \cdot A$   
 ⟨proof⟩

**lemma** *finvimageE2*:

**assumes**  $a \in_{\circ} r^{-1} \bullet \cdot A$  **and**  $a \in_{\circ} r^{-1} \bullet \cdot A \implies P$   
**shows**  $P$   
 ⟨proof⟩

**lemma** *finvimage-iff*:  $a \in_{\circ} r^{-1} \bullet \cdot A \longleftrightarrow (\exists b \in_{\circ} A. [a, b]_{\circ} \in_{\circ} r)$  ⟨proof⟩

**lemma** *finvimage-iff1*:  $a \in_{\circ} r^{-1} \bullet \cdot A \longleftrightarrow a \in_{\circ} \mathcal{D} \bullet (r \uparrow^r \bullet A)$  ⟨proof⟩

**lemma** *finvimage-iff2*:  $a \in_{\circ} r^{-1} \bullet \cdot A \longleftrightarrow a \in_{\circ} r^{-1} \bullet \cdot A$  ⟨proof⟩

Set operations.

**lemma** *finvimage-vempty[simp]*:  $0^{-1} \bullet \cdot A = 0$  ⟨proof⟩

**lemma** *finvimage-of-vempty[simp]*:  $r^{-1} \bullet \cdot 0 = 0$  ⟨proof⟩

**lemma** *finvimage-vsingleton-in[intro, simp]*:

**assumes**  $b \in_{\circ} A$   
**shows**  $\text{set } \{[a, b]_{\circ}\}^{-1} \bullet \cdot A = \text{set } \{a\}$   
 ⟨proof⟩

**lemma** *finvimage-vsingleton-nin[intro, simp]*:

**assumes**  $b \notin_{\circ} A$   
**shows**  $\text{set } \{[a, b]_{\circ}\}^{-1} \bullet \cdot A = 0$   
 ⟨proof⟩

**lemma** *finvimage-vsingleton-vinsert[intro, simp]*:

$\text{set } \{[a, b]_{\circ}\}^{-1} \bullet \cdot \text{vinsert } b A = \text{set } \{a\}$   
 ⟨proof⟩

**lemma** *finvimage-mono*:

**assumes**  $r' \subseteq_{\circ} r$  **and**  $A' \subseteq_{\circ} A$   
**shows**  $(r'^{-1} \bullet \cdot A') \subseteq_{\circ} (r^{-1} \bullet \cdot A)$   
 ⟨proof⟩

**lemma** *finvimage-vinsert*:  $r^{-1} \bullet \cdot (\text{vinsert } a A) = r^{-1} \bullet \cdot \text{set } \{a\} \cup_{\circ} r^{-1} \bullet \cdot A$  ⟨proof⟩

**lemma** *finvimage-vunion-left*:  $(r \cup_{\circ} s)^{-1} \bullet \cdot A = r^{-1} \bullet \cdot A \cup_{\circ} s^{-1} \bullet \cdot A$  ⟨proof⟩

**lemma** *finvimage-vunion-right*:  $r^{-1} \bullet \cdot (A \cup_{\circ} B) = r^{-1} \bullet \cdot A \cup_{\circ} r^{-1} \bullet \cdot B$  ⟨proof⟩

**lemma** *finvimage-vintersection*:  $r^{-1} \bullet \cdot (A \cap_{\circ} B) \subseteq_{\circ} r^{-1} \bullet \cdot A \cap_{\circ} r^{-1} \bullet \cdot B$  ⟨proof⟩

**lemma** *finvimage-vdiff*:  $r^{-1} \bullet \cdot A -_{\circ} r^{-1} \bullet \cdot B \subseteq_{\circ} r^{-1} \bullet \cdot (A -_{\circ} B)$  ⟨proof⟩

Special properties.

**lemma** *finvimage-set-def*:  $r^{-1} \bullet \cdot A = \text{set } \{a. \exists b \in_{\circ} A. [a, b]_{\circ} \in_{\circ} r\}$  ⟨proof⟩

**lemma** *finvimage-eq-fdomain-frestriction*:  $r^{-1} \bullet \cdot A = \mathcal{D} \bullet (r \uparrow^r \bullet A)$  ⟨proof⟩

**lemma** *finvimage-frange[simp]*:  $r^{-1} \bullet \cdot \mathcal{R} \bullet r = \mathcal{D} \bullet r$   
 ⟨proof⟩

**lemma** *finvimage-frange-vsubset[simp]*:

**assumes**  $\mathcal{R} \bullet r \subseteq_{\circ} B$

**shows**  $r - \cdot B = \mathcal{D} \cdot r$   
 ⟨proof⟩

Connections.

**lemma** *finvimage-fid-on[simp]*:  $(\text{fid-on } A) - \cdot B = A \cap_o B$  ⟨proof⟩

**lemma** *finvimage-fconst-on-vsubset-fdomain[simp]*:  $(\text{fconst-on } A \ c) - \cdot B \subseteq_o A$   
 ⟨proof⟩

**lemma** *finvimage-fconst-on-ne[simp]*:  
**assumes**  $c \in_o B$   
**shows**  $(\text{fconst-on } A \ c) - \cdot B = A$   
 ⟨proof⟩

**lemma** *finvimage-fconst-on-vempty[simp]*:  
**assumes**  $c \notin_o B$   
**shows**  $(\text{fconst-on } A \ c) - \cdot B = 0$   
 ⟨proof⟩

**lemma** *finvimage-fcomp*:  $(g \circ_o f) - \cdot x = f - \cdot (g - \cdot x)$   
 ⟨proof⟩

**lemma** *finvimage-fconverse[simp]*:  $r^{-1} \cdot - \cdot A = r \cdot A$  ⟨proof⟩

**lemma** *finvimage-frestriction[simp]*:  $(r \upharpoonright^l \cdot A) - \cdot B = A \cap_o r - \cdot B$  ⟨proof⟩

**lemma** *finvimage-frrestriction[simp]*:  $(r \upharpoonright^r \cdot A) - \cdot B = (r - \cdot (A \cap_o B))$  ⟨proof⟩

**lemma** *finvimage-frestriction[simp]*:  $(r \upharpoonright \cdot A) - \cdot B = A \cap_o (r - \cdot (A \cap_o B))$   
 ⟨proof⟩

Previous connections.

**lemma** *fdomain-fcomp[simp]*:  $\mathcal{D} \cdot (r \circ_o s) = s - \cdot \mathcal{D} \cdot r$  ⟨proof⟩

## 2.10.5 Classification of relations

### Binary relation

**locale** *fbrelation* =  
**fixes**  $r :: V$   
**assumes** *fbrelation[simp]*:  $\text{fpairs } r = r$

**locale** *fbrelation-pair* =  $r_1$ : *fbrelation*  $r_1$  +  $r_2$ : *fbrelation*  $r_2$  **for**  $r_1 \ r_2$

Rules.

**lemma** *fpairs-eqI[intro!]*:  
**assumes**  $\bigwedge x. x \in_o r \implies \exists a \ b. x = [a, b]_o$   
**shows**  $\text{fpairs } r = r$   
 ⟨proof⟩

**lemma** *fpairs-eqD[dest]*:  
**assumes**  $\text{fpairs } r = r$   
**shows**  $\bigwedge x. x \in_o r \implies \exists a \ b. x = [a, b]_o$   
 ⟨proof⟩

**lemma** *fpairs-eqE[elim!]*:  
**assumes**  $\text{fpairs } r = r$  **and**  $(\bigwedge x. x \in_o r \implies \exists a \ b. x = [a, b]_o) \implies P$   
**shows**  $P$

*<proof>*

**lemmas** *fbrelationI*[*intro!*] = *fbrelation.intro*

**lemmas** *fbrelationD*[*dest!*] = *fbrelation.fbrelation*

**lemma** *fbrelationE*[*elim!*]:

**assumes** *fbrelation r* **and** (*fpairs r = r*)  $\implies P$

**shows** *P*

*<proof>*

**lemma** *fbrelationE1*:

**assumes** *fbrelation r* **and**  $x \in_{\circ} r$

**obtains** *a b* **where**  $x = [a, b]_{\circ}$

*<proof>*

**lemma** *fbrelationD1*[*dest!*]:

**assumes** *fbrelation r* **and**  $x \in_{\circ} r$

**shows**  $\exists a b. x = [a, b]_{\circ}$

*<proof>*

Set operations.

**lemma** *fbrelation-ubset*:

**assumes** *fbrelation s* **and**  $r \subseteq_{\circ} s$

**shows** *fbrelation r*

*<proof>*

**lemma** *fbrelation-vinsert*: *fbrelation (vinsert [a, b]<sub>∘</sub> r)*  $\longleftrightarrow$  *fbrelation r*

*<proof>*

**lemma** (**in** *fbrelation*) *fbrelation-vinsertI*: *fbrelation (vinsert [a, b]<sub>∘</sub> r)*

*<proof>*

**lemma** *fbrelation-vinsertD*[*dest!*]:

**assumes** *fbrelation (vinsert [a, b]<sub>∘</sub> r)*

**shows** *fbrelation r*

*<proof>*

**lemma** *fbrelation-vunion*: *fbrelation (r ∪<sub>∘</sub> s)*  $\longleftrightarrow$  *fbrelation r*  $\wedge$  *fbrelation s*

*<proof>*

**lemma** (**in** *fbrelation-pair*) *fbrelation-vunionI*: *fbrelation (r<sub>1</sub> ∪<sub>∘</sub> r<sub>2</sub>)*

*<proof>*

**lemma** *fbrelation-vunionD*[*dest!*]:

**assumes** *fbrelation (r ∪<sub>∘</sub> s)*

**shows** *fbrelation r* **and** *fbrelation s*

*<proof>*

**lemma** (**in** *fbrelation*) *fbrelation-vintersectionI*: *fbrelation (r ∩<sub>∘</sub> s)*

*<proof>*

**lemma** (**in** *fbrelation*) *fbrelation-vdiffI*: *fbrelation (r -<sub>∘</sub> s)*

*<proof>*

Connections.

**lemma** *fbrelation-veempty*: *fbrelation 0* *<proof>*

**lemma** *fbrelation-vsingleton*: *fbrelation (set {[a, b]<sub>∘</sub>)* *<proof>*

**global-interpretation** *frel-vsingleton*: *fbrelation*  $\langle \text{set } \{[a, b]_{\circ}\} \rangle$   
 $\langle \text{proof} \rangle$

**lemma** *fbrelation-vdoubleton*: *fbrelation*  $(\text{set } \{[a, b]_{\circ}, [c, d]_{\circ}\}) \langle \text{proof} \rangle$

**lemma** *fbrelation-sid-on[simp]*: *fbrelation*  $(\text{fid-on } A) \langle \text{proof} \rangle$

**lemma** *fbrelation-fconst-on[simp]*: *fbrelation*  $(\text{fconst-on } A \ c) \langle \text{proof} \rangle$

**lemma** (**in** *fbrelation-pair*) *fbrelation-fcomp*: *fbrelation*  $(r_1 \circ_{\bullet} r_2)$   
 $\langle \text{proof} \rangle$

**sublocale** *fbrelation-pair*  $\subseteq$  *fcomp*<sub>21</sub>: *fbrelation*  $\langle r_2 \circ_{\bullet} r_1 \rangle$   
 $\langle \text{proof} \rangle$

**sublocale** *fbrelation-pair*  $\subseteq$  *fcomp*<sub>12</sub>: *fbrelation*  $\langle r_1 \circ_{\bullet} r_2 \rangle$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation*) *fbrelation-fconverse*: *fbrelation*  $(r^{-1} \bullet)$   
 $\langle \text{proof} \rangle$

**lemma** *fbrelation-flrestriction[intro, simp]*: *fbrelation*  $(r \upharpoonright^l \bullet \ A) \langle \text{proof} \rangle$

**lemma** *fbrelation-frrestriction[intro, simp]*: *fbrelation*  $(r \upharpoonright^r \bullet \ A) \langle \text{proof} \rangle$

**lemma** *fbrelation-frestriction[intro, simp]*: *fbrelation*  $(r \upharpoonright \bullet \ A) \langle \text{proof} \rangle$

Previous connections.

**lemma** (**in** *fbrelation*) *fconverse-fconverse[simp]*:  $(r^{-1} \bullet)^{-1} \bullet = r$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation-pair*) *fconverse-mono[simp]*:  $r_1^{-1} \bullet \subseteq_{\circ} r_2^{-1} \bullet \iff r_1 \subseteq_{\circ} r_2$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation-pair*) *fconverse-inject[simp]*:  $r_1^{-1} \bullet = r_2^{-1} \bullet \iff r_1 = r_2$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation*) *fconverse-vsubset-swap-2*:

**assumes**  $r^{-1} \bullet \subseteq_{\circ} s$

**shows**  $r \subseteq_{\circ} s^{-1} \bullet$

$\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation*) *flrestriction-fdomain[simp]*:  $r \upharpoonright^l \bullet \ \mathcal{D} \bullet \ r = r$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation*) *frrestriction-frange[simp]*:  $r \upharpoonright^r \bullet \ \mathcal{R} \bullet \ r = r$   
 $\langle \text{proof} \rangle$

Special properties.

**lemma** *vsubset-vtimes-fbrelation*:

**assumes**  $r \subseteq_{\circ} A \times_{\bullet} B$

**shows** *fbrelation*  $r$

$\langle \text{proof} \rangle$

**lemma** (**in** *fbrelation*) *fbrelation-vintersection-vdomain*:

**assumes**  $v\text{disjnt } (\mathcal{D} \bullet \ r) \ (\mathcal{D} \bullet \ s)$

**shows**  $v\text{disjnt } r \ s$

*<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*:  $r \subseteq_o \mathcal{D}_\bullet r \times_\bullet \mathcal{R}_\bullet r$

*<proof>*

**lemma** (*in fbrelation*) *fconverse-eq-frel*[*intro*, *simp*]:

**assumes**  $\bigwedge a b. [a, b]_o \in_o r \implies [b, a]_o \in_o r$

**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**  $i \in_0 j$  **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**  $rank\ i \in_0 rank\ j$   
*<proof>*

Elementary corollaries.

**lemma** *Ord-Vset-in-Vset-succI[intro]*:

**assumes**  $Ord\ \alpha$   
**shows**  $Vset\ \alpha \in_0 Vset\ (succ\ \alpha)$   
*<proof>*

**lemma** *Ord-in-in-VsetI[intro]*:

**assumes**  $Ord\ \alpha$  **and**  $a \in_0 \alpha$   
**shows**  $a \in_0 Vset\ \alpha$   
*<proof>*

Transitivity of the constant  $Vfrom$ .

**lemma** *Vfrom-trans[intro]*:

**assumes**  $Transset\ A$  **and**  $x \in_0 X$  **and**  $X \in_0 Vfrom\ A\ i$   
**shows**  $x \in_0 Vfrom\ A\ i$   
*<proof>*

**lemma** *Vset-trans[intro]*:

**assumes**  $x \in_0 X$  **and**  $X \in_0 Vset\ i$   
**shows**  $x \in_0 Vset\ i$   
*<proof>*

Monotonicity of the constant  $Vfrom$ .

**lemma** *Vfrom-in-mono*:

**assumes**  $A \subseteq_0 B$  **and**  $i \in_0 j$   
**shows**  $Vfrom\ A\ i \in_0 Vfrom\ B\ j$   
*<proof>*

**lemmas**  $Vset-in-mono = Vfrom-in-mono[OF\ order-refl,\ of\ -\ 0]$

**lemma** *Vfrom-uset-mono*:

**assumes**  $A \subseteq_0 B$  **and**  $i \subseteq_0 j$   
**shows**  $V\text{from } A \ i \subseteq_0 V\text{from } B \ j$   
 $\langle \text{proof} \rangle$

**lemmas**  $V\text{set-}v\text{subset-}mono = V\text{from-}v\text{subset-}mono[OF \ \text{order-}reft, \ \text{of} \ - \ - \ 0]$

**lemma**  $arg1\text{-}v\text{subset-}V\text{from}: a \subseteq_0 V\text{from } a \ i \ \langle \text{proof} \rangle$

**lemma**  $VPow\text{-}v\text{subset-}V\text{set}:$

— Based on Theorem 9.10 from [59]

**assumes**  $X \in_0 V\text{set } i$

**shows**  $VPow \ X \subseteq_0 V\text{set } i$

$\langle \text{proof} \rangle$

**lemma**  $V\text{from-}v\text{subset-}VPow\text{-}V\text{from}:$

**assumes**  $Trans\text{set } A$

**shows**  $V\text{from } A \ i \subseteq_0 VPow \ (V\text{from } A \ i)$

$\langle \text{proof} \rangle$

**lemma**  $arg1\text{-}v\text{subset-}VPow\text{-}V\text{from}:$

**assumes**  $Trans\text{set } A$

**shows**  $A \subseteq_0 VPow \ (V\text{from } A \ i)$

$\langle \text{proof} \rangle$

### 2.11.3 Operations closed with respect to $V\text{set}$

Empty set.

**lemma**  $Limit\text{-}v\text{empty-}in\text{-}V\text{set}I:$

**assumes**  $Limit \ \alpha$

**shows**  $0 \in_0 V\text{set } \alpha$

$\langle \text{proof} \rangle$

Subset.

**lemma**  $v\text{subset-}in\text{-}V\text{set}I[\text{intro}]:$

**assumes**  $a \subseteq_0 A$  **and**  $A \in_0 V\text{set } i$

**shows**  $a \in_0 V\text{set } i$

$\langle \text{proof} \rangle$

**lemma**  $Ord\text{-}v\text{subset-}in\text{-}V\text{set-}succI:$

**assumes**  $Ord \ \alpha$  **and**  $A \subseteq_0 V\text{set } \alpha$

**shows**  $A \in_0 V\text{set } (\text{succ } \alpha)$

$\langle \text{proof} \rangle$

Power set.

**lemma**  $Limit\text{-}VPow\text{-}in\text{-}V\text{set}I[\text{intro}]:$

**assumes**  $Limit \ \alpha$  **and**  $A \in_0 V\text{set } \alpha$

**shows**  $VPow \ A \in_0 V\text{set } \alpha$

$\langle \text{proof} \rangle$

**lemma**  $VPow\text{-}in\text{-}V\text{set-}revD:$

**assumes**  $VPow \ A \in_0 V\text{set } i$

**shows**  $A \in_0 V\text{set } i$

$\langle \text{proof} \rangle$

**lemma**  $Ord\text{-}VPow\text{-}in\text{-}V\text{set-}succI:$

**assumes**  $Ord \ \alpha$  **and**  $a \in_0 V\text{set } \alpha$

**shows**  $VPow \ a \in_0 V\text{set } (\text{succ } \alpha)$

*<proof>*

**lemma** *Ord-VPow-in-Vset-succD*:

**assumes** *Ord*  $\alpha$  **and** *VPow*  $a \in_0$  *Vset* (*succ*  $\alpha$ )

**shows**  $a \in_0$  *Vset*  $\alpha$

*<proof>*

Union of elements.

**lemma** *VUnion-in-VsetI[intro]*:

**assumes**  $A \in_0$  *Vset*  $i$

**shows**  $\bigcup_0 A \in_0$  *Vset*  $i$

*<proof>*

**lemma** *Limit-VUnion-in-VsetD*:

**assumes** *Limit*  $\alpha$  **and**  $\bigcup_0 A \in_0$  *Vset*  $\alpha$

**shows**  $A \in_0$  *Vset*  $\alpha$

*<proof>*

Intersection of elements.

**lemma** *VInter-in-VsetI[intro]*:

**assumes**  $A \in_0$  *Vset*  $\alpha$

**shows**  $\bigcap_0 A \in_0$  *Vset*  $\alpha$

*<proof>*

Singleton.

**lemma** *Limit-vsingleton-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$  **and**  $a \in_0$  *Vset*  $\alpha$

**shows** *set*  $\{a\} \in_0$  *Vset*  $\alpha$

*<proof>*

**lemma** *Limit-vsingleton-in-VsetD*:

**assumes** *set*  $\{a\} \in_0$  *Vset*  $\alpha$

**shows**  $a \in_0$  *Vset*  $\alpha$

*<proof>*

**lemma** *Ord-vsingleton-in-Vset-succI*:

**assumes** *Ord*  $\alpha$  **and**  $a \in_0$  *Vset*  $\alpha$

**shows** *set*  $\{a\} \in_0$  *Vset* (*succ*  $\alpha$ )

*<proof>*

Doubleton.

**lemma** *Limit-vdoubleton-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$  **and**  $a \in_0$  *Vset*  $\alpha$  **and**  $b \in_0$  *Vset*  $\alpha$

**shows** *set*  $\{a, b\} \in_0$  *Vset*  $\alpha$

*<proof>*

**lemma** *vdoubleton-in-VsetD*:

**assumes** *set*  $\{a, b\} \in_0$  *Vset*  $\alpha$

**shows**  $a \in_0$  *Vset*  $\alpha$  **and**  $b \in_0$  *Vset*  $\alpha$

*<proof>*

**lemma** *Ord-vdoubleton-in-Vset-succI*:

**assumes** *Ord*  $\alpha$  **and**  $a \in_0$  *Vset*  $\alpha$  **and**  $b \in_0$  *Vset*  $\alpha$

**shows** *set*  $\{a, b\} \in_0$  *Vset* (*succ*  $\alpha$ )

*<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*  $\alpha$  **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*  $\alpha$  **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*  $\alpha$  **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*  $\alpha$   
**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*  $\alpha$  **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**  $\langle a, b \rangle \in_0 Vset \alpha$

**shows**  $a \in_0 Vset \alpha$  **and**  $b \in_0 Vset \alpha$

*\langle proof \rangle*

Cartesian product.

**lemma** *Limit-vtimes-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$  **and**  $A \in_0 Vset \alpha$  **and**  $B \in_0 Vset \alpha$

**shows**  $A \times_0 B \in_0 Vset \alpha$

*\langle proof \rangle*

Binary relations.

**lemma** (**in** *vbrelation*) *vbrelation-Limit-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$  **and**  $\mathcal{D}_0 r \in_0 Vset \alpha$  **and**  $\mathcal{R}_0 r \in_0 Vset \alpha$

**shows**  $r \in_0 Vset \alpha$

*\langle proof \rangle*

**lemma**

**assumes**  $r \in_0 Vset \alpha$

**shows** *vdomain-in-VsetI*:  $\mathcal{D}_0 r \in_0 Vset \alpha$

**and** *vrange-in-VsetI*:  $\mathcal{R}_0 r \in_0 Vset \alpha$

**and** *vfield-in-VsetI*:  $\mathcal{F}_0 r \in_0 Vset \alpha$

*\langle proof \rangle*

**lemma** (**in** *vbrelation*) *vbrelation-Limit-vsubset-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $\mathcal{D}_0 r \subseteq_0 Vset \alpha$  **and**  $\mathcal{R}_0 r \subseteq_0 Vset \alpha$

**shows**  $r \subseteq_0 Vset \alpha$

*\langle proof \rangle*

**lemma**

**assumes**  $r \in_0 Vset \alpha$

**shows** *fdomain-in-VsetI*:  $\mathcal{D}_\bullet r \in_0 Vset \alpha$

**and** *frange-in-VsetI*:  $\mathcal{R}_\bullet r \in_0 Vset \alpha$

**and** *ffield-in-VsetI*:  $\mathcal{F}_\bullet r \in_0 Vset \alpha$

*\langle proof \rangle*

**lemma** (**in** *vsu*) *vsu-Limit-vrange-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$  **and**  $\mathcal{R}_0 r \subseteq_0 Vset \alpha$  **and** *vfinite*  $(\mathcal{D}_0 r)$

**shows**  $\mathcal{R}_0 r \in_0 Vset \alpha$

*\langle proof \rangle*

**lemma** (**in** *vsu*) *vsu-Limit-vsu-in-VsetI[intro]*:

**assumes** *Limit*  $\alpha$

**and**  $\mathcal{D}_0 r \in_0 Vset \alpha$

**and**  $\mathcal{R}_0 r \subseteq_0 Vset \alpha$

**and** *vfinite*  $(\mathcal{D}_0 r)$

**shows**  $r \in_0 Vset \alpha$

*\langle proof \rangle*

**lemma** *Limit-vcomp-in-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $r \in_0 Vset \alpha$  **and**  $s \in_0 Vset \alpha$

**shows**  $r \circ_0 s \in_0 Vset \alpha$

*\langle proof \rangle*

Operations on indexed families of sets.

**lemma** *Limit-vifintersection-in-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $\bigwedge i. i \in_0 I \implies A i \in_0 Vset \alpha$  **and** *vfinite*  $I$

**shows**  $(\bigcap_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vifunion-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and**  $\bigwedge i. i \in_0 I \implies A_i \in_0 \text{Vset } \alpha$  **and** *vfinite*  $I$   
**shows**  $(\bigcup_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vifunion-in-Vset-if-VLambda-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and** *VLambda*  $I A \in_0 \text{Vset } \alpha$   
**shows**  $(\bigcup_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vproduct-in-VsetI:*

**assumes** *Limit*  $\alpha$   
**and**  $I \in_0 \text{Vset } \alpha$   
**and**  $\bigwedge i. i \in_0 I \implies A_i \in_0 \text{Vset } \alpha$   
**and** *vfinite*  $I$   
**shows**  $(\prod_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vproduct-in-Vset-if-VLambda-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and** *VLambda*  $I A \in_0 \text{Vset } \alpha$   
**shows**  $(\prod_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vdunion-in-Vset-if-VLambda-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and** *VLambda*  $I A \in_0 \text{Vset } \alpha$   
**shows**  $(\coprod_{i \in I} A_i) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *vrange-vprojection-in-VsetI:*

**assumes** *Limit*  $\alpha$   
**and**  $A \in_0 \text{Vset } \alpha$   
**and**  $\bigwedge f. f \in_0 A \implies \text{vsu } f$   
**and**  $\bigwedge f. f \in_0 A \implies x \in_0 \mathcal{D}_0 f$   
**shows**  $\mathcal{R}_0 (\lambda f \in_0 A. f(|x|)) \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-vcpower-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and**  $n \in_0 \text{Vset } \alpha$  **and**  $A \in_0 \text{Vset } \alpha$  **and** *vfinite*  $n$   
**shows**  $A \hat{\times}^n \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

Finite sets.

**lemma** *Limit-vfinite-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and**  $A \subseteq_0 \text{Vset } \alpha$  **and** *vfinite*  $A$   
**shows**  $A \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

Ordinal numbers.

**lemma** *Limit-omega-in-VsetI:*

**assumes** *Limit*  $\alpha$   
**shows**  $a_{\mathbb{N}} \in_0 \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** *Limit-succ-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and**  $a \in_0 \text{Vset } \alpha$

**shows**  $\text{succ } a \in_{\circ} \text{Vset } \alpha$

*<proof>*

Sequences.

**lemma** (in *vfsequence*) *vfsequence-Limit-vcons-in-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $x \in_{\circ} \text{Vset } \alpha$  **and**  $xs \in_{\circ} \text{Vset } \alpha$

**shows**  $\text{vcons } xs \ x \in_{\circ} \text{Vset } \alpha$

*<proof>*

*ftimes*.

**lemma** *Limit-ftimes-in-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $A \in_{\circ} \text{Vset } \alpha$  **and**  $B \in_{\circ} \text{Vset } \alpha$

**shows**  $A \times_{\bullet} B \in_{\circ} \text{Vset } \alpha$

*<proof>*

Auxiliary results.

**lemma** *vempty-in-Vset-succ[simp, intro]*:  $0 \in_{\circ} \text{Vfrom } a \ (\text{succ } b)$

*<proof>*

**lemma** *Limit-vid-on-in-Vset*:

**assumes** *Limit*  $\alpha$  **and**  $A \in_{\circ} \text{Vset } \alpha$

**shows**  $\text{vid-on } A \in_{\circ} \text{Vset } \alpha$

*<proof>*

**lemma** *Ord-vpair-in-Vset-succI[intro]*:

**assumes** *Ord*  $\alpha$  **and**  $a \in_{\circ} \text{Vset } \alpha$  **and**  $b \in_{\circ} \text{Vset } \alpha$

**shows**  $\langle a, b \rangle \in_{\circ} \text{Vset } (\text{succ } (\text{succ } \alpha))$

*<proof>*

**lemma** *Limit-vifunion-vsubset-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $\bigwedge i. i \in_{\circ} I \implies A \ i \in_{\circ} \text{Vset } \alpha$

**shows**  $(\bigcup_{i \in_{\circ} I}. A \ i) \subseteq_{\circ} \text{Vset } \alpha$

*<proof>*

**lemma** *Limit-vproduct-vsubset-Vset-succI*:

**assumes** *Limit*  $\alpha$  **and**  $I \in_{\circ} \text{Vset } \alpha$  **and**  $\bigwedge i. i \in_{\circ} I \implies A \ i \subseteq_{\circ} \text{Vset } \alpha$

**shows**  $(\prod_{i \in_{\circ} I}. A \ i) \subseteq_{\circ} \text{Vset } (\text{succ } \alpha)$

*<proof>*

**lemma** *Limit-vproduct-vsubset-Vset-succI'*:

**assumes** *Limit*  $\alpha$  **and**  $I \in_{\circ} \text{Vset } \alpha$  **and**  $\bigwedge i. i \in_{\circ} I \implies A \ i \in_{\circ} \text{Vset } \alpha$

**shows**  $(\prod_{i \in_{\circ} I}. A \ i) \subseteq_{\circ} \text{Vset } (\text{succ } \alpha)$

*<proof>*

**lemma** (in *vfsequence*) *vfsequence-Ord-vcons-in-Vset-succI*:

**assumes** *Ord*  $\alpha$

**and**  $\omega \in_{\circ} \alpha$

**and**  $x \in_{\circ} \text{Vset } \alpha$

**and**  $xs \in_{\circ} \text{Vset } (\text{succ } (\text{succ } (\text{succ } \alpha)))$

**shows**  $\text{vcons } xs \ x \in_{\circ} \text{Vset } (\text{succ } (\text{succ } (\text{succ } \alpha)))$

*<proof>*

**lemma** *Limit-VUnion-vdomain-in-VsetI*:

**assumes** *Limit*  $\alpha$  **and**  $Q \in_{\circ} \text{Vset } \alpha$

**shows**  $(\bigcup_{r \in_{\circ} Q}. \mathcal{D}_{\circ} \ r) \in_{\circ} \text{Vset } \alpha$

*<proof>*

**lemma** *Limit-VUnion-vrange-in-VsetI:*

**assumes** *Limit*  $\alpha$  **and**  $Q \in_0 Vset \alpha$

**shows**  $(\bigcup_0 r \in_0 Q. \mathcal{R}_0 r) \in_0 Vset \alpha$

*<proof>*

#### 2.11.4 Axioms for $Vset \alpha$

The subsection demonstrates that the axioms of ZFC except for the Axiom Schema of Replacement hold in  $Vset \alpha$  for any limit ordinal  $\alpha$  such that  $\omega \in_0 \alpha$ <sup>1</sup>.

**locale**  $\mathcal{Z} =$

**fixes**  $\alpha$

**assumes** *Limit- $\alpha$ [intro, simp]:* *Limit*  $\alpha$

**and** *omega-in- $\alpha$ [intro, simp]:*  $\omega \in_0 \alpha$

**begin**

**lemmas** *[intro] =  $\mathcal{Z}$ -axioms*

**lemma** *vempty-Z-def:*  $0 = set \{x. x \neq x\}$  *<proof>*

**lemma** *vempty-is-zet[intro, simp]:*  $0 \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Extensionality:*

**assumes**  $a \in_0 Vset \alpha$  **and**  $x = y$  **and**  $x \in_0 a$

**shows**  $y \in_0 a$  **and**  $x \in_0 Vset \alpha$  **and**  $y \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Pairing:*

**assumes**  $a \in_0 Vset \alpha$  **and**  $b \in_0 Vset \alpha$

**shows**  $set \{a, b\} \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Unions:*

**assumes**  $a \in_0 Vset \alpha$

**shows**  $\bigcup_0 a \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Powers:*

**assumes**  $a \in_0 Vset \alpha$

**shows**  $VPow a \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Regularity:*

**assumes**  $a \neq 0$  **and**  $a \in_0 Vset \alpha$

**obtains**  $x$  **where**  $x \in_0 a$  **and**  $x \cap_0 a = 0$

*<proof>*

**lemma** *Axiom-of-Infinity:*  $\omega \in_0 Vset \alpha$

*<proof>*

**lemma** *Axiom-of-Choice:*

**assumes**  $A \in_0 Vset \alpha$

**obtains**  $f$  **where**  $f \in_0 Vset \alpha$  **and**  $\bigwedge x. x \in_0 A \implies x \neq 0 \implies f(\{x\}) \in_0 x$

*<proof>*

---

<sup>1</sup>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- $\alpha$* : *Ord*  $\alpha$  *<proof>*

**lemma** (in  $\mathcal{Z}$ ) *Z-Vset- $\omega$ 2-vsubset-Vset*: *Vset*  $(\omega + \omega) \subseteq_o$  *Vset*  $\alpha$  *<proof>*

**lemma** (in  $\mathcal{Z}$ ) *Z-Limit- $\alpha\omega$* : *Limit*  $(\alpha + \omega)$  *<proof>*

**lemma** (in  $\mathcal{Z}$ ) *Z- $\alpha$ - $\alpha\omega$* :  $\alpha \in_o$   $\alpha + \omega$  *<proof>*

**lemma** (in  $\mathcal{Z}$ ) *Z- $\omega$ - $\alpha\omega$* :  $\omega \in_o$   $\alpha + \omega$  *<proof>*

**lemma** *Z- $\omega\omega$* : *Z*  $(\omega + \omega)$  *<proof>*

**lemma** (in  $\mathcal{Z}$ ) *in-omega-in-omega-plus[intro]*:  
**assumes**  $a \in_o \omega$   
**shows**  $a \in_o$  *Vset*  $(\alpha + \omega)$   
*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *ord-of-nat-in-Vset[simp]*:  $a_{\mathbb{N}} \in_o$  *Vset*  $\alpha$  *<proof>*  
*vfsequences-on.*

**lemma** (in  $\mathcal{Z}$ ) *vfsequences-on-in-VsetI*:  
**assumes**  $X \in_o$  *Vset*  $\alpha$   
**shows** *vfsequences-on*  $X \in_o$  *Vset*  $\alpha$   
*<proof>*

### 2.11.5 Existence of a disjoint subset in *Vset* $\alpha$

**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*  $(\text{mk-doubleton } Y \text{ `elts } X)$

**lemma** *inj-on-mk-doubleton*: *inj-on*  $(\text{mk-doubleton } X)$   $(\text{elts } X)$   
*<proof>*

**lemma** *mk-doubleton-image-vsubset-veqpoll*:  
**assumes**  $X \subseteq_o Y$   
**shows** *mk-doubleton-image*  $X X \approx_o$  *mk-doubleton-image*  $X Y$   
*<proof>*

**lemma** *mk-doubleton-image-veqpoll*:  
**assumes**  $X \subseteq_o Y$   
**shows**  $X \approx_o$  *mk-doubleton-image*  $X Y$   
*<proof>*

**lemma** *vdisjnt-mk-doubleton-image*: *vdisjnt*  $(\text{mk-doubleton-image } X Y)$   $Y$   
*<proof>*

**lemma** *Limit-mk-doubleton-image-vsubset-Vset*:  
**assumes** *Limit*  $\alpha$  **and**  $X \subseteq_o Y$  **and**  $Y \in_o$  *Vset*  $\alpha$

**shows** *mk-doubleton-image*  $X Y \subseteq_{\circ} Vset \alpha$   
 ⟨*proof*⟩

**lemma** *Ord-mk-doubleton-image-vsubset-Vset-succ*:  
**assumes** *Ord*  $\alpha$  **and**  $X \subseteq_{\circ} Y$  **and**  $Y \in_{\circ} Vset \alpha$   
**shows** *mk-doubleton-image*  $X Y \subseteq_{\circ} Vset (succ \alpha)$   
 ⟨*proof*⟩

**lemma** *Limit-ex-epoll-vdisjnt*:  
**assumes** *Limit*  $\alpha$  **and**  $X \subseteq_{\circ} Y$  **and**  $Y \in_{\circ} Vset \alpha$   
**obtains**  $Z$  **where**  $X \approx_{\circ} Z$  **and** *vdisjnt*  $Z Y$  **and**  $Z \subseteq_{\circ} Vset \alpha$   
 ⟨*proof*⟩

**lemma** *Ord-ex-epoll-vdisjnt*:  
**assumes** *Ord*  $\alpha$  **and**  $X \subseteq_{\circ} Y$  **and**  $Y \in_{\circ} Vset \alpha$   
**obtains**  $Z$  **where**  $X \approx_{\circ} Z$  **and** *vdisjnt*  $Z Y$  **and**  $Z \subseteq_{\circ} Vset (succ \alpha)$   
 ⟨*proof*⟩

## 2.12 $n$ -ary operation

### 2.12.1 Partial $n$ -ary operation

**locale**  $pnop = vsv f$  **for**  $A n f :: V +$   
**assumes**  $pnop-n: n \in_o \omega$   
**and**  $pnop-vdomain: \mathcal{D}_o f \subseteq_o A \hat{\times}_x n$   
**and**  $pnop-vrange: \mathcal{R}_o f \subseteq_o A$

Rules.

**lemma**  $pnopI[intro]:$   
**assumes**  $vsv f$   
**and**  $n \in_o \omega$   
**and**  $\mathcal{D}_o f \subseteq_o A \hat{\times}_x n$   
**and**  $\mathcal{R}_o f \subseteq_o A$   
**shows**  $pnop A n f$   
 $\langle proof \rangle$

**lemma**  $pnopD[dest]:$   
**assumes**  $pnop A n f$   
**shows**  $vsv f$   
**and**  $n \in_o \omega$   
**and**  $\mathcal{D}_o f \subseteq_o A \hat{\times}_x n$   
**and**  $\mathcal{R}_o f \subseteq_o A$   
 $\langle proof \rangle$

**lemma**  $pnopE[elim]:$   
**assumes**  $pnop A n f$   
**obtains**  $vsv f$   
**and**  $n \in_o \omega$   
**and**  $\mathcal{D}_o f \subseteq_o A \hat{\times}_x n$   
**and**  $\mathcal{R}_o f \subseteq_o A$   
 $\langle proof \rangle$

### 2.12.2 Total $n$ -ary operation

**locale**  $nop = vsv f$  **for**  $A n f :: V +$   
**assumes**  $nop-n: n \in_o \omega$   
**and**  $nop-vdomain: \mathcal{D}_o f = A \hat{\times}_x n$   
**and**  $nop-vrange: \mathcal{R}_o f \subseteq_o A$

**sublocale**  $nop \subseteq pnop A n f$   
 $\langle proof \rangle$

Rules.

**lemma**  $nopI[intro]:$   
**assumes**  $vsv f$   
**and**  $n \in_o \omega$   
**and**  $\mathcal{D}_o f = A \hat{\times}_x n$   
**and**  $\mathcal{R}_o f \subseteq_o A$   
**shows**  $nop A n f$   
 $\langle proof \rangle$

**lemma**  $nopD[dest]:$   
**assumes**  $nop A n f$   
**shows**  $vsv f$   
**and**  $n \in_o \omega$   
**and**  $\mathcal{D}_o f = A \hat{\times}_x n$

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  $\mathcal{D}_\circ f = A \hat{\times} n$   
 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 ∈<sub>∘</sub> ω*  
 and *nop-v11-vdomain:  $\mathcal{D}_\circ f = A \hat{\times} n$*   
 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  $\mathcal{D}_\circ f = A \hat{\times} n$   
 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  $\mathcal{D}_\circ f = A \hat{\times} n$   
 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  $\mathcal{D}_\circ f = A \hat{\times} n$   
 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 ∈<sub>∘</sub> ω*  
 and *nop-onto-vdomain:  $\mathcal{D}_\circ f = A \hat{\times} n$*   
 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**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $\mathcal{R}_{\circ} f = A$   
**shows**  $nop\text{-}onto\ A\ n\ f$   
 $\langle proof \rangle$

**lemma**  $nop\text{-}ontoD[dest]$ :  
**assumes**  $nop\text{-}onto\ A\ n\ f$   
**shows**  $vsv\ f$   
**and**  $n \in_{\circ} \omega$   
**and**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $\mathcal{R}_{\circ} f = A$   
 $\langle proof \rangle$

**lemma**  $nop\text{-}ontoE[elim]$ :  
**assumes**  $nop\text{-}onto\ A\ n\ f$   
**obtains**  $vsv\ f$   
**and**  $n \in_{\circ} \omega$   
**and**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $\mathcal{R}_{\circ} f = A$   
 $\langle proof \rangle$

### 2.12.5 Bijective $n$ -ary operation

**locale**  $nop\text{-}bij = v11\ f\ \text{for}\ A\ n\ f :: V +$   
**assumes**  $nop\text{-}bij\text{-}n: n \in_{\circ} \omega$   
**and**  $nop\text{-}bij\text{-}vdomain: \mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $nop\text{-}bij\text{-}vrange: \mathcal{R}_{\circ} f = A$

**sublocale**  $nop\text{-}bij \subseteq nop\text{-}v11$   
 $\langle proof \rangle$

**sublocale**  $nop\text{-}bij \subseteq nop\text{-}onto$   
 $\langle proof \rangle$

Rules.

**lemma**  $nop\text{-}bijI[intro]$ :  
**assumes**  $v11\ f$   
**and**  $n \in_{\circ} \omega$   
**and**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $\mathcal{R}_{\circ} f = A$   
**shows**  $nop\text{-}bij\ A\ n\ f$   
 $\langle proof \rangle$

**lemma**  $nop\text{-}bijD[dest]$ :  
**assumes**  $nop\text{-}bij\ A\ n\ f$   
**shows**  $v11\ f$   
**and**  $n \in_{\circ} \omega$   
**and**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$   
**and**  $\mathcal{R}_{\circ} f = A$   
 $\langle proof \rangle$

**lemma**  $nop\text{-}bijE[elim]$ :  
**assumes**  $nop\text{-}bij\ A\ n\ f$   
**obtains**  $v11\ f$   
**and**  $n \in_{\circ} \omega$   
**and**  $\mathcal{D}_{\circ} f = A \hat{\ }_x n$

and  $\mathcal{R}_o f = A$   
 ⟨proof⟩

### 2.12.6 Scalar

**locale** *scalar* =  
**fixes**  $A f$   
**assumes** *scalar-nop*:  $\text{nop } A \ 0 \ f$

**sublocale** *scalar*  $\subseteq$  *nop*  $A \ 0 \ f$   
**rewrites** *scalar-vdomain*[*simp*]:  $A \hat{\times} 0 = \text{set } \{0\}$   
 ⟨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*  $A \ \langle 1_{\mathbf{N}} \rangle \ f$  **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-v11* = *nop-v11*  $A \ \langle 1_{\mathbf{N}} \rangle \ f$  **for**  $A \ f$

**sublocale** *unop-v11*  $\subseteq$  *unop*  $A \ f$  ⟨proof⟩

Rules.

**lemma** *unop-v11I*[*intro*]:  
**assumes** *nop-v11*  $A \ (1_{\mathbf{N}}) \ f$   
**shows** *unop-v11*  $A \ f$   
 ⟨proof⟩

**lemma** *unop-v11D*[*dest*]:  
**assumes** *unop-v11*  $A \ f$   
**shows** *nop-v11*  $A \ (1_{\mathbf{N}}) \ f$

*<proof>*

**lemma** *unop-v11E[elim]*:  
**assumes** *unop-v11 A f*  
**obtains** *nop-v11 A (1<sub>N</sub>) f*  
*<proof>*

### 2.12.9 Surjective unary operation

**locale** *unop-onto = nop-onto A <1<sub>N</sub>> f for A f*

**sublocale** *unop-onto ⊆ unop A f <proof>*

Rules.

**lemma** *unop-ontoI[intro]*:  
**assumes** *nop-onto A (1<sub>N</sub>) f*  
**shows** *unop-onto A f*  
*<proof>*

**lemma** *unop-ontoD[dest]*:  
**assumes** *unop-onto A f*  
**shows** *nop-onto A (1<sub>N</sub>) f*  
*<proof>*

**lemma** *unop-ontoE[elim]*:  
**assumes** *unop-onto A f*  
**obtains** *nop-onto A (1<sub>N</sub>) f*  
*<proof>*

**lemma** *unop-ontoI'[intro]*:  
**assumes** *unop A f and A ⊆<sub>o</sub> R<sub>o</sub> f*  
**shows** *unop-onto A f*  
*<proof>*

### 2.12.10 Bijective unary operation

**locale** *unop-bij = nop-bij A <1<sub>N</sub>> f for A f*

**sublocale** *unop-bij ⊆ unop-v11 A f*  
*<proof>*

**sublocale** *unop-bij ⊆ unop-onto A f*  
*<proof>*

Rules.

**lemma** *unop-bijI[intro]*:  
**assumes** *nop-bij A (1<sub>N</sub>) f*  
**shows** *unop-bij A f*  
*<proof>*

**lemma** *unop-bijD[dest]*:  
**assumes** *unop-bij A f*  
**shows** *nop-bij A (1<sub>N</sub>) f*  
*<proof>*

**lemma** *unop-bijE[elim]*:  
**assumes** *unop-bij A f*  
**obtains** *nop-bij A (1<sub>N</sub>) f*

*<proof>*

**lemma** *unop-bijI*[*intro*]:  
**assumes** *unop-v11*  $A f$  **and**  $A \subseteq_o \mathcal{R}_o f$   
**shows** *unop-bij*  $A f$   
*<proof>*

### 2.12.11 Partial binary operation

**locale** *pbinop* = *pnop*  $A \langle 2_{\mathbf{N}} \rangle f$  **for**  $A f$

**sublocale** *pbinop*  $\subseteq$  *dom: fbrelation*  $\langle \mathcal{D}_o f \rangle$   
*<proof>*

Rules.

**lemmas** *pbinopI*[*intro*] = *pbinop.intro*

**lemma** *pbinopD*[*dest*]:  
**assumes** *pbinop*  $A f$   
**shows** *pnop*  $A \langle 2_{\mathbf{N}} \rangle f$   
*<proof>*

**lemma** *pbinopE*[*elim*]:  
**assumes** *pbinop*  $A f$   
**obtains** *pnop*  $A \langle 2_{\mathbf{N}} \rangle f$   
*<proof>*

Elementary properties.

**lemma** (**in** *pbinop*) *fbino-vcard*:  
**assumes**  $x \in_o \mathcal{D}_o f$   
**shows** *vcard*  $x = 2_{\mathbf{N}}$   
*<proof>*

### 2.12.12 Total binary operation

**locale** *binop* = *nop*  $A \langle 2_{\mathbf{N}} \rangle f$  **for**  $A f$

**sublocale** *binop*  $\subseteq$  *pbinop* *<proof>*

Rules.

**lemmas** *binopI*[*intro*] = *binop.intro*

**lemma** *binopD*[*dest*]:  
**assumes** *binop*  $A f$   
**shows** *nop*  $A \langle 2_{\mathbf{N}} \rangle f$   
*<proof>*

**lemma** *binopE*[*elim*]:  
**assumes** *binop*  $A f$   
**obtains** *nop*  $A \langle 2_{\mathbf{N}} \rangle f$   
*<proof>*

Elementary properties.

**lemma** *binop-eqI*:  
**assumes** *binop*  $A g$   
**and** *binop*  $A f$   
**and**  $\bigwedge a b. [\![ a \in_o A; b \in_o A ]\!] \implies g(a, b)_{\bullet} = f(a, b)_{\bullet}$

**shows**  $g = f$   
 ⟨*proof*⟩

**lemma** (in *binop*) *binop-app-in-vrange*[*intro*]:  
**assumes**  $a \in_0 A$  **and**  $b \in_0 A$   
**shows**  $f(a, b) \bullet \in_0 \mathcal{R}_0 f$   
 ⟨*proof*⟩

### 2.12.13 Injective binary operation

**locale** *binop-v11* = *nop-v11*  $A \langle 2_{\mathbb{N}} \rangle f$  **for**  $A f$

**sublocale** *binop-v11*  $\subseteq$  *binop*  $A f$  ⟨*proof*⟩

Rules.

**lemma** *binop-v11I*[*intro*]:  
**assumes** *nop-v11*  $A \langle 2_{\mathbb{N}} \rangle f$   
**shows** *binop-v11*  $A f$   
 ⟨*proof*⟩

**lemma** *binop-v11D*[*dest*]:  
**assumes** *binop-v11*  $A f$   
**shows** *nop-v11*  $A \langle 2_{\mathbb{N}} \rangle f$   
 ⟨*proof*⟩

**lemma** *binop-v11E*[*elim*]:  
**assumes** *binop-v11*  $A f$   
**obtains** *nop-v11*  $A \langle 2_{\mathbb{N}} \rangle f$   
 ⟨*proof*⟩

### 2.12.14 Surjective binary operation

**locale** *binop-onto* = *nop-onto*  $A \langle 2_{\mathbb{N}} \rangle f$  **for**  $A f$

**sublocale** *binop-onto*  $\subseteq$  *binop*  $A f$  ⟨*proof*⟩

Rules.

**lemma** *binop-ontoI*[*intro*]:  
**assumes** *nop-onto*  $A \langle 2_{\mathbb{N}} \rangle f$   
**shows** *binop-onto*  $A f$   
 ⟨*proof*⟩

**lemma** *binop-ontoD*[*dest*]:  
**assumes** *binop-onto*  $A f$   
**shows** *nop-onto*  $A \langle 2_{\mathbb{N}} \rangle f$   
 ⟨*proof*⟩

**lemma** *binop-ontoE*[*elim*]:  
**assumes** *binop-onto*  $A f$   
**obtains** *nop-onto*  $A \langle 2_{\mathbb{N}} \rangle f$   
 ⟨*proof*⟩

**lemma** *binop-ontoI'*[*intro*]:  
**assumes** *binop*  $A f$  **and**  $A \subseteq_0 \mathcal{R}_0 f$   
**shows** *binop-onto*  $A f$   
 ⟨*proof*⟩

**2.12.15 Bijective binary operation**

**locale**  $\text{binop-bij} = \text{nop-bij } A \langle 2_{\mathbb{N}} \rangle f$  **for**  $A f$

**sublocale**  $\text{binop-bij} \subseteq \text{binop-v11 } A f$   
 $\langle \text{proof} \rangle$

**sublocale**  $\text{binop-bij} \subseteq \text{binop-onto } A f$   
 $\langle \text{proof} \rangle$

Rules.

**lemma**  $\text{binop-bijI}[\text{intro}]$ :  
**assumes**  $\text{nop-bij } A (2_{\mathbb{N}}) f$   
**shows**  $\text{binop-bij } A f$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{binop-bijD}[\text{dest}]$ :  
**assumes**  $\text{binop-bij } A f$   
**shows**  $\text{nop-bij } A (2_{\mathbb{N}}) f$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{binop-bijE}[\text{elim}]$ :  
**assumes**  $\text{binop-bij } A f$   
**obtains**  $\text{nop-bij } A (2_{\mathbb{N}}) f$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{binop-bijI}'[\text{intro}]$ :  
**assumes**  $\text{binop-v11 } A f$  **and**  $A \subseteq_{\circ} \mathcal{R}_{\circ} f$   
**shows**  $\text{binop-bij } A f$   
 $\langle \text{proof} \rangle$

**2.12.16 Flip**

**definition**  $\text{fflip} :: V \Rightarrow V$   
**where**  $\text{fflip } f = (\lambda ab \in_{\circ} (\mathcal{D}_{\circ} f)^{-1} \bullet. f(\langle ab \langle 1_{\mathbb{N}} \rangle, ab \langle 0 \rangle \rangle) \bullet)$

Elementary properties.

**lemma**  $\text{fflip-vempty}[\text{simp}]$ :  $\text{fflip } 0 = 0$   $\langle \text{proof} \rangle$

**lemma**  $\text{fflip-vsuv}$ :  $\text{vsu } (\text{fflip } f)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vdomain-fflip}[\text{simp}]$ :  $\mathcal{D}_{\circ} (\text{fflip } f) = (\mathcal{D}_{\circ} f)^{-1} \bullet$   
 $\langle \text{proof} \rangle$

**lemma** (**in**  $\text{pbinop}$ )  $\text{vrange-fflip}$ :  $\mathcal{R}_{\circ} (\text{fflip } f) = \mathcal{R}_{\circ} f$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{fflip-app}[\text{simp}]$ :  
**assumes**  $[a, b]_{\circ} \in_{\circ} \mathcal{D}_{\circ} f$   
**shows**  $\text{fflip } f(\langle b, a \rangle \bullet) = f(\langle a, b \rangle \bullet)$   
 $\langle \text{proof} \rangle$

**lemma** (**in**  $\text{pbinop}$ )  $\text{pbinop-fflip-fflip}$ :  $\text{fflip } (\text{fflip } f) = f$   
 $\langle \text{proof} \rangle$

**lemma** (**in**  $\text{binop}$ )  $\text{pbinop-fflip-app}[\text{simp}]$ :  
**assumes**  $a \in_{\circ} A$  **and**  $b \in_{\circ} A$

**shows**  $\text{flip } f(b, a) \bullet = f(a, b) \bullet$   
(proof)

**lemma** *flip-vs-singleton*:  $\text{flip } (\text{set } \{ \langle [a, b]_{\circ}, c \rangle \}) = \text{set } \{ \langle [b, a]_{\circ}, c \rangle \}$   
(proof)

## 2.13 Construction of integer numbers, rational numbers and real numbers

### 2.13.1 Background

The set of real numbers  $\mathbf{R}_\circ$  is defined in a way such that it agrees with the set of natural numbers  $\omega$ . However, otherwise, real numbers are allowed to be arbitrary sets in  $Vset(\omega + \omega)$ .<sup>2</sup> 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**  $set\ x \lesssim_\circ VPow\ \omega$   
 ⟨*proof*⟩

### 2.13.2 Real numbers

#### Definition

**abbreviation** *real* :: *nat*  $\Rightarrow$  *real*  
**where** *real*  $\equiv$  *of-nat*

**definition** *nat-of-real* :: *real*  $\Rightarrow$  *nat*  
**where** *nat-of-real* = *inv-into UNIV real*

**definition** *vreal-of-real-impl* :: *real*  $\Rightarrow$  *V*  
**where** *vreal-of-real-impl* = (*SOME V-of::real* $\Rightarrow$ *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*:  
 $set\ (range\ vreal-of-real-impl) \lesssim_\circ VPow\ \omega$   
 ⟨*proof*⟩

**definition** *vreal-impl* :: *V*

---

<sup>2</sup>The idea itself is not new, e.g., see [26].

**where** *vreal-impl* =  
 (  
   *SOME* *y*.  
   *range vreal-of-real-impl*  $\approx$  *elts y*  $\wedge$   
   *vdisjnt y*  $\omega$   $\wedge$   
   *y*  $\in_{\circ}$  *Vset* ( $\omega + \omega$ )  
 )

**lemma** *vreal-impl-eppoll*: *range vreal-of-real-impl*  $\approx$  *elts vreal-impl*  
**and** *vreal-impl-vdisjnt*: *vdisjnt vreal-impl*  $\omega$   
**and** *vreal-impl-in-Vset-ss-omega*: *vreal-impl*  $\in_{\circ}$  *Vset* ( $\omega + \omega$ )  
 ⟨*proof*⟩

**definition** *vreal-of-real-impl'* ::  $V \Rightarrow V$   
**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''* ::  $real \Rightarrow V$   
**where** *vreal-of-real-impl''* = *vreal-of-real-impl'*  $\circ$  *vreal-of-real-impl*

**lemma** *vreal-of-real-impl''-disjnt*: *disjnt* (*range vreal-of-real-impl''*) (*elts*  $\omega$ )  
 ⟨*proof*⟩

**lemma** *inj-vreal-of-real-impl''*: *inj vreal-of-real-impl''*  
 ⟨*proof*⟩

Main definitions.

**definition** *vreal-of-real* ::  $real \Rightarrow V$   
**where** *vreal-of-real* *x* =  
 (*if* *x*  $\in$   $\mathbb{N}$  *then* (*nat-of-real* *x*) $_{\mathbb{N}}$  *else* *vreal-of-real-impl''* *x*)

**notation** *vreal-of-real* ( $\langle \_ \rangle_{\mathbb{R}}$ ) [1000] 999)

**declare** [[*coercion vreal-of-real* ::  $real \Rightarrow V$ ]]

**definition** *vreal* ::  $V (\mathbb{R}_{\circ})$   
**where** *vreal* = *set* (*range vreal-of-real*)

**definition** *real-of-vreal* ::  $V \Rightarrow real$   
**where** *real-of-vreal* = *inv-into UNIV vreal-of-real*

Rules.

**lemma** *vreal-of-real-in-vrealI*[*intro*, *simp*]:  $a_{\mathbb{R}} \in_{\circ} \mathbb{R}_{\circ}$   
 ⟨*proof*⟩

**lemma** *vreal-of-real-in-vrealE*[*elim*]:  
**assumes**  $a \in_{\circ} \mathbb{R}_{\circ}$   
**obtains** *b* **where**  $b_{\mathbb{R}} = a$   
 ⟨*proof*⟩

Elementary properties.

**lemma** *vnat-eq-vreal*:  $x_{\mathbb{N}} = x_{\mathbb{R}}$  ⟨*proof*⟩

**lemma** *omega-vsubset-vreal*:  $\omega \subseteq_o \mathbb{R}_o$

*<proof>*

**lemma** *inj-vreal-of-real*: *inj vreal-of-real*

*<proof>*

**lemma** *vreal-in-Vset- $\omega$ 2*:  $\mathbb{R}_o \in_o Vset (\omega + \omega)$

*<proof>*

**lemma** *real-of-vreal-vreal-of-real[simp]*: *real-of-vreal* ( $a_{\mathbb{R}}$ ) =  $a$

*<proof>*

## Transfer rules

**definition** *cr-vreal* ::  $V \Rightarrow real \Rightarrow bool$

**where** *cr-vreal*  $a b \leftrightarrow (a = vreal-of-real b)$

**lemma** *cr-vreal-right-total[transfer-rule]*: *right-total cr-vreal*

*<proof>*

**lemma** *cr-vreal-bi-unique[transfer-rule]*: *bi-unique cr-vreal*

*<proof>*

**lemma** *cr-vreal-transfer-domain-rule[transfer-domain-rule]*:

*Domainp cr-vreal = ( $\lambda x. x \in_o \mathbb{R}_o$ )*

*<proof>*

**lemma** *vreal-transfer[transfer-rule]*:

*(rel-set cr-vreal) (elts  $\mathbb{R}_o$ ) (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*:  $[a_{\mathbb{R}}]_o \in_o \mathbb{R}_o \hat{\times} 1_{\mathbb{N}}$  *<proof>*

**lemma** *vreal-fpair-in-fproduct-vreal*:  $[a_{\mathbb{R}}, b_{\mathbb{R}}]_o \in_o \mathbb{R}_o \hat{\times} 2_{\mathbb{N}}$  *<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* =

$(\lambda x \in_o \mathbb{R}_o \hat{\times} 2_{\mathbb{N}}. (real-of-vreal (x(0_{\mathbb{N}})) + real-of-vreal (x(1_{\mathbb{N}}))))_{\mathbb{R}}$

**abbreviation** *vreal-plus-app* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle +_{\mathbb{R}} \rangle$  65)

**where** *vreal-plus-app*  $a b \equiv vreal-plus(a, b)$ .

**notation** *vreal-plus-app* (**infixl**  $\langle +_{\mathbb{R}} \rangle$  65)

**lemma** *vreal-plus-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*cr-vreal*  $\implies$  *cr-vreal*  $\implies$  *cr-vreal*)  
 $(+_{\mathbf{R}}) (+)$   
 $\langle \textit{proof} \rangle$

Multiplication.

**definition** *vreal-mult* ::  $V$   
**where** *vreal-mult* =  
 $(\lambda x \in_{\circ} \mathbf{R}_{\circ} \hat{\times} 2_{\mathbf{N}}. (\textit{real-of-vreal} (x(0_{\mathbf{N}})) * \textit{real-of-vreal} (x(1_{\mathbf{N}}))))_{\mathbf{R}}$

**abbreviation** *vreal-mult-app* (**infixl**  $\langle *_{\mathbf{R}} \rangle$  70)  
**where** *vreal-mult-app*  $a \ b \equiv \textit{vreal-mult} (a, b)$ .  
**notation** *vreal-mult-app* (**infixl**  $\langle *_{\mathbf{R}} \rangle$  70)

**lemma** *vreal-mult-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*cr-vreal*  $\implies$  *cr-vreal*  $\implies$  *cr-vreal*)  $(*_{\mathbf{R}}) (*)$   
 $\langle \textit{proof} \rangle$

Unary minus.

**definition** *vreal-uminus* ::  $V$   
**where** *vreal-uminus* =  $(\lambda x \in_{\circ} \mathbf{R}_{\circ}. (\textit{uminus} (\textit{real-of-vreal} x)))_{\mathbf{R}}$

**abbreviation** *vreal-uminus-app* ( $\langle -_{\mathbf{R}} \rightarrow$  [81] 80)  
**where**  $-_{\mathbf{R}} a \equiv \textit{vreal-uminus} (a)$

**lemma** *vreal-uminus-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*cr-vreal*  $\implies$  *cr-vreal*) (*vreal-uminus-app*) (*uminus*)  
 $\langle \textit{proof} \rangle$

Multiplicative inverse.

**definition** *vreal-inverse* ::  $V$   
**where** *vreal-inverse* =  $(\lambda x \in_{\circ} \mathbf{R}_{\circ}. (\textit{inverse} (\textit{real-of-vreal} x)))_{\mathbf{R}}$

**abbreviation** *vreal-inverse-app* ( $\langle (-^{-1}_{\mathbf{R}}) \rangle$  [1000] 999)  
**where**  $a^{-1}_{\mathbf{R}} \equiv \textit{vreal-inverse} (a)$

**lemma** *vreal-inverse-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** (*cr-vreal*  $\implies$  *cr-vreal*) (*vreal-inverse-app*) (*inverse*)  
 $\langle \textit{proof} \rangle$

Order.

**definition** *vreal-le* ::  $V$   
**where** *vreal-le* =  
 $\textit{set} \{ [a, b]_{\circ} \mid a \ b. [a, b]_{\circ} \in_{\circ} \mathbf{R}_{\circ} \hat{\times} 2_{\mathbf{N}} \wedge \textit{real-of-vreal} a \leq \textit{real-of-vreal} b \}$

**abbreviation** *vreal-le'* ( $\langle (- / \leq_{\mathbf{R}} -) \rangle$  [51, 51] 50)  
**where**  $a \leq_{\mathbf{R}} b \equiv [a, b]_{\circ} \in_{\circ} \textit{vreal-le}$

**lemma** *small-vreal-le*[*simp*]:  
 $\textit{small}$   
 $\{ [a, b]_{\circ} \mid a \ b. [a, b]_{\circ} \in_{\circ} \mathbf{R}_{\circ} \hat{\times} 2_{\mathbf{N}} \wedge \textit{real-of-vreal} a \leq \textit{real-of-vreal} b \}$   
 $\langle \textit{proof} \rangle$

**lemma** *vreal-le-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vreal \implies cr\text{-}vreal \implies (=)$ ) *vreal-le'* ( $\leq$ )  
*<proof>*

Strict order.

**definition** *vreal-ls* ::  $V$   
**where** *vreal-ls* =  
 $set \{[a, b]_o \mid a \ b. [a, b]_o \in_o \mathbb{R}_o \hat{\times} 2_{\mathbb{N}} \wedge real\text{-of}\text{-}vreal \ a < real\text{-of}\text{-}vreal \ b\}$

**abbreviation** *vreal-ls'* ( $\langle(-/ <_{\mathbb{R}} -)\rangle$  [51, 51] 50)  
**where**  $a <_{\mathbb{R}} b \equiv [a, b]_o \in_o vreal\text{-}ls$

**lemma** *small-vreal-ls*[*simp*]:  
*small*  
 $\{[a, b]_o \mid a \ b. [a, b]_o \in_o \mathbb{R}_o \hat{\times} 2_{\mathbb{N}} \wedge real\text{-of}\text{-}vreal \ a < real\text{-of}\text{-}vreal \ b\}$   
*<proof>*

**lemma** *vreal-ls-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vreal \implies cr\text{-}vreal \implies (=)$ ) *vreal-ls'* ( $<$ )  
*<proof>*

Subtraction.

**definition** *vreal-minus* ::  $V$   
**where** *vreal-minus* =  
 $(\lambda x \in_o \mathbb{R}_o. \hat{\times} 2_{\mathbb{N}}. (real\text{-of}\text{-}vreal \ (x(0_{\mathbb{N}})) - real\text{-of}\text{-}vreal \ (x(1_{\mathbb{N}}))))_{\mathbb{R}}$

**abbreviation** *vreal-minus-app* (**infixl**  $\langle-_{\mathbb{R}}\rangle$  65)  
**where**  $vreal\text{-}minus\text{-}app \ a \ b \equiv vreal\text{-}minus([a, b])_{\bullet}$

**lemma** *vreal-minus-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vreal \implies cr\text{-}vreal \implies cr\text{-}vreal$ ) ( $-_{\mathbb{R}}$ ) ( $-$ )  
*<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*:  $0_{\mathbb{R}} \in_o \mathbb{R}_o$   
*<proof>*

**lemma** *vreal-one-closed*:  $1_{\mathbb{R}} \in_o \mathbb{R}_o$   
*<proof>*

**lemma** *vreal-plus-closed*:  
**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$   
**shows**  $x +_{\mathbb{R}} y \in_o \mathbb{R}_o$   
*<proof>*

**lemma** *vreal-uminus-closed*:  
**assumes**  $x \in_o \mathbb{R}_o$   
**shows**  $-_{\mathbb{R}} x \in_o \mathbb{R}_o$   
*<proof>*

**lemma** *vreal-mult-closed*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$

**shows**  $x *_R y \in_o \mathbb{R}_o$

*<proof>*

**lemma** *vreal-inverse-closed*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x^{-1}_R \in_o \mathbb{R}_o$

*<proof>*

Associative Law for Addition: Definition 2.2.1.a.

**lemma** *vreal-assoc-law-addition*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$  **and**  $z \in_o \mathbb{R}_o$

**shows**  $(x +_R y) +_R z = x +_R (y +_R z)$

*<proof>*

Commutative Law for Addition: Definition 2.2.1.b.

**lemma** *vreal-commutative-law-addition*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$

**shows**  $x +_R y = y +_R x$

*<proof>*

Identity Law for Addition: Definition 2.2.1.c.

**lemma** *vreal-identity-law-addition*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x +_R 0_R = x$

*<proof>*

Inverses Law for Addition: Definition 2.2.1.d.

**lemma** *vreal-inverses-law-addition*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x +_R (-_R x) = 0_R$

*<proof>*

Associative Law for Multiplication: Definition 2.2.1.e.

**lemma** *vreal-assoc-law-multiplication*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$  **and**  $z \in_o \mathbb{R}_o$

**shows**  $(x *_R y) *_R z = x *_R (y *_R z)$

*<proof>*

Commutative Law for Multiplication: Definition 2.2.1.f.

**lemma** *vreal-commutative-law-multiplication*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$

**shows**  $x *_R y = y *_R x$

*<proof>*

Identity Law for Multiplication: Definition 2.2.1.g.

**lemma** *vreal-identity-law-multiplication*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x *_R 1_R = x$

*<proof>*

Inverses Law for Multiplication: Definition 2.2.1.h.

**lemma** *vreal-inverses-law-multiplication*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $x \neq 0_R$

**shows**  $x *_R x^{-1}_R = 1_R$

*<proof>*

Distributive Law: Definition 2.2.1.i.

**lemma** *vreal-distributive-law*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$  **and**  $z \in_o \mathbb{R}_o$

**shows**  $x *_R (y +_R z) = x *_R y +_R x *_R z$

*<proof>*

Trichotomy Law: Definition 2.2.1.j.

**lemma** *vreal-trichotomy-law*:

**assumes**  $x \in_o \mathbb{R}_o$   $y \in_o \mathbb{R}_o$

**shows**

$(x <_R y \wedge \sim(x = y) \wedge \sim(y <_R x)) \vee$

$(\sim(x <_R y) \wedge x = y \wedge \sim(y <_R x)) \vee$

$(\sim(x <_R y) \wedge \sim(x = y) \wedge y <_R x)$

*<proof>*

Transitive Law: Definition 2.2.1.k.

**lemma** *vreal-transitive-law*:

**assumes**  $x \in_o \mathbb{R}_o$

**and**  $y \in_o \mathbb{R}_o$

**and**  $z \in_o \mathbb{R}_o$

**and**  $x <_R y$  **and**  $y <_R z$

**shows**  $x <_R z$

*<proof>*

Addition Law of Order: Definition 2.2.1.l.

**lemma** *vreal-addition-law-of-order*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$  **and**  $z \in_o \mathbb{R}_o$  **and**  $x <_R y$

**shows**  $x +_R z <_R y +_R z$

*<proof>*

Multiplication Law of Order: Definition 2.2.1.m.

**lemma** *vreal-multiplication-law-of-order*:

**assumes**  $x \in_o \mathbb{R}_o$

**and**  $y \in_o \mathbb{R}_o$

**and**  $z \in_o \mathbb{R}_o$

**and**  $x <_R y$

**and**  $0_R <_R z$

**shows**  $x *_R z <_R y *_R z$

*<proof>*

Non-Triviality: Definition 2.2.1.n.

**lemma** *vreal-non-triviality*:  $0_R \neq 1_R$

*<proof>*

Least upper bound property: Definition 2.2.3.

**lemma** *least-upper-bound-property*:

**defines** *vreal-ub*  $S M \equiv (S \subseteq_o \mathbb{R}_o \wedge M \in_o \mathbb{R}_o \wedge (\forall x \in_o S. x \leq_R M))$

**assumes**  $A \subseteq_o \mathbb{R}_o$  **and**  $A \neq 0$  **and**  $\exists M. \textit{vreal-ub } A M$

**obtains**  $M$  **where** *vreal-ub*  $A M$  **and**  $\wedge T. \textit{vreal-ub } A T \implies M \leq_R T$

*<proof>*

## Fundamental properties of other operations

Minus.

**lemma** *vreal-minus-closed*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$

**shows**  $x -_{\mathbb{R}} y \in_o \mathbb{R}_o$

*<proof>*

**lemma** *vreal-minus-eq-plus-uminus*:

**assumes**  $x \in_o \mathbb{R}_o$  **and**  $y \in_o \mathbb{R}_o$

**shows**  $x -_{\mathbb{R}} y = x +_{\mathbb{R}} (-_{\mathbb{R}} y)$

*<proof>*

Unary minus.

**lemma** *vreal-uminus-uminus*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x = -_{\mathbb{R}} (-_{\mathbb{R}} x)$

*<proof>*

Multiplicative inverse.

**lemma** *vreal-inverse-inverse*:

**assumes**  $x \in_o \mathbb{R}_o$

**shows**  $x = (x^{-1}_{\mathbb{R}})^{-1}_{\mathbb{R}}$

*<proof>*

## Further properties

Addition.

**global-interpretation** *vreal-plus: binop-onto*  $\langle \mathbb{R}_o \rangle$  *vreal-plus*

*<proof>*

Unary minus.

**global-interpretation** *vreal-uminus: v11 vreal-uminus*

**rewrites** *vreal-uminus-vdomain*[*simp*]:  $\mathcal{D}_o$  *vreal-uminus* =  $\mathbb{R}_o$

**and** *vreal-uminus-vrange*[*simp*]:  $\mathcal{R}_o$  *vreal-uminus* =  $\mathbb{R}_o$

*<proof>*

Multiplication.

**global-interpretation** *vreal-mult: binop-onto*  $\langle \mathbb{R}_o \rangle$  *vreal-mult*

*<proof>*

Multiplicative inverse.

**global-interpretation** *vreal-inverse: v11 vreal-inverse*

**rewrites** *vreal-inverse-vdomain*[*simp*]:  $\mathcal{D}_o$  *vreal-inverse* =  $\mathbb{R}_o$

**and** *vreal-inverse-vrange*[*simp*]:  $\mathcal{R}_o$  *vreal-inverse* =  $\mathbb{R}_o$

*<proof>*

### 2.13.3 Integer numbers

#### Definition

**definition** *vint-of-int* ::  $int \Rightarrow V$

**where** *vint-of-int* = *vreal-of-real*

**notation** *vint-of-int* ( $\langle -_{\mathbb{Z}} \rangle$  [999] 999)

**declare** [[*coercion vint-of-int* ::  $int \Rightarrow V$ ]]

**definition** *vint* ::  $V (\langle \mathbb{Z}_o \rangle)$

**where** *vint* = *set (range vint-of-int)*

**definition** *int-of-vint* ::  $V \Rightarrow \text{int}$   
**where** *int-of-vint* = *inv-into UNIV vint-of-int*

Rules.

**lemma** *vint-of-int-in-vintI*[*intro, simp*]:  $a_{\mathbf{Z}} \in_{\circ} \mathbf{Z}_{\circ}$  *<proof>*

**lemma** *vint-of-int-in-vintE*[*elim*]:  
**assumes**  $a \in_{\circ} \mathbf{Z}_{\circ}$   
**obtains**  $b$  **where**  $b_{\mathbf{Z}} = a$   
*<proof>*

### Elementary properties

**lemma** *vint-vsubset-vreal*:  $\mathbf{Z}_{\circ} \subseteq_{\circ} \mathbf{R}_{\circ}$   
*<proof>*

**lemma** *inj-vint-of-int*: *inj vint-of-int*  
*<proof>*

**lemma** *vint-in-Vset- $\omega$ 2*:  $\mathbf{Z}_{\circ} \in_{\circ} \text{Vset } (\omega + \omega)$   
*<proof>*

**lemma** *int-of-vint-vint-of-int*[*simp*]: *int-of-vint* ( $a_{\mathbf{Z}}$ ) =  $a$   
*<proof>*

Transfer rules.

**definition** *cr-vint* ::  $V \Rightarrow \text{int} \Rightarrow \text{bool}$   
**where** *cr-vint*  $a b \iff (a = \text{vint-of-int } b)$

**lemma** *cr-vint-right-total*[*transfer-rule*]: *right-total cr-vint*  
*<proof>*

**lemma** *cr-vint-bi-unique*[*transfer-rule*]: *bi-unique cr-vint*  
*<proof>*

**lemma** *cr-vint-transfer-domain-rule*[*transfer-domain-rule*]:  
 $\text{Domainp } \text{cr-vint} = (\lambda x. x \in_{\circ} \mathbf{Z}_{\circ})$   
*<proof>*

**lemma** *vint-transfer*[*transfer-rule*]:  
 $(\text{rel-set } \text{cr-vint}) (\text{elts } \mathbf{Z}_{\circ}) (\text{UNIV}::\text{int set})$   
*<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*:  $[a_{\mathbf{Z}}]_{\circ} \in_{\circ} \mathbf{Z}_{\circ} \hat{\times} 1_{\mathbf{N}}$  *<proof>*

**lemma** *vint-fpair-in-fproduct-vint*:  $[a_{\mathbf{Z}}, b_{\mathbf{Z}}]_{\circ} \in_{\circ} \mathbf{Z}_{\circ} \hat{\times} 2_{\mathbf{N}}$  *<proof>*

Zero.

**lemma** *vint-zero*:  $0_{\mathbf{Z}} = (0::V)$  *<proof>*

One.

**lemma** *vint-one*:  $1_{\mathbf{Z}} = (1::V)$  *<proof>*

Addition.

**definition** *vint-plus* ::  $V$

**where** *vint-plus* =

$$(\lambda x \in_o \mathbf{Z}_o \hat{\times} 2_{\mathbf{N}}. (\text{int-of-vint } (x(0_{\mathbf{N}})) + \text{int-of-vint } (x(1_{\mathbf{N}}))))_{\mathbf{Z}}$$

**abbreviation** *vint-plus-app* (**infixl**  $\langle +_{\mathbf{Z}} \rangle$  65)

**where** *vint-plus-app*  $a \ b \equiv \text{vint-plus}(a, b)$ .

**lemma** *vint-plus-transfer*[*transfer-rule*]:

**includes** *lifting-syntax*

**shows**  $(\text{cr-vint} \implies \text{cr-vint} \implies \text{cr-vint}) \ (+_{\mathbf{Z}}) \ (+)$   
*<proof>*

Multiplication.

**definition** *vint-mult* ::  $V$

**where** *vint-mult* =

$$(\lambda x \in_o \mathbf{Z}_o \hat{\times} 2_{\mathbf{N}}. (\text{int-of-vint } (x(0_{\mathbf{N}})) * \text{int-of-vint } (x(1_{\mathbf{N}}))))_{\mathbf{Z}}$$

**abbreviation** *vint-mult-app* (**infixl**  $\langle *_{\mathbf{Z}} \rangle$  65)

**where** *vint-mult-app*  $a \ b \equiv \text{vint-mult}(a, b)$ .

**lemma** *vint-mult-transfer*[*transfer-rule*]:

**includes** *lifting-syntax*

**shows**  $(\text{cr-vint} \implies \text{cr-vint} \implies \text{cr-vint}) \ (*_{\mathbf{Z}}) \ (*)$   
*<proof>*

Unary minus.

**definition** *vint-uminus* ::  $V$

**where** *vint-uminus* =  $(\lambda x \in_o \mathbf{Z}_o. (\text{uminus } (\text{int-of-vint } x)))_{\mathbf{Z}}$

**abbreviation** *vint-uminus-app* ( $\langle -_{\mathbf{Z}} \rightarrow$  [81] 80)

**where**  $-_{\mathbf{Z}} \ a \equiv \text{vint-uminus}(a)$

**lemma** *vint-uminus-transfer*[*transfer-rule*]:

**includes** *lifting-syntax*

**shows**  $(\text{cr-vint} \implies \text{cr-vint}) \ (\text{vint-uminus-app}) \ (\text{uminus})$   
*<proof>*

Order.

**definition** *vint-le* ::  $V$

**where** *vint-le* =

$$\text{set } \{[a, b]_o \mid a \ b. [a, b]_o \in_o \mathbf{Z}_o \hat{\times} 2_{\mathbf{N}} \wedge \text{int-of-vint } a \leq \text{int-of-vint } b\}$$

**abbreviation** *vint-le'* ( $\langle (-/ \leq_{\mathbf{Z}} -) \rangle$  [51, 51] 50)

**where**  $a \leq_{\mathbf{Z}} \ b \equiv [a, b]_o \in_o \text{vint-le}$

**lemma** *small-vint-le*[*simp*]:

*small*  $\{[a, b]_o \mid a \ b. [a, b]_o \in_o \mathbf{Z}_o \hat{\times} 2_{\mathbf{N}} \wedge \text{int-of-vint } a \leq \text{int-of-vint } b\}$   
*<proof>*

**lemma** *vint-le-transfer*[*transfer-rule*]:

**includes** *lifting-syntax*

**shows**  $(\text{cr-vint} \implies \text{cr-vint} \implies (=)) \ \text{vint-le}' \ (\leq)$   
*<proof>*

Strict order.

**definition** *vint-ls* ::  $V$

**where** *vint-ls* =

$set \{[a, b]_o \mid a, b, [a, b]_o \in_o \mathbb{Z}_o \hat{\times} 2_{\mathbb{N}} \wedge int-of-vint\ a < int-of-vint\ b\}$

**abbreviation** *vint-ls'* ( $\langle -/ <_{\mathbb{Z}} - \rangle$ ) [51, 51] 50)

**where**  $a <_{\mathbb{Z}} b \equiv [a, b]_o \in_o vint-ls$

**lemma** *small-vint-ls[simp]*:

*small*  $\{[a, b]_o \mid a, b, [a, b]_o \in_o \mathbb{Z}_o \hat{\times} 2_{\mathbb{N}} \wedge int-of-vint\ a < int-of-vint\ b\}$   
*<proof>*

**lemma** *vint-ls-transfer[transfer-rule]*:

**includes** *lifting-syntax*

**shows**  $(cr-vint \implies cr-vint \implies (=))\ vint-ls' (<)$

*<proof>*

Subtraction.

**definition** *vint-minus* ::  $V$

**where** *vint-minus* =

$(\lambda x \in_o \mathbb{Z}_o \hat{\times} 2_{\mathbb{N}}. (int-of-vint\ (x(0_{\mathbb{N}})) - int-of-vint\ (x(1_{\mathbb{N}}))))_{\mathbb{Z}}$

**abbreviation** *vint-minus-app* (**infixl**  $\langle -_{\mathbb{Z}} \rangle$  65)

**where**  $vint-minus-app\ a\ b \equiv vint-minus([a, b])_{\bullet}$

**lemma** *vint-minus-transfer[transfer-rule]*:

**includes** *lifting-syntax*

**shows**  $(cr-vint \implies cr-vint \implies cr-vint)\ (-_{\mathbb{Z}})\ (-)$

*<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*:  $0_{\mathbb{Z}} \in_o \mathbb{Z}_o$  *<proof>*

**lemma** *vint-one-closed*:  $1_{\mathbb{Z}} \in_o \mathbb{Z}_o$  *<proof>*

**lemma** *vint-plus-closed*:

**assumes**  $x \in_o \mathbb{Z}_o$  **and**  $y \in_o \mathbb{Z}_o$

**shows**  $x +_{\mathbb{Z}} y \in_o \mathbb{Z}_o$

*<proof>*

**lemma** *vint-mult-closed*:

**assumes**  $x \in_o \mathbb{Z}_o$  **and**  $y \in_o \mathbb{Z}_o$

**shows**  $x *_{\mathbb{Z}} y \in_o \mathbb{Z}_o$

*<proof>*

**lemma** *vint-uminus-closed*:

**assumes**  $x \in_o \mathbb{Z}_o$

**shows**  $-_{\mathbb{Z}} x \in_o \mathbb{Z}_o$

*<proof>*

Associative Law for Addition: Definition 1.4.1.a.

**lemma** *vint-assoc-law-addition*:

**assumes**  $x \in_o \mathbb{Z}_o$  **and**  $y \in_o \mathbb{Z}_o$  **and**  $z \in_o \mathbb{Z}_o$

**shows**  $(x +_{\mathbf{Z}} y) +_{\mathbf{Z}} z = x +_{\mathbf{Z}} (y +_{\mathbf{Z}} z)$   
 ⟨proof⟩

Commutative Law for Addition: Definition 1.4.1.b.

**lemma** *vint-commutative-law-addition*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x +_{\mathbf{Z}} y = y +_{\mathbf{Z}} x$   
 ⟨proof⟩

Identity Law for Addition: Definition 1.4.1.c.

**lemma** *vint-identity-law-addition*:  
**assumes** [*simp*]:  $x \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x +_{\mathbf{Z}} 0_{\mathbf{Z}} = x$   
 ⟨proof⟩

Inverses Law for Addition: Definition 1.4.1.d.

**lemma** *vint-inverses-law-addition*:  
**assumes** [*simp*]:  $x \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x +_{\mathbf{Z}} (-_{\mathbf{Z}} x) = 0_{\mathbf{Z}}$   
 ⟨proof⟩

Associative Law for Multiplication: Definition 1.4.1.e.

**lemma** *vint-assoc-law-multiplication*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $z \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $(x *_{\mathbf{Z}} y) *_{\mathbf{Z}} z = x *_{\mathbf{Z}} (y *_{\mathbf{Z}} z)$   
 ⟨proof⟩

Commutative Law for Multiplication: Definition 1.4.1.f.

**lemma** *vint-commutative-law-multiplication*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x *_{\mathbf{Z}} y = y *_{\mathbf{Z}} x$   
 ⟨proof⟩

Identity Law for multiplication: Definition 1.4.1.g.

**lemma** *vint-identity-law-multiplication*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x *_{\mathbf{Z}} 1_{\mathbf{Z}} = x$   
 ⟨proof⟩

Distributive Law for Multiplication: Definition 1.4.1.h.

**lemma** *vint-distributive-law*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $z \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x *_{\mathbf{Z}} (y +_{\mathbf{Z}} z) = (x *_{\mathbf{Z}} y) +_{\mathbf{Z}} (x *_{\mathbf{Z}} z)$   
 ⟨proof⟩

No Zero Divisors Law: Definition 1.4.1.i.

**lemma** *vint-no-zero-divisors-law*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $x *_{\mathbf{Z}} y = 0_{\mathbf{Z}}$   
**shows**  $x = 0_{\mathbf{Z}} \vee y = 0_{\mathbf{Z}}$   
 ⟨proof⟩

Trichotomy Law: Definition 1.4.1.j

**lemma** *vint-trichotomy-law*:  
**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  
 $(x <_{\mathbf{Z}} y \wedge \sim(x = y) \wedge \sim(y <_{\mathbf{Z}} x)) \vee$

$$(\sim(x <_{\mathbf{Z}} y) \wedge x = y \wedge \sim(y <_{\mathbf{Z}} x)) \vee$$

$$(\sim(x <_{\mathbf{Z}} y) \wedge \sim(x = y) \wedge y <_{\mathbf{Z}} x)$$

*<proof>*

Transitive Law: Definition 1.4.1.k

**lemma** *vint-transitive-law*:

**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $z \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $x <_{\mathbf{Z}} y$   
**and**  $y <_{\mathbf{Z}} z$   
**shows**  $x <_{\mathbf{Z}} z$

*<proof>*

Addition Law of Order: Definition 1.4.1.l

**lemma** *vint-addition-law-of-order*:

**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $z \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $x <_{\mathbf{Z}} y$   
**shows**  $x +_{\mathbf{Z}} z <_{\mathbf{Z}} y +_{\mathbf{Z}} z$

*<proof>*

Multiplication Law of Order: Definition 1.4.1.m

**lemma** *vint-multiplication-law-of-order*:

**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $z \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $x <_{\mathbf{Z}} y$   
**and**  $0_{\mathbf{Z}} <_{\mathbf{Z}} z$   
**shows**  $x *_{\mathbf{Z}} z <_{\mathbf{Z}} y *_{\mathbf{Z}} z$

*<proof>*

Non-Triviality: Definition 1.4.1.n

**lemma** *vint-non-triviality*:  $0_{\mathbf{Z}} \neq 1_{\mathbf{Z}}$

*<proof>*

Well-Ordering Principle.

**lemma** *well-ordering-principle*:

**assumes**  $A \subseteq_{\circ} \mathbf{Z}_{\circ}$   
**and**  $a \in_{\circ} \mathbf{Z}_{\circ}$   
**and**  $A \neq 0$   
**and**  $\bigwedge x. x \in_{\circ} A \implies a <_{\mathbf{Z}} x$   
**obtains**  $b$  **where**  $b \in_{\circ} A$  **and**  $\bigwedge x. x \in_{\circ} A \implies b \leq_{\mathbf{Z}} x$

*<proof>*

## Fundamental properties of other operations

Minus.

**lemma** *vint-minus-closed*:

**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x -_{\mathbf{Z}} y \in_{\circ} \mathbf{Z}_{\circ}$

*<proof>*

**lemma** *vint-minus-eq-plus-uminus*:

**assumes**  $x \in_{\circ} \mathbf{Z}_{\circ}$  **and**  $y \in_{\circ} \mathbf{Z}_{\circ}$   
**shows**  $x -_{\mathbf{Z}} y = x +_{\mathbf{Z}} (-_{\mathbf{Z}} y)$

*<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*  $\langle \mathbb{Z}_{\circ} \rangle$  *vint-plus*

*<proof>*

Unary minus.

**global-interpretation** *vint-uminus*: *v11* *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*  $\langle \mathbb{Z}_{\circ} \rangle$  *vint-mult*

*<proof>*

Misc.

**lemma** (**in**  $\mathcal{Z}$ ) *vint-in-Vset*[*intro*]:  $\mathbb{Z}_{\circ} \in_{\circ} Vset \alpha$

*<proof>*

## 2.13.4 Rational numbers

### Definition

**definition** *vrat-of-rat* ::  $rat \Rightarrow V$

**where** *vrat-of-rat*  $x = vreal-of-real (real-of-rat x)$

**notation** *vrat-of-rat* ( $\langle -_{\mathbb{Q}} \rangle$  [999] 999)

**declare** [[*coercion* *vrat-of-rat* ::  $rat \Rightarrow V$ ]]

**definition** *vrat* ::  $V (\langle \mathbb{Q}_{\circ} \rangle)$

**where** *vrat* = *set* (*range* *vrat-of-rat*)

**definition** *rat-of-vrat* ::  $V \Rightarrow rat$

**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- $\omega$ 2*:  $\mathbb{Q}_{\circ} \in_{\circ} Vset (\omega + \omega)$

*<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* ::  $V \Rightarrow \text{rat} \Rightarrow \text{bool}$   
**where** *cr-vrat*  $a b \longleftrightarrow (a = \text{vrat-of-rat } b)$

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

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

**lemma** *cr-vrat-transfer-domain-rule[transfer-domain-rule]*:  
 $\text{Domainp } \text{cr-vrat} = (\lambda x. x \in_{\circ} \mathbb{Q}_{\circ})$   
 ⟨*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*:  $[a_{\mathbb{Q}}]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 1_{\mathbb{N}}$  ⟨*proof*⟩

**lemma** *vrat-fpair-in-fproduct-vrat*:  $[a_{\mathbb{Q}}, b_{\mathbb{Q}}]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}}$  ⟨*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* =  
 $(\lambda x \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}}. (\text{rat-of-vrat } (x(0_{\mathbb{N}})) + \text{rat-of-vrat } (x(1_{\mathbb{N}}))))_{\mathbb{Q}}$

**abbreviation** *vrat-plus-app* (**infixl**  $\langle +_{\mathbb{Q}} \rangle$  65)  
**where** *vrat-plus-app*  $a b \equiv \text{vrat-plus}(a, b)$ .

**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* =  
 $(\lambda x \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}}. (\text{rat-of-vrat } (x(0_{\mathbb{N}})) * \text{rat-of-vrat } (x(1_{\mathbb{N}}))))_{\mathbb{Q}}$

**abbreviation** *vrat-mult-app* (**infixl**  $\langle *_{\mathbb{Q}} \rangle$  65)  
**where** *vrat-mult-app*  $a b \equiv \text{vrat-mult}(a, b)$ .

**lemma** *vrat-mult-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vrat \implies cr\text{-}vrat \implies cr\text{-}vrat$ ) ( $*_{\mathbb{Q}}$ ) ( $*$ )  
*\langle proof \rangle*

Unary minus.

**definition** *vrat-uminus* ::  $V$   
**where** *vrat-uminus* =  $(\lambda x \in_{\circ} \mathbb{Q}_{\circ}. (\text{uminus} (\text{rat-of-vrat } x))_{\mathbb{Q}})$

**abbreviation** *vrat-uminus-app* ( $\langle -_{\mathbb{Q}} \rightarrow$  [81] 80)  
**where**  $-_{\mathbb{Q}} a \equiv \text{vrat-uminus}(a)$

**lemma** *vrat-uminus-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vrat \implies cr\text{-}vrat$ ) (*vrat-uminus-app*) (*uminus*)  
*\langle proof \rangle*

Multiplicative inverse.

**definition** *vrat-inverse* ::  $V$   
**where** *vrat-inverse* =  $(\lambda x \in_{\circ} \mathbb{Q}_{\circ}. (\text{inverse} (\text{rat-of-vrat } x))_{\mathbb{Q}})$

**abbreviation** *vrat-inverse-app* ( $\langle (-^{-1}_{\mathbb{Q}}) \rangle$  [1000] 999)  
**where**  $a^{-1}_{\mathbb{Q}} \equiv \text{vrat-inverse}(a)$

**lemma** *vrat-inverse-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vrat \implies cr\text{-}vrat$ ) (*vrat-inverse-app*) (*inverse*)  
*\langle proof \rangle*

Order.

**definition** *vrat-le* ::  $V$   
**where** *vrat-le* =  
 $set \{ [a, b]_{\circ} \mid a b. [a, b]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}} \wedge \text{rat-of-vrat } a \leq \text{rat-of-vrat } b \}$

**abbreviation** *vrat-le'* ( $\langle (-/ \leq_{\mathbb{Q}} -) \rangle$  [51, 51] 50)  
**where**  $a \leq_{\mathbb{Q}} b \equiv [a, b]_{\circ} \in_{\circ} \text{vrat-le}$

**lemma** *small-vrat-le*[*simp*]:  
 $small \{ [a, b]_{\circ} \mid a b. [a, b]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}} \wedge \text{rat-of-vrat } a \leq \text{rat-of-vrat } b \}$   
*\langle proof \rangle*

**lemma** *vrat-le-transfer*[*transfer-rule*]:  
**includes** *lifting-syntax*  
**shows** ( $cr\text{-}vrat \implies cr\text{-}vrat \implies (=)$ ) *vrat-le'* ( $\leq$ )  
*\langle proof \rangle*

Strict order.

**definition** *vrat-ls* ::  $V$   
**where** *vrat-ls* =  
 $set \{ [a, b]_{\circ} \mid a b. [a, b]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \hat{\times} 2_{\mathbb{N}} \wedge \text{rat-of-vrat } a < \text{rat-of-vrat } b \}$

**abbreviation** *vrat-ls'* ( $\langle (-/ <_{\mathbb{Q}} -) \rangle$  [51, 51] 50)  
**where**  $a <_{\mathbb{Q}} b \equiv [a, b]_{\circ} \in_{\circ} \text{vrat-ls}$

**lemma** *small-vrat-ls[simp]*:  
*small*  $\{[a, b]_{\circ} \mid a, b. [a, b]_{\circ} \in_{\circ} \mathbb{Q}_{\circ} \widehat{\times} 2_{\mathbb{N}} \wedge \text{rat-of-vrat } a < \text{rat-of-vrat } b\}$   
*<proof>*

**lemma** *vrat-ls-transfer[transfer-rule]*:  
**includes** *lifting-syntax*  
**shows**  $(\text{cr-vrat} \implies \text{cr-vrat} \implies (=)) \text{ vrat-ls}' (<)$   
*<proof>*

Subtraction.

**definition** *vrat-minus* ::  $V$   
**where** *vrat-minus* =  
 $(\lambda x \in_{\circ} \mathbb{Q}_{\circ} \widehat{\times} 2_{\mathbb{N}}. (\text{rat-of-vrat } (x(0_{\mathbb{N}})) - \text{rat-of-vrat } (x(1_{\mathbb{N}}))))_{\mathbb{Q}}$

**abbreviation** *vrat-minus-app* (**infixl**  $\langle -_{\mathbb{Q}} \rangle$  65)  
**where** *vrat-minus-app*  $a \ b \equiv \text{vrat-minus}(a, b)$ .

**lemma** *vrat-minus-transfer[transfer-rule]*:  
**includes** *lifting-syntax*  
**shows**  $(\text{cr-vrat} \implies \text{cr-vrat} \implies \text{cr-vrat})$   
 $(-_{\mathbb{Q}}) (-)$   
*<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**  $x \in_0 \mathbb{Q}_0$  **and**  $y \in_0 \mathbb{Q}_0$

**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*]:  $x \in_0 \mathbb{Q}_0$

**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*]:  $x \in_0 \mathbb{Q}_0$

**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**  $x \in_0 \mathbb{Q}_0$  **and**  $y \in_0 \mathbb{Q}_0$  **and**  $z \in_0 \mathbb{Q}_0$

**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**  $x \in_0 \mathbb{Q}_0$  **and**  $y \in_0 \mathbb{Q}_0$

**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**  $x \in_0 \mathbb{Q}_0$

**shows**  $x *_{\mathbb{Q}} 1_{\mathbb{Q}} = x$

*<proof>*

Inverses Law for Multiplication: Definition 2.2.1.8.

**lemma** *vrat-inverses-law-multiplication:*

**assumes**  $x \in_0 \mathbb{Q}_0$  **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**  $x \in_0 \mathbb{Q}_0$  **and**  $y \in_0 \mathbb{Q}_0$  **and**  $z \in_0 \mathbb{Q}_0$

**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**  $x \in_0 \mathbb{Q}_0$  **and**  $y \in_0 \mathbb{Q}_0$

**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**  $x \in_o \mathbb{Q}_o$   
**and**  $y \in_o \mathbb{Q}_o$   
**and**  $z \in_o \mathbb{Q}_o$   
**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**  $x \in_o \mathbb{Q}_o$  **and**  $y \in_o \mathbb{Q}_o$  **and**  $z \in_o \mathbb{Q}_o$  **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**  $x \in_o \mathbb{Q}_o$   
**and**  $y \in_o \mathbb{Q}_o$   
**and**  $z \in_o \mathbb{Q}_o$   
**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**  $x \in_o \mathbb{Q}_o$  **and**  $y \in_o \mathbb{Q}_o$   
**shows**  $x -_{\mathbb{Q}} y \in_o \mathbb{Q}_o$

*<proof>*

**lemma** *vrat-minus-eq-plus-uminus*:

**assumes**  $x \in_o \mathbb{Q}_o$  **and**  $y \in_o \mathbb{Q}_o$   
**shows**  $x -_{\mathbb{Q}} y = x +_{\mathbb{Q}} (-_{\mathbb{Q}} y)$

*<proof>*

Unary minus.

**lemma** *vrat-uminus-uminus*:

**assumes**  $x \in_o \mathbb{Q}_o$   
**shows**  $x = -_{\mathbb{Q}} (-_{\mathbb{Q}} x)$

*<proof>*

Multiplicative inverse.

**lemma** *vrat-inverse-inverse*:

**assumes**  $x \in_o \mathbb{Q}_o$   
**shows**  $x = (x^{-1}_{\mathbb{Q}})^{-1}_{\mathbb{Q}}$

*<proof>*

**Further properties**

Addition.

**global-interpretation** *vrat-plus*: *binop-onto*  $\langle \mathbb{Q}_o \rangle$  *vrat-plus*  
 ⟨*proof*⟩

Unary minus.

**global-interpretation** *vrat-uminus*: *v11* *vrat-uminus*  
**rewrites** *vrat-uminus-vdomain*[*simp*]:  $\mathcal{D}_o$  *vrat-uminus* =  $\mathbb{Q}_o$   
**and** *vrat-uminus-vrange*[*simp*]:  $\mathcal{R}_o$  *vrat-uminus* =  $\mathbb{Q}_o$   
 ⟨*proof*⟩

Multiplication.

**global-interpretation** *vrat-mult*: *binop-onto*  $\langle \mathbb{Q}_o \rangle$  *vrat-mult*  
 ⟨*proof*⟩

Multiplicative inverse.

**global-interpretation** *vrat-inverse*: *v11* *vrat-inverse*  
**rewrites** *vrat-inverse-vdomain*[*simp*]:  $\mathcal{D}_o$  *vrat-inverse* =  $\mathbb{Q}_o$   
**and** *vrat-inverse-vrange*[*simp*]:  $\mathcal{R}_o$  *vrat-inverse* =  $\mathbb{Q}_o$   
 ⟨*proof*⟩

Misc.

**lemma** (**in**  $\mathcal{Z}$ ) *vrat-in-Vset*[*intro*]:  $\mathbb{Q}_o \in_o Vset \alpha$   
 ⟨*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*  $\mathbb{R}_o$ )  
 ⟨*proof*⟩

**lemma** *vreal-vlepoll-VPow-omega*:  $\mathbb{R}_o \lesssim_o VPow \omega$   
 ⟨*proof*⟩

**lemma** (**in**  $\mathcal{Z}$ ) *vreal-in-Vset*[*intro*]:  $\mathbb{R}_o \in_o Vset \alpha$   
 ⟨*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]<sup>3</sup>

**definition** *repl-ex-fun* ::  $V$   
**where** *repl-ex-fun* =  $(\lambda i \in_{\circ} \omega. V\text{from } \omega \ i)$

**mk-VLambda** *repl-ex-fun-def*  
 $|vsv \text{ repl-ex-fun-vsuv}|$   
 $|vdomain \text{ repl-ex-fun-vdomain}|$   
 $|app \text{ repl-ex-fun-app}|$

**lemma** *repl-ex-fun-vrange*:  $\mathcal{R}_{\circ} \text{ repl-ex-fun } \subseteq_{\circ} Vset (\omega + \omega)$   
 $\langle proof \rangle$

**lemma** *Limit-vsuv-not-in-Vset-if-vrange-not-in-Vset*:  
**assumes** *Limit*  $\alpha$  **and**  $\mathcal{R}_{\circ} f \notin_{\circ} Vset \alpha$   
**shows**  $f \notin_{\circ} Vset \alpha$   
 $\langle proof \rangle$

**lemma** *Ord-not-in-Vset*:  
**assumes** *Ord*  $\alpha$   
**shows**  $\alpha \notin_{\circ} Vset \alpha$   
 $\langle proof \rangle$

**lemma** *Ord-succ-vsusbset-Vfrom-succ*:  
**assumes** *Transset*  $A$  **and** *Ord*  $a$  **and**  $a \in_{\circ} V\text{from } A \ i$   
**shows**  $\text{succ } a \subseteq_{\circ} V\text{from } A (\text{succ } i)$   
 $\langle proof \rangle$

**lemma** *Ord-succ-in-Vfrom-succ*:  
**assumes** *Transset*  $A$  **and** *Ord*  $a$  **and**  $a \in_{\circ} V\text{from } A \ i$   
**shows**  $\text{succ } a \in_{\circ} V\text{from } A (\text{succ } (\text{succ } i))$   
 $\langle proof \rangle$

**lemma**  *$\omega$ -vplus-in-Vfrom- $\omega$* :  
**assumes**  $j \in_{\circ} \omega$   
**shows**  $\omega + j \in_{\circ} V\text{from } \omega (\text{succ } (2_{\mathbb{N}} * j))$   
 $\langle proof \rangle$

**lemma** *repl-ex-fun-vrange-not-in-Vset*:  $\mathcal{R}_{\circ} \text{ repl-ex-fun } \notin_{\circ} Vset (\omega + \omega)$   
 $\langle proof \rangle$

**lemma** *repl-ex-fun-not-in-Vset*:  $\text{repl-ex-fun} \notin_{\circ} Vset (\omega + \omega)$   
 $\langle proof \rangle$

---

<sup>3</sup>[https://en.wikipedia.org/wiki/Zermelo\\_set\\_theory](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* =  $\mathcal{Z}$   $\alpha$  + *vfsequence*  $\mathfrak{S}$  **for**  $\alpha$   $\mathfrak{S}$  +  
**assumes** *vrange-vsubset-Vset*:  $\mathcal{R}_\circ \mathfrak{S} \sqsubseteq_\circ Vset \alpha$

Rules.

**lemma**  $\mathcal{Z}$ -*vfsequenceI*[*intro*]:  
**assumes**  $\mathcal{Z} \alpha$  **and** *vfsequence*  $\mathfrak{S}$  **and**  $\mathcal{R}_\circ \mathfrak{S} \sqsubseteq_\circ Vset \alpha$   
**shows**  $\mathcal{Z}$ -*vfsequence*  $\alpha \mathfrak{S}$   
*<proof>*

**lemmas**  $\mathcal{Z}$ -*vfsequenceD*[*dest*] =  $\mathcal{Z}$ -*vfsequence. axioms*

**lemma**  $\mathcal{Z}$ -*vfsequenceE*[*elim*]:  
**assumes**  $\mathcal{Z}$ -*vfsequence*  $\alpha \mathfrak{S}$   
**obtains**  $\mathcal{Z} \alpha$  **and** *vfsequence*  $\mathfrak{S}$  **and**  $\mathcal{R}_\circ \mathfrak{S} \sqsubseteq_\circ Vset \alpha$   
*<proof>*

Elementary properties.

**context**  $\mathcal{Z}$ -*vfsequence*  
**begin**

**lemma** (**in**  $\mathcal{Z}$ -*vfsequence*)  $\mathcal{Z}$ -*vfsequence-vdomain-in-Vset*[*intro, simp*]:  
 $\mathcal{D}_\circ \mathfrak{S} \in_\circ Vset \alpha$   
*<proof>*

**lemma** (**in**  $\mathcal{Z}$ -*vfsequence*)  $\mathcal{Z}$ -*vfsequence-vrange-in-Vset*[*intro, simp*]:  
 $\mathcal{R}_\circ \mathfrak{S} \in_\circ Vset \alpha$   
*<proof>*

**lemma** (**in**  $\mathcal{Z}$ -*vfsequence*)  $\mathcal{Z}$ -*vfsequence-struct-in-Vset*:  $\mathfrak{S} \in_\circ Vset \alpha$   
*<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*  $\alpha \mathfrak{S}$  **for**  $\alpha \mathfrak{S}$  +  
**assumes**  $\mathcal{Z}$ -*ts-length*:  $2_{\mathbb{N}} \leq vcard \mathfrak{S}$   
**and**  $\mathcal{Z}$ -*ts-closed*[*intro*]:  $A \in_\circ \mathfrak{S}(\mathcal{T}) \implies A \sqsubseteq_\circ \mathfrak{S}(\mathcal{A})$   
**and**  $\mathcal{Z}$ -*ts-domain*[*intro, simp*]:  $\mathfrak{S}(\mathcal{A}) \in_\circ \mathfrak{S}(\mathcal{T})$   
**and**  $\mathcal{Z}$ -*ts-vintersection*[*intro*]:  
 $A \in_\circ \mathfrak{S}(\mathcal{T}) \implies B \in_\circ \mathfrak{S}(\mathcal{T}) \implies A \cap_\circ B \in_\circ \mathfrak{S}(\mathcal{T})$   
**and**  $\mathcal{Z}$ -*ts-VUnion*[*intro*]:  $X \sqsubseteq_\circ \mathfrak{S}(\mathcal{T}) \implies \bigcup_\circ X \in_\circ \mathfrak{S}(\mathcal{T})$

Rules.

**lemma**  $Z\text{-tsI}[\text{intro}]$ :

**assumes**  $Z\text{-vfsequence } \alpha \mathfrak{S}$

**and**  $2_{\mathbb{N}} \leq \text{vcard } \mathfrak{S}$

**and**  $\bigwedge A. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \subseteq_{\circ} \mathfrak{S}(\mathcal{A})$

**and**  $\mathfrak{S}(\mathcal{A}) \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge A B. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies B \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge X. X \subseteq_{\circ} \mathfrak{S}(\mathcal{T}) \implies \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\mathcal{T})$

**shows**  $Z\text{-ts } \alpha \mathfrak{S}$

$\langle \text{proof} \rangle$

**lemma**  $Z\text{-tsD}[\text{dest}]$ :

**assumes**  $Z\text{-ts } \alpha \mathfrak{S}$

**shows**  $Z\text{-vfsequence } \alpha \mathfrak{S}$

**and**  $2_{\mathbb{N}} \leq \text{vcard } \mathfrak{S}$

**and**  $\bigwedge A. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \subseteq_{\circ} \mathfrak{S}(\mathcal{A})$

**and**  $\mathfrak{S}(\mathcal{A}) \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge A B. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies B \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge X. X \subseteq_{\circ} \mathfrak{S}(\mathcal{T}) \implies \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\mathcal{T})$

$\langle \text{proof} \rangle$

**lemma**  $Z\text{-tsE}[\text{elim}]$ :

**assumes**  $Z\text{-ts } \alpha \mathfrak{S}$

**obtains**  $Z\text{-vfsequence } \alpha \mathfrak{S}$

**and**  $2_{\mathbb{N}} \leq \text{vcard } \mathfrak{S}$

**and**  $\bigwedge A. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \subseteq_{\circ} \mathfrak{S}(\mathcal{A})$

**and**  $\mathfrak{S}(\mathcal{A}) \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge A B. A \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies B \in_{\circ} \mathfrak{S}(\mathcal{T}) \implies A \cap_{\circ} B \in_{\circ} \mathfrak{S}(\mathcal{T})$

**and**  $\bigwedge X. X \subseteq_{\circ} \mathfrak{S}(\mathcal{T}) \implies \bigcup_{\circ} X \in_{\circ} \mathfrak{S}(\mathcal{T})$

$\langle \text{proof} \rangle$

Elementary properties.

**lemma** (in  $Z\text{-ts}$ )  $Z\text{-ts-vempty-in-ts}$ :  $0 \in_{\circ} \mathfrak{S}(\mathcal{T})$

$\langle \text{proof} \rangle$

### 2.15.4 Indiscrete topology

**definition**  $ts\text{-indiscrete} :: V \Rightarrow V$

**where**  $ts\text{-indiscrete } A = [A, \text{set } \{0, A\}]_{\circ}$ .

**named-theorems**  $ts\text{-indiscrete-simps}$

**lemma**  $ts\text{-indiscrete-A}[ts\text{-indiscrete-simps}]$ :  $ts\text{-indiscrete } A(\mathcal{A}) = A$

$\langle \text{proof} \rangle$

**lemma**  $ts\text{-indiscrete-T}[ts\text{-indiscrete-simps}]$ :  $ts\text{-indiscrete } A(\mathcal{T}) = \text{set } \{0, A\}$

$\langle \text{proof} \rangle$

**lemma** (in  $Z$ )  $Z\text{-ts-ts-indiscrete}$ :

**assumes**  $A \in_{\circ} V\text{set } \alpha$

**shows**  $Z\text{-ts } \alpha (ts\text{-indiscrete } A)$

$\langle \text{proof} \rangle$

## 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*] = *A-def*

### 2.16.2 Semigroup

#### Foundations

**definition** *mbinop* where [*sgrp-struct-field-simps*]: *mbinop* =  $1_{\mathbb{N}}$

**locale** *Z-sgrp-basis* = *Z-vfsequence*  $\alpha$   $\mathfrak{S}$  + *op*: *binop*  $\langle \mathfrak{S}(\mathcal{A}) \rangle$   $\langle \mathfrak{S}(\text{mbinop}) \rangle$   
**for**  $\alpha$   $\mathfrak{S}$  +  
**assumes** *Z-sgrp-length*: *vcard*  $\mathfrak{S}$  =  $2_{\mathbb{N}}$

**abbreviation** *sgrp-app* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \odot_{\circ 1} \rangle$  70)  
**where** *sgrp-app*  $\mathfrak{S}$   $a$   $b \equiv \mathfrak{S}(\text{mbinop})(a, b)$ .

**notation** *sgrp-app* (**infixl**  $\langle \odot_{\circ} \rangle$  70)

Rules.

**lemma** *Z-sgrp-basisI*[*intro*]:  
**assumes** *Z-vfsequence*  $\alpha$   $\mathfrak{S}$   
**and** *vcard*  $\mathfrak{S}$  =  $2_{\mathbb{N}}$   
**and** *binop*  $(\mathfrak{S}(\mathcal{A}))$   $(\mathfrak{S}(\text{mbinop}))$   
**shows** *Z-sgrp-basis*  $\alpha$   $\mathfrak{S}$   
*<proof>*

**lemma** *Z-sgrp-basisD*[*dest*]:  
**assumes** *Z-sgrp-basis*  $\alpha$   $\mathfrak{S}$   
**shows** *Z-vfsequence*  $\alpha$   $\mathfrak{S}$   
**and** *vcard*  $\mathfrak{S}$  =  $2_{\mathbb{N}}$   
**and** *binop*  $(\mathfrak{S}(\mathcal{A}))$   $(\mathfrak{S}(\text{mbinop}))$   
*<proof>*

**lemma** *Z-sgrp-basisE*[*elim*]:  
**assumes** *Z-sgrp-basis*  $\alpha$   $\mathfrak{S}$   
**shows** *Z-vfsequence*  $\alpha$   $\mathfrak{S}$   
**and** *vcard*  $\mathfrak{S}$  =  $2_{\mathbb{N}}$   
**and** *binop*  $(\mathfrak{S}(\mathcal{A}))$   $(\mathfrak{S}(\text{mbinop}))$   
*<proof>*

#### Simple semigroup

**locale** *Z-sgrp* = *Z-sgrp-basis*  $\alpha$   $\mathfrak{S}$  **for**  $\alpha$   $\mathfrak{S}$  +  
**assumes** *Z-sgrp-assoc*:  
 $\llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   
 $(a \odot_{\circ \mathfrak{S}} b) \odot_{\circ \mathfrak{S}} c = a \odot_{\circ \mathfrak{S}} (b \odot_{\circ \mathfrak{S}} c)$

Rules.

**lemma** *Z-sgrpI*[*intro*]:  
**assumes** *Z-sgrp-basis*  $\alpha$   $\mathfrak{S}$   
**and**  $\bigwedge a b c. \llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   
 $(a \odot_{\circ \mathfrak{S}} b) \odot_{\circ \mathfrak{S}} c = a \odot_{\circ \mathfrak{S}} (b \odot_{\circ \mathfrak{S}} c)$

**shows**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

$\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-sgrp}D[\text{dest}]$ :

**assumes**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

**shows**  $\mathcal{Z}\text{-sgrp-basis } \alpha \mathfrak{S}$

**and**  $\bigwedge a b c. \llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}); c \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies$

$(a \circ_{\circ\mathfrak{S}} b) \circ_{\circ\mathfrak{S}} c = a \circ_{\circ\mathfrak{S}} (b \circ_{\circ\mathfrak{S}} c)$

$\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-sgrp}E[\text{elim}]$ :

**assumes**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

**obtains**  $\mathcal{Z}\text{-sgrp-basis } \alpha \mathfrak{S}$

**and**  $\bigwedge a b c. \llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}); c \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies$

$(a \circ_{\circ\mathfrak{S}} b) \circ_{\circ\mathfrak{S}} c = a \circ_{\circ\mathfrak{S}} (b \circ_{\circ\mathfrak{S}} c)$

$\langle \text{proof} \rangle$

### 2.16.3 Commutative semigroup

**locale**  $\mathcal{Z}\text{-csgrp} = \mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$  **for**  $\alpha \mathfrak{S} +$

**assumes**  $\mathcal{Z}\text{-csgrp-commutative}$ :

$\llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies a \circ_{\circ\mathfrak{S}} b = b \circ_{\circ\mathfrak{S}} a$

Rules.

**lemma**  $\mathcal{Z}\text{-csgrp}I[\text{intro}]$ :

**assumes**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

**and**  $\bigwedge a b. \llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies a \circ_{\circ\mathfrak{S}} b = b \circ_{\circ\mathfrak{S}} a$

**shows**  $\mathcal{Z}\text{-csgrp } \alpha \mathfrak{S}$

$\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-csgrp}D[\text{dest}]$ :

**assumes**  $\mathcal{Z}\text{-csgrp } \alpha \mathfrak{S}$

**shows**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

**and**  $\bigwedge a b. \llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies a \circ_{\circ\mathfrak{S}} b = b \circ_{\circ\mathfrak{S}} a$

$\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-csgrp}E[\text{elim}]$ :

**assumes**  $\mathcal{Z}\text{-csgrp } \alpha \mathfrak{S}$

**obtains**  $\mathcal{Z}\text{-sgrp } \alpha \mathfrak{S}$

**and**  $\bigwedge a b. \llbracket a \in_0 \mathfrak{S}(\mathcal{A}); b \in_0 \mathfrak{S}(\mathcal{A}) \rrbracket \implies a \circ_{\circ\mathfrak{S}} b = b \circ_{\circ\mathfrak{S}} a$

$\langle \text{proof} \rangle$

### 2.16.4 Semiring

#### Foundations

**definition**  $vplus :: V$  **where**  $[sgrp\text{-struct-field-simps}]$ :  $vplus = 1_N$

**definition**  $vmult :: V$  **where**  $[sgrp\text{-struct-field-simps}]$ :  $vmult = 2_N$

**abbreviation**  $vplus\text{-app} :: V \Rightarrow V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle +_{\circ 1} \rangle$  65)

**where**  $a +_{\circ\mathfrak{S}} b \equiv \mathfrak{S}(vplus)(a, b)$ .

**notation**  $vplus\text{-app}$  (**infixl**  $\langle +_{\circ 1} \rangle$  65)

**abbreviation**  $vmult\text{-app} :: V \Rightarrow V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle *_{\circ 1} \rangle$  70)

**where**  $a *_{\circ\mathfrak{S}} b \equiv \mathfrak{S}(vmult)(a, b)$ .

**notation**  $vmult\text{-app}$  (**infixl**  $\langle *_{\circ 1} \rangle$  70)

**Simple semiring**

**locale**  $\mathcal{Z}\text{-srng} = \mathcal{Z}\text{-vfsequence } \alpha \ \mathfrak{S} \text{ for } \alpha \ \mathfrak{S} +$   
**assumes**  $\mathcal{Z}\text{-srng-length: } \text{vcard } \mathfrak{S} = 3_{\mathbb{N}}$   
**and**  $\mathcal{Z}\text{-srng-}\mathcal{Z}\text{-csgrp-vplus: } \mathcal{Z}\text{-csgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ}$   
**and**  $\mathcal{Z}\text{-srng-}\mathcal{Z}\text{-sgrp-vmult: } \mathcal{Z}\text{-sgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vmult)]_{\circ}$   
**and**  $\mathcal{Z}\text{-srng-distrib-right:}$   

$$\llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$$

$$(a +_{\circ \mathfrak{S}} b) *_{\circ \mathfrak{S}} c = (a *_{\circ \mathfrak{S}} c) +_{\circ \mathfrak{S}} (b *_{\circ \mathfrak{S}} c)$$
**and**  $\mathcal{Z}\text{-srng-distrib-left:}$   

$$\llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$$

$$a *_{\circ \mathfrak{S}} (b +_{\circ \mathfrak{S}} c) = (a *_{\circ \mathfrak{S}} b) +_{\circ \mathfrak{S}} (a *_{\circ \mathfrak{S}} c)$$
**begin**

**sublocale**  $vplus: \mathcal{Z}\text{-csgrp } \alpha \ \langle [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ} \rangle$   
**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**  $sgrp\text{-app } [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ} = vplus\text{-app } \mathfrak{S}$   
 $\langle \text{proof} \rangle$

**sublocale**  $vmult: \mathcal{Z}\text{-sgrp } \alpha \ \langle [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vmult)]_{\circ} \rangle$   
**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**  $sgrp\text{-app } [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vmult)]_{\circ} = vmult\text{-app } \mathfrak{S}$   
 $\langle \text{proof} \rangle$

**end**

Rules.

**lemma**  $\mathcal{Z}\text{-srngI}[\text{intro}]$ :  
**assumes**  $\mathcal{Z}\text{-vfsequence } \alpha \ \mathfrak{S}$   
**and**  $\text{vcard } \mathfrak{S} = 3_{\mathbb{N}}$   
**and**  $\mathcal{Z}\text{-csgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ}$   
**and**  $\mathcal{Z}\text{-sgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vmult)]_{\circ}$   
**and**  $\wedge a \ b \ c. \llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   

$$(a +_{\circ \mathfrak{S}} b) *_{\circ \mathfrak{S}} c = (a *_{\circ \mathfrak{S}} c) +_{\circ \mathfrak{S}} (b *_{\circ \mathfrak{S}} c)$$
**and**  $\wedge a \ b \ c. \llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   

$$a *_{\circ \mathfrak{S}} (b +_{\circ \mathfrak{S}} c) = (a *_{\circ \mathfrak{S}} b) +_{\circ \mathfrak{S}} (a *_{\circ \mathfrak{S}} c)$$
**shows**  $\mathcal{Z}\text{-srng } \alpha \ \mathfrak{S}$   
 $\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-srngD}[\text{dest}]$ :  
**assumes**  $\mathcal{Z}\text{-srng } \alpha \ \mathfrak{S}$   
**shows**  $\mathcal{Z}\text{-vfsequence } \alpha \ \mathfrak{S}$   
**and**  $\text{vcard } \mathfrak{S} = 3_{\mathbb{N}}$   
**and**  $\mathcal{Z}\text{-csgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ}$   
**and**  $\mathcal{Z}\text{-sgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vmult)]_{\circ}$   
**and**  $\wedge a \ b \ c. \llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   

$$(a +_{\circ \mathfrak{S}} b) *_{\circ \mathfrak{S}} c = (a *_{\circ \mathfrak{S}} c) +_{\circ \mathfrak{S}} (b *_{\circ \mathfrak{S}} c)$$
**and**  $\wedge a \ b \ c. \llbracket a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) \rrbracket \implies$   

$$a *_{\circ \mathfrak{S}} (b +_{\circ \mathfrak{S}} c) = (a *_{\circ \mathfrak{S}} b) +_{\circ \mathfrak{S}} (a *_{\circ \mathfrak{S}} c)$$
 $\langle \text{proof} \rangle$

**lemma**  $\mathcal{Z}\text{-srngE}[\text{elim}]$ :  
**assumes**  $\mathcal{Z}\text{-srng } \alpha \ \mathfrak{S}$   
**obtains**  $\mathcal{Z}\text{-vfsequence } \alpha \ \mathfrak{S}$   
**and**  $\text{vcard } \mathfrak{S} = 3_{\mathbb{N}}$   
**and**  $\mathcal{Z}\text{-csgrp } \alpha \ [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(vplus)]_{\circ}$

**and**  $\mathcal{Z}\text{-sgrp} \alpha [\mathfrak{S}(\mathcal{A}), \mathfrak{S}(\text{vmult})]_{\circ}$   
**and**  $\wedge a b c. [[ a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) ]] \implies$   
 $(a +_{\circ\mathfrak{S}} b) *_{\circ\mathfrak{S}} c = (a *_{\circ\mathfrak{S}} c) +_{\circ\mathfrak{S}} (b *_{\circ\mathfrak{S}} c)$   
**and**  $\wedge a b c. [[ a \in_{\circ} \mathfrak{S}(\mathcal{A}); b \in_{\circ} \mathfrak{S}(\mathcal{A}); c \in_{\circ} \mathfrak{S}(\mathcal{A}) ]] \implies$   
 $a *_{\circ\mathfrak{S}} (b +_{\circ\mathfrak{S}} c) = (a *_{\circ\mathfrak{S}} b) +_{\circ\mathfrak{S}} (a *_{\circ\mathfrak{S}} c)$   
*<proof>*

### 2.16.5 Integer numbers form a semiring

**definition**  $\text{vint-struct} :: V \langle \mathfrak{S}_{\mathbb{Z}} \rangle$   
**where**  $\text{vint-struct} = [\mathbb{Z}_{\circ}, \text{vint-plus}, \text{vint-mult}]_{\circ}$ .

**named-theorems**  $\text{vint-struct-simps}$

**lemma**  $\text{vint-struct-}\mathcal{A}[\text{vint-struct-simps}]: \mathfrak{S}_{\mathbb{Z}}(\mathcal{A}) = \mathbb{Z}_{\circ}$ .  
*<proof>*

**lemma**  $\text{vint-struct-vplus}[\text{vint-struct-simps}]: \mathfrak{S}_{\mathbb{Z}}(\text{vplus}) = \text{vint-plus}$   
*<proof>*

**lemma**  $\text{vint-struct-vmult}[\text{vint-struct-simps}]: \mathfrak{S}_{\mathbb{Z}}(\text{vmult}) = \text{vint-mult}$   
*<proof>*

**context**  $\mathcal{Z}$   
**begin**

**lemma**  $\mathcal{Z}\text{-srng-vint}: \mathcal{Z}\text{-srng} \alpha \mathfrak{S}_{\mathbb{Z}}$   
*<proof>*

Interpretation.

**interpretation**  $\text{vint}: \mathcal{Z}\text{-srng} \alpha \langle \mathfrak{S}_{\mathbb{Z}} \rangle$   
**rewrites**  $\mathfrak{S}_{\mathbb{Z}}(\mathcal{A}) = \mathbb{Z}_{\circ}$   
**and**  $\mathfrak{S}_{\mathbb{Z}}(\text{vplus}) = \text{vint-plus}$   
**and**  $\mathfrak{S}_{\mathbb{Z}}(\text{vmult}) = \text{vint-mult}$   
**and**  $\text{vplus-app} (\mathfrak{S}_{\mathbb{Z}}) = \text{vint-plus-app}$   
**and**  $\text{vmult-app} (\mathfrak{S}_{\mathbb{Z}}) = \text{vint-mult-app}$   
*<proof>*

**thm**  $\text{vint.vmult.}\mathcal{Z}\text{-sgrp-assoc}$

**thm**  $\text{vint.vplus.}\mathcal{Z}\text{-sgrp-assoc}$

**thm**  $\text{vint.}\mathcal{Z}\text{-srng-distrib-left}$

**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  $b' = b \implies a' = a \implies a \sqcup b = b' \sqcup a'$   
 and  $b' = b \implies a' = a \implies a \sqcup b' = b \sqcup a'$   
 and  $b' = b \implies a' = a \implies a' \sqcup b = b' \sqcup a$   
 and  $b' = b \implies a' = a \implies a \sqcup b' = b \sqcup a'$   
 and  $b' = b \implies a' = a \implies a' \sqcup b' = b \sqcup a$   
 <proof>

lemma (in *semilattice-inf*) *inf-commute'*:

**shows**  $b' = b \implies a' = a \implies a \sqcap b = b' \sqcap a'$   
**and**  $b' = b \implies a' = a \implies a \sqcap b' = b \sqcap a'$   
**and**  $b' = b \implies a' = a \implies a' \sqcap b = b' \sqcap a$   
**and**  $b' = b \implies a' = a \implies a \sqcap b' = b \sqcap a'$   
**and**  $b' = b \implies a' = a \implies a' \sqcap b' = b \sqcap a$   
 ⟨*proof*⟩

**lemmas** [ *V-cs-simps* ] =  
*if-P*  
*if-not-P*  
*inf.absorb1*  
*inf.absorb2*  
*sup.absorb1*  
*sup.absorb2*  
*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**  $S \subseteq_o T$   
**shows**  $elts\ S \subseteq elts\ T$   
 ⟨*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**  $x \in list.set\ xs$   
**shows**  $P\ x$   
 ⟨*proof*⟩

**lemma** *list-map-id*:  
**assumes**  $list\ all\ (\lambda x. f\ x = x)\ xs$

**shows**  $\text{map } f \text{ } xs = xs$   
 ⟨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*)  $\text{if3} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$   
**where**  $\text{if3 } a \ b \ c \equiv$   
 (   
    $\lambda i. \text{if } i = 0 \Rightarrow a$   
     |  $i = 1_{\mathbb{N}} \Rightarrow b$   
     | *otherwise*  $\Rightarrow c$   
 )

**lemma** *if3-0*[ *V-cs-simps* ]:  $\text{if3 } a \ b \ c \ 0 = a$  ⟨proof⟩  
**lemma** *if3-1*[ *V-cs-simps* ]:  $\text{if3 } a \ b \ c \ (1_{\mathbb{N}}) = b$  ⟨proof⟩  
**lemma** *if3-2*[ *V-cs-simps* ]:  $\text{if3 } a \ b \ c \ (2_{\mathbb{N}}) = c$  ⟨proof⟩

**lemma** *insertI1'*:  
**assumes**  $x' = x$   
**shows**  $x \in_{\circ} \text{insert } x' \ A$   
 ⟨proof⟩

**lemma** *in-vsingleton*[ *V-cs-intros* ]:  
**assumes**  $f = a$   
**shows**  $f \in_{\circ} \text{set } \{a\}$   
 ⟨proof⟩

**lemma** *a-in-succ-a*:  $a \in_{\circ} \text{succ } a$  ⟨proof⟩

**lemma** *a-in-succ-xI*:  
**assumes**  $a \in_{\circ} x$   
**shows**  $a \in_{\circ} \text{succ } x$   
 ⟨proof⟩

**lemma** *vone-ne*[ *V-cs-intros* ]:  $1_{\mathbb{N}} \neq 0$  ⟨proof⟩

**lemmas** [ *V-cs-simps* ] =  
*vinset-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*  
*vinsertI1'*  
*vinsertI2*  
*vfinite-vinsert*  
*vfinite-vsingleton*  
*vdisjnt-nin-right*  
*vdisjnt-nin-left*  
*vunionI1*  
*vunionI2*  
*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**  $\mathcal{R}_{\circ} (vconst-on A a) \subseteq_{\circ} R$   
*<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-vrestriction*  
*vdomain-vrestriction-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-vsν*

**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* ] =  
*ftimesI2*  
*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*  $\alpha$  **and** *Ord*  $\beta$  **and**  $\alpha \subseteq_{\circ} \beta$   
**shows**  $\alpha \subseteq_{\circ} \text{succ } \beta$   
*<proof>*

**lemma** *Ord-in-Vset-succ*[ *V-cs-intros* ]:  
**assumes** *Ord*  $\alpha$  **and**  $a \in_{\circ} \text{Vset } \alpha$   
**shows**  $a \in_{\circ} \text{Vset } (\text{succ } \alpha)$   
*<proof>*

**lemma** *Ord-vsubset-Vset-succ*[*V-cs-intros*]:  
**assumes** *Ord*  $\alpha$  **and**  $B \subseteq_{\circ} Vset\ \alpha$   
**shows**  $B \subseteq_{\circ} Vset\ (succ\ \alpha)$   
*<proof>*

**lemmas** (in  $\mathcal{Z}$ ) [*V-cs-intros*] =  
*omega-in- $\alpha$*   
*Ord- $\alpha$*   
*Limit- $\alpha$*

**lemmas** [*V-cs-intros*] =  
*vempty-in-Vset-succ*  
*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**  $a = \text{numeral } b$   
**shows**  $(0::\text{nat}) < a$   
*<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**  $\mathcal{Z}$ )  $\mathcal{Z}$ - $\beta$ :  
**assumes**  $\beta = \alpha$   
**shows**  $\mathcal{Z} \beta$   
*<proof>*

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

## 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  $f : a \mapsto_{\mathcal{C}} b$  ( $b$  can be assumed to be arbitrary) instead of  $f \in_{\circ} \mathcal{C}(Arr)$  and  $\mathcal{C}(Dom)(f) = a$ .

**definition** *is-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

**where** *is-arr*  $\mathcal{C} a b f \longleftrightarrow f \in_{\circ} \mathcal{C}(Arr) \wedge \mathcal{C}(Dom)(f) = a \wedge \mathcal{C}(Cod)(f) = b$

**syntax** *-is-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$  ( $\langle \cdot : \cdot \mapsto_1 \cdot \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-arr*  $\equiv$  *is-arr*

**translations**  $f : a \mapsto_{\mathcal{C}} b \equiv CONST$  *is-arr*  $\mathcal{C} a b f$

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*  $\mathcal{C} a b \equiv set \{f. f : a \mapsto_{\mathcal{C}} b\}$

**lemma** *small-Hom*[*simp*]: *small*  $\{f. f : a \mapsto_{\mathcal{C}} b\}$   $\langle proof \rangle$

Rules.

**lemma** *HomI*[*dg-shared-cs-intros*, *dg-cs-intros*]:

**assumes**  $f : a \mapsto_{\mathcal{C}} b$   
**shows**  $f \in_{\circ} Hom \mathcal{C} a b$   
 $\langle proof \rangle$

**lemma** *in-Hom-iff*[*dg-shared-cs-simps*, *dg-cs-simps*]:

$f \in_{\circ} Hom \mathcal{C} a b \longleftrightarrow f : a \mapsto_{\mathcal{C}} b$   
 $\langle proof \rangle$

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**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$

**and**  $a' \in_{\circ} \mathfrak{C}(\text{Obj})$

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

**and**  $b' \in_{\circ} \mathfrak{C}(\text{Obj})$

**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  $\alpha$ , such that  $\omega < \alpha$ ; the set of objects is assumed to be a subset of the set  $V_{\alpha}$  in the von Neumann hierarchy of sets (e.g., see [59]). Such digraphs are called  $\alpha$ -*digraphs* to make the dependence on the parameter  $\alpha$  explicit.<sup>1</sup> 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  $\alpha$ -digraph is a tiny  $\alpha$ -digraph if and only if the set of its objects and the set of its arrows both belong to the set  $V_{\alpha}$ . 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  $V_{\alpha}$  instead of the universe  $U$ . Also, in what follows, any member of the set  $V_{\alpha}$  will be referred to as an  $\alpha$ -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  $V_{\alpha}$ .

### 3.2.5 Digraph: definition and elementary properties

**locale** *digraph* =  $\mathcal{Z}$   $\alpha$  + *vfsequence*  $\mathfrak{C}$  + *Dom*: *vsv*  $\langle \mathfrak{C}(\text{Dom}) \rangle$  + *Cod*: *vsv*  $\langle \mathfrak{C}(\text{Cod}) \rangle$

**for**  $\alpha$   $\mathfrak{C}$  +

**assumes** *dg-length*[*dg-cs-simps*]: *vcard*  $\mathfrak{C}$  =  $4_{\mathbb{N}}$

**and** *dg-Dom-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_{\circ}(\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$

**and** *dg-Dom-vrange*:  $\mathcal{R}_{\circ}(\mathfrak{C}(\text{Dom})) \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

**and** *dg-Cod-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_{\circ}(\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$

**and** *dg-Cod-vrange*:  $\mathcal{R}_{\circ}(\mathfrak{C}(\text{Cod})) \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

**and** *dg-Obj-vsubset-Vset*:  $\mathfrak{C}(\text{Obj}) \subseteq_{\circ} \text{Vset } \alpha$

**and** *dg-Hom-vifunion-in-Vset*[*dg-cs-intros*]:

$\llbracket A \subseteq_{\circ} \mathfrak{C}(\text{Obj}); B \subseteq_{\circ} \mathfrak{C}(\text{Obj}); A \in_{\circ} \text{Vset } \alpha; B \in_{\circ} \text{Vset } \alpha \rrbracket \implies$

$(\bigcup_{\circ} a \in_{\circ} A. \bigcup_{\circ} b \in_{\circ} B. \text{Hom } \mathfrak{C} a b) \in_{\circ} \text{Vset } \alpha$

<sup>1</sup>The prefix " $\alpha$ -" may be omitted whenever it is possible to infer the value of  $\alpha$  from the context. This applies not only to the digraphs, but all other entities that are parameterized by a limit ordinal  $\alpha$  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*  $\alpha' \mathfrak{C}$   
*<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*  $\alpha \mathfrak{A}$   
**and** *digraph*  $\alpha \mathfrak{B}$   
**and**  $\mathfrak{A}(\text{Obj}) = \mathfrak{B}(\text{Obj})$   
**and**  $\mathfrak{A}(\text{Arr}) = \mathfrak{B}(\text{Arr})$   
**and**  $\mathfrak{A}(\text{Dom}) = \mathfrak{B}(\text{Dom})$   
**and**  $\mathfrak{A}(\text{Cod}) = \mathfrak{B}(\text{Cod})$   
**shows**  $\mathfrak{A} = \mathfrak{B}$   
*<proof>*

**lemma** (in *digraph*) *dg-def*:  $\mathfrak{C} = [\mathfrak{C}(\text{Obj}), \mathfrak{C}(\text{Arr}), \mathfrak{C}(\text{Dom}), \mathfrak{C}(\text{Cod})]$ .  
*<proof>*

**lemma** (in *digraph*) *dg-Obj-if-Dom-vrange*:  
**assumes**  $a \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{C}(\text{Dom}))$   
**shows**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$   
*<proof>*

**lemma** (in *digraph*) *dg-Obj-if-Cod-vrange*:  
**assumes**  $a \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{C}(\text{Cod}))$   
**shows**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$   
*<proof>*

**lemma** (in *digraph*) *dg-is-arrD*:  
**assumes**  $f : a \mapsto_{\mathfrak{C}} b$   
**shows**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$   
**and**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$   
**and**  $b \in_{\circ} \mathfrak{C}(\text{Obj})$   
**and**  $\mathfrak{C}(\text{Dom})(f) = a$   
**and**  $\mathfrak{C}(\text{Cod})(f) = b$   
*<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**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$   
**and**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$

**and**  $b \in_{\circ} \mathfrak{C}(\text{Obj})$   
**and**  $\mathfrak{C}(\text{Dom})(f) = a$   
**and**  $\mathfrak{C}(\text{Cod})(f) = b$   
 ⟨proof⟩

**lemma** (in *digraph*) *dg-in-ArrE[elim]*:  
**assumes**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$   
**obtains**  $a \ b$  where  $f : a \mapsto_{\mathfrak{C}} b$  **and**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$  **and**  $b \in_{\circ} \mathfrak{C}(\text{Obj})$   
 ⟨proof⟩

**lemma** (in *digraph*) *dg-Hom-in-Vset[dg-cs-intros]*:  
**assumes**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$  **and**  $b \in_{\circ} \mathfrak{C}(\text{Obj})$   
**shows**  $\text{Hom } \mathfrak{C} \ a \ b \in_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemmas** [*dg-cs-intros*] = *digraph.dg-Hom-in-Vset*

Size.

**lemma** (in *digraph*) *dg-Arr-vsubset-Vset*:  $\mathfrak{C}(\text{Arr}) \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *digraph*) *dg-Dom-vsubset-Vset*:  $\mathfrak{C}(\text{Dom}) \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *digraph*) *dg-Cod-vsubset-Vset*:  $\mathfrak{C}(\text{Cod}) \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *digraph*) *dg-digraph-in-Vset-4*:  $\mathfrak{C} \in_{\circ} \text{Vset } (\alpha + 4_{\mathbf{N}})$   
 ⟨proof⟩

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

**lemma** (in *digraph*) *dg-in-Obj-in-Vset[dg-cs-intros]*:  
**assumes**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$   
**shows**  $a \in_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

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

**lemma** (in *digraph*) *dg-in-Arr-in-Vset[dg-cs-intros]*:  
**assumes**  $a \in_{\circ} \mathfrak{C}(\text{Arr})$   
**shows**  $a \in_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

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

**lemma** (in *digraph*) *dg-Cod-in-Vset*:  
**assumes**  $\mathcal{Z} \ \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{C}(\text{Cod}) \in_{\circ} \text{Vset } \beta$

*<proof>*

**lemma** (in *digraph*) *dg-in-Vset*:

**assumes**  $Z \beta$  **and**  $\alpha \in_{\circ} \beta$

**shows**  $\mathfrak{C} \in_{\circ} Vset \beta$

*<proof>*

**lemma** (in *digraph*) *dg-digraph-if-ge-Limit*:

**assumes**  $Z \beta$  **and**  $\alpha \in_{\circ} \beta$

**shows** *digraph*  $\beta \mathfrak{C}$

*<proof>*

**lemma** *small-digraph[simp]*: *small*  $\{\mathfrak{C}. \text{digraph } \alpha \mathfrak{C}\}$

*<proof>*

**lemma** (in  $Z$ ) *digraphs-in-Vset*:

**assumes**  $Z \beta$  **and**  $\alpha \in_{\circ} \beta$

**shows** *set*  $\{\mathfrak{C}. \text{digraph } \alpha \mathfrak{C}\} \in_{\circ} Vset \beta$

*<proof>*

**lemma** *digraph-if-digraph*:

**assumes** *digraph*  $\beta \mathfrak{C}$

**and**  $Z \alpha$

**and**  $\mathfrak{C}(\text{Obj}) \subseteq_{\circ} Vset \alpha$

**and**  $\bigwedge A B. [\![ A \subseteq_{\circ} \mathfrak{C}(\text{Obj}); B \subseteq_{\circ} \mathfrak{C}(\text{Obj}); A \in_{\circ} Vset \alpha; B \in_{\circ} Vset \alpha \]\!] \implies$

$(\bigcup_{a \in_{\circ} A} a. \bigcup_{b \in_{\circ} B} b. \text{Hom } \mathfrak{C} a b) \in_{\circ} Vset \alpha$

**shows** *digraph*  $\alpha \mathfrak{C}$

*<proof>*

Further properties.

**lemma** (in *digraph*) *dg-Dom-app-in-Obj*:

**assumes**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$

**shows**  $\mathfrak{C}(\text{Dom})(f) \in_{\circ} \mathfrak{C}(\text{Obj})$

*<proof>*

**lemma** (in *digraph*) *dg-Cod-app-in-Obj*:

**assumes**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$

**shows**  $\mathfrak{C}(\text{Cod})(f) \in_{\circ} \mathfrak{C}(\text{Obj})$

*<proof>*

**lemma** (in *digraph*) *dg-Arr-venempty-if-Obj-venempty*:

**assumes**  $\mathfrak{C}(\text{Obj}) = 0$

**shows**  $\mathfrak{C}(\text{Arr}) = 0$

*<proof>*

**lemma** (in *digraph*) *dg-Dom-venempty-if-Arr-venempty*:

**assumes**  $\mathfrak{C}(\text{Arr}) = 0$

**shows**  $\mathfrak{C}(\text{Dom}) = 0$

*<proof>*

**lemma** (in *digraph*) *dg-Cod-venempty-if-Arr-venempty*:

**assumes**  $\mathfrak{C}(\text{Arr}) = 0$

**shows**  $\mathfrak{C}(\text{Cod}) = 0$

*<proof>*

### 3.2.6 Opposite digraph

#### Definition and elementary properties

See Chapter II-2 in [39].

**definition**  $op-dg :: V \Rightarrow V$   
**where**  $op-dg \mathfrak{C} = [\mathfrak{C}(\backslash Obj), \mathfrak{C}(\backslash Arr), \mathfrak{C}(\backslash Cod), \mathfrak{C}(\backslash Dom)]$ .

Components.

**lemma**  $op-dg-components[dg-op-simps]$ :

**shows**  $op-dg \mathfrak{C}(\backslash Obj) = \mathfrak{C}(\backslash Obj)$   
**and**  $op-dg \mathfrak{C}(\backslash Arr) = \mathfrak{C}(\backslash Arr)$   
**and**  $op-dg \mathfrak{C}(\backslash Dom) = \mathfrak{C}(\backslash Cod)$   
**and**  $op-dg \mathfrak{C}(\backslash Cod) = \mathfrak{C}(\backslash Dom)$   
 $\langle proof \rangle$

**lemma**  $op-dg-component-intros[dg-op-intros]$ :

**shows**  $a \in_{\circ} \mathfrak{C}(\backslash Obj) \implies a \in_{\circ} op-dg \mathfrak{C}(\backslash Obj)$   
**and**  $f \in_{\circ} \mathfrak{C}(\backslash Arr) \implies f \in_{\circ} op-dg \mathfrak{C}(\backslash Arr)$   
 $\langle proof \rangle$

Elementary properties.

**lemma**  $op-dg-is-arr[dg-op-simps]$ :  $f : b \mapsto_{op-dg \mathfrak{C}} a \iff f : a \mapsto_{\mathfrak{C}} b$   
 $\langle proof \rangle$

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

**lemma**  $op-dg-Hom[dg-op-simps]$ :  $Hom (op-dg \mathfrak{C}) a b = Hom \mathfrak{C} b a$   
 $\langle proof \rangle$

#### Further properties

**lemma** (**in** *digraph*)  $digraph-op[dg-op-intros]$ :  $digraph \alpha (op-dg \mathfrak{C})$   
 $\langle proof \rangle$

**lemmas**  $digraph-op[dg-op-intros] = digraph.digraph-op$

**lemma** (**in** *digraph*)  $dg-op-dg-op-dg[dg-op-simps]$ :  $op-dg (op-dg \mathfrak{C}) = \mathfrak{C}$   
 $\langle proof \rangle$

**lemmas**  $dg-op-dg-op-dg[dg-op-simps] = digraph.dg-op-dg-op-dg$

**lemma**  $eq-op-dg-iff[dg-op-simps]$ :

**assumes**  $digraph \alpha \mathfrak{A}$  **and**  $digraph \alpha \mathfrak{B}$   
**shows**  $op-dg \mathfrak{A} = op-dg \mathfrak{B} \iff \mathfrak{A} = \mathfrak{B}$   
 $\langle proof \rangle$

### 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* =  $Z$   $\alpha$  + *vfsequence*  $\mathfrak{C}$  + *Dom*: *vsu*  $\langle \mathfrak{C}(\text{Dom}) \rangle$  + *Cod*: *vsu*  $\langle \mathfrak{C}(\text{Cod}) \rangle$

for  $\alpha$   $\mathfrak{C}$  +

assumes *tiny-dg-length*[*dg-cs-simps*]: *vcard*  $\mathfrak{C} = 4_{\mathbb{N}}$   
 and *tiny-dg-Dom-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_o(\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$   
 and *tiny-dg-Dom-vrange*:  $\mathcal{R}_o(\mathfrak{C}(\text{Dom})) \subseteq_o \mathfrak{C}(\text{Obj})$   
 and *tiny-dg-Cod-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_o(\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$   
 and *tiny-dg-Cod-vrange*:  $\mathcal{R}_o(\mathfrak{C}(\text{Cod})) \subseteq_o \mathfrak{C}(\text{Obj})$   
 and *tiny-dg-Obj-in-Vset*[*dg-small-cs-intros*]:  $\mathfrak{C}(\text{Obj}) \in_o \text{Vset } \alpha$   
 and *tiny-dg-Arr-in-Vset*[*dg-small-cs-intros*]:  $\mathfrak{C}(\text{Arr}) \in_o \text{Vset } \alpha$

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*  $\alpha' \mathfrak{C}$   
 ⟨*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*  $\alpha$   $\mathfrak{C}$  and  $\mathfrak{C}(\text{Obj}) \in_o \text{Vset } \alpha$  and  $\mathfrak{C}(\text{Arr}) \in_o \text{Vset } \alpha$   
 shows *tiny-digraph*  $\alpha \mathfrak{C}$   
 ⟨*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*:  $\mathfrak{C}(\text{Dom}) \in_o \text{Vset } \alpha$

⟨*proof*⟩

lemma (in *tiny-digraph*) *tiny-dg-Cod-in-Vset*:  $\mathfrak{C}(\text{Cod}) \in_o \text{Vset } \alpha$

⟨*proof*⟩

lemma (in *tiny-digraph*) *tiny-dg-in-Vset*:  $\mathfrak{C} \in_o \text{Vset } \alpha$

⟨*proof*⟩

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

**lemma** *tiny-digraphs-ubset-Vset*: *set*  $\{\mathfrak{C}. \text{tiny-digraph } \alpha \ \mathfrak{C}\} \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨*proof*⟩

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

### Opposite tiny digraph

**lemma** (in *tiny-digraph*) *tiny-digraph-op*: *tiny-digraph*  $\alpha \ (\text{op-dg } \mathfrak{C})$   
 ⟨*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]<sup>2</sup>.

**locale** *finite-digraph* = *digraph*  $\alpha \ \mathfrak{C}$  **for**  $\alpha \ \mathfrak{C} +$   
**assumes** *fin-dg-Obj-vfinite[dg-small-cs-intros]*: *vfinite*  $(\mathfrak{C}(\text{Obj}))$   
**and** *fin-dg-Arr-vfinite[dg-small-cs-intros]*: *vfinite*  $(\mathfrak{C}(\text{Arr}))$

**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*  $\alpha' \ \mathfrak{C}$   
 ⟨*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*  $\subseteq$  *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*  $\{\mathfrak{C}. \text{finite-digraph } \alpha \ \mathfrak{C}\}$   
 ⟨*proof*⟩

<sup>2</sup><https://ncatlab.org/nlab/show/finite+category>

**lemma** *finite-digraphs-vsubset-Vset*: set  $\{\mathfrak{C}. \text{finite-digraph } \alpha \ \mathfrak{C}\} \sqsubseteq_0 \text{Vset } \alpha$   
 ⟨proof⟩

### Opposite finite digraph

**lemma** (in *finite-digraph*) *fininte-digraph-op*: *finite-digraph*  $\alpha$  (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  $\alpha$  such that  $\omega < \alpha$ . Such digraph homomorphisms are referred to either as  $\alpha$ -digraph homomorphisms or homomorphisms of  $\alpha$ -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* =

$Z \alpha + \text{vsequence } \mathfrak{F} + \text{HomDom: digraph } \alpha \mathfrak{A} + \text{HomCod: digraph } \alpha \mathfrak{B}$

**for**  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$

**assumes** *dghm-length*[*dg-cs-simps*]: *vcard*  $\mathfrak{F} = 4_{\mathbb{N}}$

**and** *dghm-HomDom*[*dg-cs-simps*]:  $\mathfrak{F}(\text{HomDom}) = \mathfrak{A}$

**and** *dghm-HomCod*[*dg-cs-simps*]:  $\mathfrak{F}(\text{HomCod}) = \mathfrak{B}$

**and** *dghm-ObjMap-vs*: *vs* ( $\mathfrak{F}(\text{ObjMap})$ )

**and** *dghm-ArrMap-vs*: *vs* ( $\mathfrak{F}(\text{ArrMap})$ )

**and** *dghm-ObjMap-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_{\circ}(\mathfrak{F}(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$

**and** *dghm-ObjMap-vrange*:  $\mathcal{R}_{\circ}(\mathfrak{F}(\text{ObjMap})) \subseteq_{\circ} \mathfrak{B}(\text{Obj})$

**and** *dghm-ArrMap-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_{\circ}(\mathfrak{F}(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$

**and** *dghm-ArrMap-is-arr*:

$f : a \mapsto_{\mathfrak{A}} b \implies \mathfrak{F}(\text{ArrMap})(f) : \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(b)$

**syntax** *-is-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle\langle - / - \mapsto_{DG^1} - \rangle\rangle$  [51, 51, 51] 51

**syntax-consts** *-is-dghm*  $\equiv$  *is-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG^{\alpha}} \mathfrak{B} \equiv \text{CONST } \textit{is-dghm } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-cn-dghm*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \equiv \mathfrak{F} : \textit{op-dg } \mathfrak{A} \mapsto_{DG^{\alpha}} \mathfrak{B}$

**syntax** *-is-cn-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle\langle - / -_{DG} \mapsto \mapsto 1 - \rangle\rangle$  [51, 51, 51] 51

**syntax-consts**  $-is-cn-dghm \rightleftharpoons is-cn-dghm$

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{DG} \mapsto \alpha \text{ } \mathfrak{B} \rightarrow CONST \text{ } is-cn-dghm \text{ } \alpha \text{ } \mathfrak{A} \text{ } \mathfrak{B} \text{ } \mathfrak{F}$

**abbreviation**  $all-dghms :: V \Rightarrow V$

**where**  $all-dghms \alpha \equiv set \{ \mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \text{ }_{DG} \alpha \text{ } \mathfrak{B} \}$

**abbreviation**  $dghms :: V \Rightarrow V \Rightarrow V \Rightarrow V$

**where**  $dghms \alpha \mathfrak{A} \mathfrak{B} \equiv set \{ \mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \text{ }_{DG} \alpha \text{ } \mathfrak{B} \}$

**sublocale**  $is-dghm \subseteq ObjMap: vsv \langle \mathfrak{F}(\text{ } ObjMap) \rangle$

**rewrites**  $\mathcal{D}_o. (\mathfrak{F}(\text{ } ObjMap)) = \mathfrak{A}(\text{ } Obj)$

$\langle proof \rangle$

**sublocale**  $is-dghm \subseteq ArrMap: vsv \langle \mathfrak{F}(\text{ } ArrMap) \rangle$

**rewrites**  $\mathcal{D}_o. (\mathfrak{F}(\text{ } ArrMap)) = \mathfrak{A}(\text{ } Arr)$

$\langle proof \rangle$

**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**  $\mathfrak{F} = \mathfrak{F}(\text{ } ArrMap)(\text{ } f)$

**shows**  $\mathfrak{F} : \mathfrak{F}(\text{ } ObjMap)(\text{ } a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ } ObjMap)(\text{ } b)$

$\langle proof \rangle$

**lemma** **(in**  $is-dghm$ )  $dghm-ArrMap-is-arr'$   $[dg-cs-intros]$ :

**assumes**  $f : a \mapsto_{\mathfrak{A}} b$

**and**  $A = \mathfrak{F}(\text{ } ObjMap)(\text{ } a)$

**and**  $B = \mathfrak{F}(\text{ } ObjMap)(\text{ } b)$

**shows**  $\mathfrak{F}(\text{ } ArrMap)(\text{ } f) : A \mapsto_{\mathfrak{B}} B$

$\langle proof \rangle$

**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**  $\mathfrak{F} : \mathfrak{A}' \mapsto \text{ }_{DG} \alpha' \text{ } \mathfrak{B}'$

$\langle proof \rangle$

**mk-ide rf**  $is-dghm-def[un\text{ } folded \text{ } is-dghm-axioms-def]$

$|intro \text{ } is-dghmI|$

$|dest \text{ } is-dghmD[dest]|$

$|elim \text{ } is-dghmE[elim]|$

**lemmas**  $[dg-cs-intros] = is-dghmD(3,4)$

Elementary properties.

**lemma**  $dghm-eqI$ :

**assumes**  $\mathfrak{G} : \mathfrak{A} \mapsto \text{ }_{DG} \alpha \text{ } \mathfrak{B}$

**and**  $\mathfrak{F} : \mathfrak{C} \mapsto \text{ }_{DG} \alpha \text{ } \mathfrak{D}$

**and**  $\mathfrak{G}(\text{ } ObjMap) = \mathfrak{F}(\text{ } ObjMap)$

**and**  $\mathfrak{G}(\text{ } ArrMap) = \mathfrak{F}(\text{ } ArrMap)$

**and**  $\mathfrak{A} = \mathfrak{C}$

**and**  $\mathfrak{B} = \mathfrak{D}$

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

**lemma** (in *is-dghm*) *dghm-def*:  $\mathfrak{F} = [\mathfrak{F}(\text{ObjMap}), \mathfrak{F}(\text{ArrMap}), \mathfrak{F}(\text{HomDom}), \mathfrak{F}(\text{HomCod})]$ .  
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-ObjMap-app-in-HomCod-Obj*[*dg-cs-intros*]:  
**assumes**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$   
**shows**  $\mathfrak{F}(\text{ObjMap})(a) \in_{\circ} \mathfrak{B}(\text{Obj})$   
 ⟨proof⟩

**lemmas** [*dg-cs-intros*] = *is-dghm.dghm-ObjMap-app-in-HomCod-Obj*

**lemma** (in *is-dghm*) *dghm-ArrMap-vrange*:  $\mathcal{R}_{\circ}(\mathfrak{F}(\text{ArrMap})) \subseteq_{\circ} \mathfrak{B}(\text{Arr})$   
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-ArrMap-app-in-HomCod-Arr*[*dg-cs-intros*]:  
**assumes**  $a \in_{\circ} \mathfrak{A}(\text{Arr})$   
**shows**  $\mathfrak{F}(\text{ArrMap})(a) \in_{\circ} \mathfrak{B}(\text{Arr})$   
 ⟨proof⟩

**lemmas** [*dg-cs-intros*] = *is-dghm.dghm-ArrMap-app-in-HomCod-Arr*

Size.

**lemma** (in *is-dghm*) *dghm-ObjMap-ubset-Vset*:  $\mathfrak{F}(\text{ObjMap}) \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-ArrMap-ubset-Vset*:  $\mathfrak{F}(\text{ArrMap}) \subseteq_{\circ} \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-ObjMap-in-Vset*:  
**assumes**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{F}(\text{ObjMap}) \in_{\circ} \text{Vset } \beta$   
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-ArrMap-in-Vset*:  
**assumes**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{F}(\text{ArrMap}) \in_{\circ} \text{Vset } \beta$   
 ⟨proof⟩

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

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

**lemma** *small-all-dghms[simp]*: *small*  $\{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto_{DG} \mathfrak{B}\}$   
 ⟨proof⟩

**lemma** (in *is-dghm*) *dghm-in-Vset-7*:  $\mathfrak{F} \in_{\circ} \text{Vset } (\alpha + 7_{\mathbb{N}})$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *all-dghms-in-Vset*:  
**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$

**shows** *all-dghms*  $\alpha \in_0 Vset \beta$   
 ⟨*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**  $\mathfrak{F}(\downarrow ArrMap)(\downarrow f) \in_0 \mathfrak{B}(\downarrow Arr)$   $\mathfrak{F}(\downarrow ObjMap)(\downarrow a) \in_0 \mathfrak{B}(\downarrow Obj)$   $\mathfrak{F}(\downarrow ObjMap)(\downarrow b) \in_0 \mathfrak{B}(\downarrow Obj)$   
 ⟨*proof*⟩

**lemma** (in *is-dghm*) *dghm-vimage-dghm-ArrMap-uset-Hom*:  
**assumes**  $a \in_0 \mathfrak{A}(\downarrow Obj)$  **and**  $b \in_0 \mathfrak{A}(\downarrow Obj)$   
**shows**  $\mathfrak{F}(\downarrow ArrMap) \circ Hom \mathfrak{A} a b \subseteq_0 Hom \mathfrak{B} (\mathfrak{F}(\downarrow ObjMap)(\downarrow a)) (\mathfrak{F}(\downarrow ObjMap)(\downarrow b))$   
 ⟨*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*  $\mathfrak{F} =$   
 $[\mathfrak{F}(\downarrow ObjMap), \mathfrak{F}(\downarrow ArrMap), op-dg (\mathfrak{F}(\downarrow HomDom)), op-dg (\mathfrak{F}(\downarrow HomCod))].$

Components.

**lemma** *op-dghm-components[dg-op-simps]*:  
**shows** *op-dghm*  $\mathfrak{F}(\downarrow ObjMap) = \mathfrak{F}(\downarrow ObjMap)$   
**and** *op-dghm*  $\mathfrak{F}(\downarrow ArrMap) = \mathfrak{F}(\downarrow ArrMap)$   
**and** *op-dghm*  $\mathfrak{F}(\downarrow HomDom) = op-dg (\mathfrak{F}(\downarrow HomDom))$   
**and** *op-dghm*  $\mathfrak{F}(\downarrow HomCod) = op-dg (\mathfrak{F}(\downarrow HomCod))$   
 ⟨*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**  $\mathfrak{A}' = op-dg \mathfrak{A}$  **and**  $\mathfrak{B}' = op-dg \mathfrak{B}$  **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*  $\mathfrak{G} = op-dghm \mathfrak{F} \iff \mathfrak{G} = \mathfrak{F}$   
 ⟨*proof*⟩

### 3.4.4 Composition of covariant digraph homomorphisms

#### Definition and elementary properties

See Chapter I-3 in [39].

**definition**  $dg\text{hm-comp} :: V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{DG\text{HM}} \rangle$  55)

**where**  $\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F} =$   
 $[\mathfrak{G}(\text{ObjMap}) \circ \mathfrak{F}(\text{ObjMap}), \mathfrak{G}(\text{ArrMap}) \circ \mathfrak{F}(\text{ArrMap}), \mathfrak{F}(\text{HomDom}), \mathfrak{G}(\text{HomCod})].$

Components.

**lemma**  $dg\text{hm-comp-components}$ :

**shows**  $(\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ObjMap}) = \mathfrak{G}(\text{ObjMap}) \circ \mathfrak{F}(\text{ObjMap})$   
**and**  $(\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ArrMap}) = \mathfrak{G}(\text{ArrMap}) \circ \mathfrak{F}(\text{ArrMap})$   
**and**  $[dg\text{-shared-cs-simps}, dg\text{-cs-simps}]: (\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{HomDom}) = \mathfrak{F}(\text{HomDom})$   
**and**  $[dg\text{-shared-cs-simps}, dg\text{-cs-simps}]: (\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{HomCod}) = \mathfrak{G}(\text{HomCod})$   
 $\langle \text{proof} \rangle$

#### Object map

**lemma**  $dg\text{hm-comp-ObjMap-vsuv}[dg\text{-cs-intros}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $vsu ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ObjMap}))$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ObjMap-vdomain}[dg\text{-cs-simps}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $\mathcal{D}_o ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ObjMap-vrange}$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $\mathcal{R}_o ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ObjMap})) \subseteq_o \mathfrak{C}(\text{Obj})$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ObjMap-app}[dg\text{-cs-simps}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$  **and**  $a \in_o \mathfrak{A}(\text{Obj})$   
**shows**  $(\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ObjMap})(a) = \mathfrak{G}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(a))$

$\langle \text{proof} \rangle$

#### Arrow map

**lemma**  $dg\text{hm-comp-ArrMap-vsuv}[dg\text{-cs-intros}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $vsu ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ArrMap}))$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ArrMap-vdomain}[dg\text{-cs-simps}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $\mathcal{D}_o ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ArrMap-vrange}[dg\text{-cs-intros}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $\mathcal{R}_o ((\mathfrak{G} \circ_{DG\text{HM}} \mathfrak{F})(\text{ArrMap})) \subseteq_o \mathfrak{C}(\text{Arr})$

$\langle \text{proof} \rangle$

**lemma**  $dg\text{hm-comp-ArrMap-app}[dg\text{-cs-simps}]$ :

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$  **and**  $f \in_o \mathfrak{A}(\text{Arr})$

**shows**  $(\mathfrak{G} \circ_{DGHM} \mathfrak{F})(\text{ArrMap})(f) = \mathfrak{G}(\text{ArrMap})(\mathfrak{F}(\text{ArrMap})(f))$   
 ⟨proof⟩

### Opposite of the composition of covariant digraph homomorphisms

**lemma** *op-dghm-dghm-comp*[*dg-op-simps*]:  
 $op\text{-dghm}(\mathfrak{G} \circ_{DGHM} \mathfrak{F}) = op\text{-dghm} \mathfrak{G} \circ_{DGHM} op\text{-dghm} \mathfrak{F}$   
 ⟨proof⟩

### Further properties

**lemma** *dghm-comp-is-dghm*[*dg-cs-intros*]:  
**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{C}$   
 ⟨proof⟩

**lemma** *dghm-comp-assoc*[*dg-cs-simps*]:  
**assumes**  $\mathfrak{H} : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{D}$  **and**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**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**  $\langle_{DGHM^\circ}$  55)  
**where**  $\mathfrak{G}_{DGHM^\circ} \mathfrak{F} =$   
 [   
 $\mathfrak{G}(\text{ObjMap}) \circ \mathfrak{F}(\text{ObjMap}),$   
 $\mathfrak{G}(\text{ArrMap}) \circ \mathfrak{F}(\text{ArrMap}),$   
 $op\text{-dg}(\mathfrak{F}(\text{HomDom})),$   
 $\mathfrak{G}(\text{HomCod})$   
 ]<sub>o</sub>

Components.

**lemma** *dghm-cn-comp-components*:  
**shows**  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap}) = \mathfrak{G}(\text{ObjMap}) \circ \mathfrak{F}(\text{ObjMap})$   
**and**  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap}) = \mathfrak{G}(\text{ArrMap}) \circ \mathfrak{F}(\text{ArrMap})$   
**and** [*dg-cn-cs-simps*]:  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{HomDom}) = op\text{-dg}(\mathfrak{F}(\text{HomDom}))$   
**and** [*dg-cn-cs-simps*]:  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{HomCod}) = \mathfrak{G}(\text{HomCod})$   
 ⟨proof⟩

### Object map: two contravariant digraph homomorphisms

**lemma** *dghm-cn-comp-ObjMap-vsuv*[*dg-cn-cs-intros*]:  
**assumes**  $\mathfrak{G} : \mathfrak{B}_{DG\mapsto\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A}_{DG\mapsto\alpha} \mathfrak{B}$   
**shows** *vsu*  $((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap}))$   
 ⟨proof⟩

**lemma** *dghm-cn-comp-ObjMap-vdomain*[*dg-cn-cs-simps*]:  
**assumes**  $\mathfrak{G} : \mathfrak{B}_{DG\mapsto\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A}_{DG\mapsto\alpha} \mathfrak{B}$   
**shows**  $\mathcal{D}_o((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$   
 ⟨proof⟩

**lemma** *dghm-cn-comp-ObjMap-vrange*:  
**assumes**  $\mathfrak{G} : \mathfrak{B}_{DG\mapsto\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A}_{DG\mapsto\alpha} \mathfrak{B}$   
**shows**  $\mathcal{R}_o((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})) \subseteq \mathfrak{C}(\text{Obj})$

*<proof>*

**lemma** *dghm-cn-comp-ObjMap-app[dg-cn-cs-simps]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{DG \mapsto \alpha} \mathfrak{B}$  and  $a \in \mathfrak{A}(\text{Obj})$   
 shows  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})(a) = \mathfrak{G}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(a))$   
*<proof>*

### Arrow map: two contravariant digraph homomorphisms

**lemma** *dghm-cn-comp-ArrMap-vsuv[dg-cn-cs-intros]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{DG \mapsto \alpha} \mathfrak{B}$   
 shows *vsu*  $((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap}))$   
*<proof>*

**lemma** *dghm-cn-comp-ArrMap-vdomain[dg-cs-simps]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{DG \mapsto \alpha} \mathfrak{B}$   
 shows  $\mathcal{D}_\circ((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$   
*<proof>*

**lemma** *dghm-cn-comp-ArrMap-vrange*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{DG \mapsto \alpha} \mathfrak{B}$   
 shows  $\mathcal{R}_\circ((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap})) \subseteq \mathfrak{C}(\text{Arr})$   
*<proof>*

**lemma** *dghm-cn-comp-ArrMap-app[dg-cn-cs-simps]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{DG \mapsto \alpha} \mathfrak{B}$  and  $a \in \mathfrak{A}(\text{Arr})$   
 shows  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap})(a) = \mathfrak{G}(\text{ArrMap})(\mathfrak{F}(\text{ArrMap})(a))$   
*<proof>*

### Object map: contravariant and covariant digraph homomorphisms

**lemma** *dghm-cn-cov-comp-ObjMap-vsuv[dg-cn-cs-intros]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{\mapsto DG\alpha} \mathfrak{B}$   
 shows *vsu*  $((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap}))$   
*<proof>*

**lemma** *dghm-cn-cov-comp-ObjMap-vdomain[dg-cn-cs-simps]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{\mapsto DG\alpha} \mathfrak{B}$   
 shows  $\mathcal{D}_\circ((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$   
*<proof>*

**lemma** *dghm-cn-cov-comp-ObjMap-vrange*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{\mapsto DG\alpha} \mathfrak{B}$   
 shows  $\mathcal{R}_\circ((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})) \subseteq \mathfrak{C}(\text{Obj})$   
*<proof>*

**lemma** *dghm-cn-cov-comp-ObjMap-app[dg-cn-cs-simps]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{\mapsto DG\alpha} \mathfrak{B}$  and  $a \in \mathfrak{A}(\text{Obj})$   
 shows  $(\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ObjMap})(a) = \mathfrak{G}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(a))$   
*<proof>*

### Arrow map: contravariant and covariant digraph homomorphisms

**lemma** *dghm-cn-cov-comp-ArrMap-vsuv[dg-cn-cs-intros]*:  
 assumes  $\mathfrak{G} : \mathfrak{B}_{DG \mapsto \alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A}_{\mapsto DG\alpha} \mathfrak{B}$   
 shows *vsu*  $((\mathfrak{G}_{DGHM^\circ} \mathfrak{F})(\text{ArrMap}))$   
*<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**  $\mathcal{D}_\circ ((\mathfrak{G}_{DGHM \circ} \mathfrak{F})(\downarrow ArrMap)) = \mathfrak{A}(\downarrow Arr)$   
*<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**  $\mathcal{R}_\circ ((\mathfrak{G}_{DGHM \circ} \mathfrak{F})(\downarrow ArrMap)) \subseteq_\circ \mathfrak{C}(\downarrow Arr)$   
*<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**  $a \in_\circ \mathfrak{A}(\downarrow Arr)$   
**shows**  $(\mathfrak{G}_{DGHM \circ} \mathfrak{F})(\downarrow ArrMap)(\downarrow a) = \mathfrak{G}(\downarrow ArrMap)(\mathfrak{F}(\downarrow ArrMap)(\downarrow a))$   
*<proof>*

### Opposite of the contravariant composition of the digraph homomorphisms

**lemma** *op-dghm-dghm-cn-comp[dg-op-simps]:*

*op-dghm*  $(\mathfrak{G}_{DGHM \circ} \mathfrak{F}) = \text{op-dghm } \mathfrak{G}_{DGHM \circ} \text{op-dghm } \mathfrak{F}$   
*<proof>*

### Further properties

**lemma** *dghm-cn-comp-is-dghm[dg-cn-cs-intros]:*

— See section 1.2 in [15].

**assumes** *digraph*  $\alpha \mathfrak{A}$  **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*  $\mathfrak{C} = [\text{vid-on } (\mathfrak{C}(\downarrow Obj)), \text{vid-on } (\mathfrak{C}(\downarrow Arr)), \mathfrak{C}, \mathfrak{C}]_\circ$ .

Components.

**lemma** *dghm-id-components:*

**shows** *dghm-id*  $\mathfrak{C}(\downarrow ObjMap) = \text{vid-on } (\mathfrak{C}(\downarrow Obj))$   
**and** *dghm-id*  $\mathfrak{C}(\downarrow ArrMap) = \text{vid-on } (\mathfrak{C}(\downarrow Arr))$   
**and** [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-id*  $\mathfrak{C}(\downarrow HomDom) = \mathfrak{C}$   
**and** [*dg-shared-cs-simps*, *dg-cs-simps*]: *dghm-id*  $\mathfrak{C}(\downarrow HomCod) = \mathfrak{C}$   
*<proof>*

#### Object map

**mk-VLambda** *dghm-id-components*(1)[*folded VLambda-vid-on*]

|*vsv dghm-id-ObjMap-vsuv*[*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*]:

$\mathcal{R}_\circ (dghm-id \ \mathfrak{C}(\text{ObjMap})) = \mathfrak{C}(\text{Obj})$   
 ⟨*proof*⟩

### Arrow map

**mk-VLambda** *dghm-id-components*(2)[*folded VLambda-vid-on*]  
 |*vsv dghm-id-ArrMap-vsuv*[*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*]:

$\mathcal{R}_\circ (dghm-id \ \mathfrak{C}(\text{ArrMap})) = \mathfrak{C}(\text{Arr})$   
 ⟨*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*:  $dghm-id \ \mathfrak{C} : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{C}$   
 ⟨*proof*⟩

**lemma** (in *digraph*) *dg-dghm-id-is-dghm'*:  
 assumes  $\mathfrak{A} = \mathfrak{C}$  and  $\mathfrak{B} = \mathfrak{C}$   
 shows  $dghm-id \ \mathfrak{C} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
 ⟨*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* ( $\mathfrak{C}(\text{Obj})$ ) *a*, *vconst-on* ( $\mathfrak{C}(\text{Arr})$ ) *f*,  $\mathfrak{C}$ ,  $\mathfrak{D}$ ].

Components.

**lemma** *dghm-const-components*:

**shows**  $dghm-const \mathfrak{C} \mathfrak{D} a f(\text{ObjMap}) = vconst-on (\mathfrak{C}(\text{Obj})) a$   
**and**  $dghm-const \mathfrak{C} \mathfrak{D} a f(\text{ArrMap}) = vconst-on (\mathfrak{C}(\text{Arr})) f$   
**and**  $[dg-shared-cs-simps, dg-cs-simps]: dghm-const \mathfrak{C} \mathfrak{D} a f(\text{HomDom}) = \mathfrak{C}$   
**and**  $[dg-shared-cs-simps, dg-cs-simps]: dghm-const \mathfrak{C} \mathfrak{D} a f(\text{HomCod}) = \mathfrak{D}$   
 $\langle proof \rangle$

### Object map

**mk-VLambda** *dghm-const-components(1)[folded VLambda-vconst-on]*  
 $|vsu \ dghm-const-ObjMap-vsuv[dg-shared-cs-intros, dg-cs-intros]|$   
 $|vdomain \ dghm-const-ObjMap-vdomain[dg-shared-cs-simps, dg-cs-simps]|$   
 $|app \ dghm-const-ObjMap-app[dg-shared-cs-simps, dg-cs-simps]|$

### Arrow map

**mk-VLambda** *dghm-const-components(2)[folded VLambda-vconst-on]*  
 $|vsu \ dghm-const-ArrMap-vsuv[dg-shared-cs-intros, dg-cs-intros]|$   
 $|vdomain \ dghm-const-ArrMap-vdomain[dg-shared-cs-simps, dg-cs-simps]|$   
 $|app \ dghm-const-ArrMap-app[dg-shared-cs-simps, dg-cs-simps]|$

### Opposite constant digraph homomorphism

**lemma** *op-dghm-dghm-const[dg-op-simps]*:  
 $op-dghm (dghm-const \mathfrak{C} \mathfrak{D} a f) = dghm-const (op-dg \mathfrak{C}) (op-dg \mathfrak{D}) a f$   
 $\langle proof \rangle$

### A constant digraph homomorphism is a digraph homomorphism

**lemma** *dghm-const-is-dghm*:  
**assumes** *digraph*  $\alpha \mathfrak{C}$  **and** *digraph*  $\alpha \mathfrak{D}$  **and**  $f : a \mapsto_{\mathfrak{D}} a$   
**shows**  $dghm-const \mathfrak{C} \mathfrak{D} a f : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{D}$   
 $\langle proof \rangle$

**lemma** *dghm-const-is-dghm'[dg-cs-intros]*:  
**assumes** *digraph*  $\alpha \mathfrak{C}$   
**and** *digraph*  $\alpha \mathfrak{D}$   
**and**  $f : a \mapsto_{\mathfrak{D}} a$   
**and**  $\mathfrak{A} = \mathfrak{C}$   
**and**  $\mathfrak{B} = \mathfrak{D}$   
**shows**  $dghm-const \mathfrak{C} \mathfrak{D} a f : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
 $\langle proof \rangle$

### Further properties

**lemma** (**in** *is-dghm*) *dghm-dghm-comp-dghm-const[dg-cs-simps]*:  
**assumes** *digraph*  $\alpha \mathfrak{C}$  **and**  $f : a \mapsto_{\mathfrak{C}} a$   
**shows**  $dghm-const \mathfrak{B} \mathfrak{C} a f \circ_{DGHM} \mathfrak{F} = dghm-const \mathfrak{A} \mathfrak{C} a f$   
 $\langle proof \rangle$

**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*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  **for**  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +  
**assumes** *ft-dghm-v11-on-Hom*:  
 $\llbracket a \in_{\circ} \mathfrak{A}(\text{Obj}); b \in_{\circ} \mathfrak{A}(\text{Obj}) \rrbracket \implies v11 (\mathfrak{F}(\text{ArrMap}) \uparrow^{\circ} \text{Hom } \mathfrak{A} a b)$

**syntax** *is-ft-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 $\langle \langle - / - \mapsto \mapsto_{DG.\text{faithful}} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *is-ft-dghm*  $\equiv$  *is-ft-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{B} \equiv \text{CONST } \textit{is-ft-dghm } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

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**  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{B}'$   
 $\langle \textit{proof} \rangle$

**mk-ide rf** *is-ft-dghm-def*[*unfolded is-ft-dghm-axioms-def*]  
 $| \textit{intro is-ft-dghmI} |$   
 $| \textit{dest is-ft-dghmD}[\textit{dest}] |$   
 $| \textit{elim is-ft-dghmE}[\textit{elim}] |$

**lemmas** [*dghm-cs-intros*] = *is-ft-dghmD*(1)

**lemma** *is-ft-dghmI'*:  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG} \mathfrak{B}$   
**and**  $\wedge a b g f.$   
 $\llbracket g : a \mapsto_{\mathfrak{A}} b; f : a \mapsto_{\mathfrak{A}} b; \mathfrak{F}(\text{ArrMap})(g) = \mathfrak{F}(\text{ArrMap})(f) \rrbracket \implies g = f$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{B}$   
 $\langle \textit{proof} \rangle$

### Opposite faithful digraph homomorphism

**lemma** (in *is-ft-dghm*) *ft-dghm-op-dghm-is-ft-dghm*:  
 $op\text{-dghm } \mathfrak{F} : op\text{-dg } \mathfrak{A} \mapsto \mapsto_{DG.\text{faithful}} op\text{-dg } \mathfrak{B}$   
 $\langle \textit{proof} \rangle$

**lemma** (in *is-ft-dghm*) *ft-dghm-op-dghm-is-ft-dghm'*[*dg-op-intros*]:  
**assumes**  $\mathfrak{A}' = op\text{-dg } \mathfrak{A}$  **and**  $\mathfrak{B}' = op\text{-dg } \mathfrak{B}$   
**shows**  $op\text{-dghm } \mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{B}'$   
 $\langle \textit{proof} \rangle$

**lemmas** *ft-dghm-op-dghm-is-ft-dghm*[*dg-op-intros*] =  
 $is\text{-ft-dghm.ft-dghm-op-dghm-is-ft-dghm}'$

**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**  $\mathfrak{G} : \mathfrak{B} \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{B}$   
**shows**  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.\text{faithful}} \mathfrak{C}$   
 $\langle \textit{proof} \rangle$

### 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**  $\mathfrak{F}(\text{ArrMap})(g) = \mathfrak{F}(\text{ArrMap})(f)$   
**shows**  $g = f$   
 $\langle \textit{proof} \rangle$

### 3.4.9 Full digraph homomorphism

#### Definition and elementary properties

See Chapter I-3 in [39].

**locale** *is-fl-dghm* = *is-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  for  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +

**assumes** *fl-dghm-surj-on-Hom*:

$\llbracket a \in_{\circ} \mathfrak{A}(\text{Obj}); b \in_{\circ} \mathfrak{A}(\text{Obj}) \rrbracket \implies$

$\mathfrak{F}(\text{ArrMap}) \circ (\text{Hom } \mathfrak{A} \ a \ b) = \text{Hom } \mathfrak{B} \ (\mathfrak{F}(\text{ObjMap})(a)) \ (\mathfrak{F}(\text{ObjMap})(b))$

**syntax** *is-fl-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ :/ } - \mapsto_{DG.full} - \rangle \rangle$  [51, 51, 51] 51)

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG.full} \mathfrak{B} \ni \text{CONST } is-fl-dghm \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F}$

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**  $\mathfrak{F} : \mathfrak{A}' \mapsto_{DG.full} \mathfrak{B}'$

*<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*  $\mathfrak{F} : \text{op-dg } \mathfrak{A} \mapsto_{DG.full} \text{op-dg } \mathfrak{B}$

*<proof>*

**lemma** (in *is-fl-dghm*) *fl-dghm-op-dghm-is-fl-dghm'*[*dg-op-intros*]:

**assumes**  $\mathfrak{A}' = \text{op-dg } \mathfrak{A}$  **and**  $\mathfrak{B}' = \text{op-dg } \mathfrak{B}$

**shows** *op-dghm*  $\mathfrak{F} : \text{op-dg } \mathfrak{A} \mapsto_{DG.full} \text{op-dg } \mathfrak{B}$

*<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**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG.full} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG.full} \mathfrak{B}$

**shows**  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto_{DG.full} \mathfrak{C}$

*<proof>*

### 3.4.10 Fully faithful digraph homomorphism

#### Definition and elementary properties

See Chapter I-3 in [39].

**locale** *is-ff-dghm* = *is-fl-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  + *is-fl-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  for  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$

**syntax** *is-ff-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ :/ } - \mapsto_{DG.ff} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-ff-dghm*  $\equiv$  *is-ff-dghm*  
**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.ff\alpha} \mathfrak{B} \equiv \text{CONST } is-ff-dghm \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F}$

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* ( $\mathfrak{F} \downarrow \text{ObjMap}$ )  
**and** *iso-dghm-ArrMap-v11*: *v11* ( $\mathfrak{F} \downarrow \text{ArrMap}$ )  
**and** *iso-dghm-ObjMap-vrange*[*dghm-cs-simps*]:  $\mathcal{R}_\circ (\mathfrak{F} \downarrow \text{ObjMap}) = \mathfrak{B} \downarrow \text{Obj}$   
**and** *iso-dghm-ArrMap-vrange*[*dghm-cs-simps*]:  $\mathcal{R}_\circ (\mathfrak{F} \downarrow \text{ArrMap}) = \mathfrak{B} \downarrow \text{Arr}$

**syntax** *-is-iso-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
*<(- :/ - \mapsto \mapsto\_{DG.iso1} -)>* [51, 51, 51] 51

**syntax-consts** *-is-iso-dghm*  $\equiv$  *is-iso-dghm*  
**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B} \equiv \text{CONST } is-iso-dghm \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F}$

**sublocale** *is-iso-dghm*  $\subseteq$  *ObjMap*: *v11*  $\langle \mathfrak{F} \downarrow \text{ObjMap} \rangle$   
**rewrites**  $\mathcal{D}_\circ (\mathfrak{F} \downarrow \text{ObjMap}) = \mathfrak{A} \downarrow \text{Obj}$  **and**  $\mathcal{R}_\circ (\mathfrak{F} \downarrow \text{ObjMap}) = \mathfrak{B} \downarrow \text{Obj}$   
*<proof>*

**sublocale**  $is\text{-}iso\text{-}dghm \subseteq ArrMap$ : v11  $\langle \mathfrak{F}(\downarrow ArrMap) \rangle$   
**rewrites**  $\mathcal{D}_\circ(\mathfrak{F}(\downarrow ArrMap)) = \mathfrak{A}(\downarrow Arr)$  **and**  $\mathcal{R}_\circ(\mathfrak{F}(\downarrow ArrMap)) = \mathfrak{B}(\downarrow Arr)$   
 $\langle proof \rangle$

**lemmas**  $[dghm\text{-}cs\text{-}simps] =$   
 $is\text{-}iso\text{-}dghm.iso\text{-}dghm\text{-}ObjMap\text{-}vrangle$   
 $is\text{-}iso\text{-}dghm.iso\text{-}dghm\text{-}ArrMap\text{-}vrangle$

Rules.

**lemma** (**in**  $is\text{-}iso\text{-}dghm$ )  $is\text{-}iso\text{-}dghm\text{-}axioms'$  $[dghm\text{-}cs\text{-}intros]$ :  
**assumes**  $\alpha' = \alpha$  **and**  $\mathfrak{A}' = \mathfrak{A}$  **and**  $\mathfrak{B}' = \mathfrak{B}$   
**shows**  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG} iso_{\alpha'} \mathfrak{B}'$   
 $\langle proof \rangle$

**mk-ide rf**  $is\text{-}iso\text{-}dghm\text{-}def$  $[unfolded\ is\text{-}iso\text{-}dghm\text{-}axioms\text{-}def]$   
 $|intro\ is\text{-}iso\text{-}dghmI|$   
 $|dest\ is\text{-}iso\text{-}dghmD[dest]|$   
 $|elim\ is\text{-}iso\text{-}dghmE[elim]|$

Elementary properties.

**lemma** (**in**  $is\text{-}iso\text{-}dghm$ )  $iso\text{-}dghm\text{-}Obj\text{-}Hom\text{-}Dom\text{-}if\text{-}Obj\text{-}Hom\text{-}Cod$  $[elim]$ :  
**assumes**  $b \in_\circ \mathfrak{B}(\downarrow Obj)$   
**obtains**  $a$  **where**  $a \in_\circ \mathfrak{A}(\downarrow Obj)$  **and**  $b = \mathfrak{F}(\downarrow ObjMap)(\downarrow a)$   
 $\langle proof \rangle$

**lemma** (**in**  $is\text{-}iso\text{-}dghm$ )  $iso\text{-}dghm\text{-}Arr\text{-}Hom\text{-}Dom\text{-}if\text{-}Arr\text{-}Hom\text{-}Cod$  $[elim]$ :  
**assumes**  $g \in_\circ \mathfrak{B}(\downarrow Arr)$   
**obtains**  $f$  **where**  $f \in_\circ \mathfrak{A}(\downarrow Arr)$  **and**  $g = \mathfrak{F}(\downarrow ArrMap)(\downarrow f)$   
 $\langle proof \rangle$

**lemma** (**in**  $is\text{-}iso\text{-}dghm$ )  $iso\text{-}dghm\text{-}ObjMap\text{-}eqE$  $[elim]$ :  
**assumes**  $\mathfrak{F}(\downarrow ObjMap)(\downarrow a) = \mathfrak{F}(\downarrow ObjMap)(\downarrow b)$   
**and**  $a \in_\circ \mathfrak{A}(\downarrow Obj)$   
**and**  $b \in_\circ \mathfrak{A}(\downarrow Obj)$   
**and**  $a = b \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**lemma** (**in**  $is\text{-}iso\text{-}dghm$ )  $iso\text{-}dghm\text{-}ArrMap\text{-}eqE$  $[elim]$ :  
**assumes**  $\mathfrak{F}(\downarrow ArrMap)(\downarrow f) = \mathfrak{F}(\downarrow ArrMap)(\downarrow g)$   
**and**  $f \in_\circ \mathfrak{A}(\downarrow Arr)$   
**and**  $g \in_\circ \mathfrak{A}(\downarrow Arr)$   
**and**  $f = g \implies P$   
**shows**  $P$   
 $\langle proof \rangle$

**sublocale**  $is\text{-}iso\text{-}dghm \subseteq is\text{-}ft\text{-}dghm$   
 $\langle proof \rangle$

**sublocale**  $is\text{-}iso\text{-}dghm \subseteq is\text{-}fl\text{-}dghm$   
 $\langle proof \rangle$

**sublocale**  $is\text{-}iso\text{-}dghm \subseteq is\text{-}ff\text{-}dghm$   $\langle proof \rangle$

**lemmas** (**in**  $is\text{-}iso\text{-}dghm$ )  $iso\text{-}dghm\text{-}is\text{-}ff\text{-}dghm = is\text{-}ff\text{-}dghm\text{-}axioms$   
**lemmas**  $[dghm\text{-}cs\text{-}intros] = is\text{-}iso\text{-}dghm.iso\text{-}dghm\text{-}is\text{-}ff\text{-}dghm$

**Opposite digraph isomorphisms**

**lemma** (in *is-iso-dghm*) *is-iso-dghm-op*:  
 $op\text{-}dghm \mathfrak{F} : op\text{-}dg \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} op\text{-}dg \mathfrak{B}$   
 ⟨proof⟩

**lemma** (in *is-iso-dghm*) *is-iso-dghm-op'*:  
 assumes  $\mathfrak{A}' = op\text{-}dg \mathfrak{A}$  and  $\mathfrak{B}' = op\text{-}dg \mathfrak{B}$   
 shows  $op\text{-}dghm \mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B}'$   
 ⟨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*  $\mathfrak{F} = [(\mathfrak{F}(\mathfrak{ObjMap}))^{-1} \circ, (\mathfrak{F}(\mathfrak{ArrMap}))^{-1} \circ, \mathfrak{F}(\mathfrak{HomCod}), \mathfrak{F}(\mathfrak{HomDom})]$ .

Components.

**lemma** *inv-dghm-components*:  
 shows *inv-dghm*  $\mathfrak{F}(\mathfrak{ObjMap}) = (\mathfrak{F}(\mathfrak{ObjMap}))^{-1} \circ$   
 and *inv-dghm*  $\mathfrak{F}(\mathfrak{ArrMap}) = (\mathfrak{F}(\mathfrak{ArrMap}))^{-1} \circ$   
 and [*dghm-cs-simps*]: *inv-dghm*  $\mathfrak{F}(\mathfrak{HomDom}) = \mathfrak{F}(\mathfrak{HomCod})$   
 and [*dghm-cs-simps*]: *inv-dghm*  $\mathfrak{F}(\mathfrak{HomCod}) = \mathfrak{F}(\mathfrak{HomDom})$   
 ⟨proof⟩

**Object map**

**lemma** (in *is-iso-dghm*) *inv-dghm-ObjMap-v11*[*dghm-cs-intros*]:  
 $v11 (inv\text{-}dghm \mathfrak{F}(\mathfrak{ObjMap}))$   
 ⟨proof⟩

**lemmas** [*dghm-cs-intros*] = *is-iso-dghm.inv-dghm-ObjMap-v11*

**lemma** (in *is-iso-dghm*) *inv-dghm-ObjMap-vdomain*[*dghm-cs-simps*]:  
 $\mathcal{D}_\circ (inv\text{-}dghm \mathfrak{F}(\mathfrak{ObjMap})) = \mathfrak{B}(\mathfrak{Obj})$   
 ⟨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**  $a' = \mathfrak{F}(\text{ObjMap})(a)$  **and**  $a \in \mathfrak{A}(\text{Obj})$   
**shows** *inv-dghm*  $\mathfrak{F}(\text{ObjMap})(a') = a$   
*<proof>*

**lemmas** [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ObjMap-app*

**lemma** (in *is-iso-dghm*) *inv-dghm-ObjMap-vrange*[*dghm-cs-simps*]:  
 $\mathcal{R}_\circ(\text{inv-dghm } \mathfrak{F}(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$   
*<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*]:  
 $v11(\text{inv-dghm } \mathfrak{F}(\text{ArrMap}))$   
*<proof>*

**lemmas** [*dghm-cs-intros*] = *is-iso-dghm.inv-dghm-ArrMap-v11*

**lemma** (in *is-iso-dghm*) *inv-dghm-ArrMap-vdomain*[*dghm-cs-simps*]:  
 $\mathcal{D}_\circ(\text{inv-dghm } \mathfrak{F}(\text{ArrMap})) = \mathfrak{B}(\text{Arr})$   
*<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**  $a' = \mathfrak{F}(\text{ArrMap})(a)$  **and**  $a \in \mathfrak{A}(\text{Arr})$   
**shows** *inv-dghm*  $\mathfrak{F}(\text{ArrMap})(a') = a$   
*<proof>*

**lemmas** [*dghm-cs-simps*] = *is-iso-dghm.inv-dghm-ArrMap-app*

**lemma** (in *is-iso-dghm*) *inv-dghm-ArrMap-vrange*[*dghm-cs-simps*]:  
 $\mathcal{R}_\circ(\text{inv-dghm } \mathfrak{F}(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$   
*<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**  $a \in \mathfrak{B}(\text{Obj})$   
**shows**  $\mathfrak{F}(\text{ObjMap})(\text{inv-dghm } \mathfrak{F}(\text{ObjMap})(a)) = a$   
*<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**  $\mathfrak{F}(\text{ArrMap})(\text{inv-dghm } \mathfrak{F}(\text{ArrMap})(f)) = f$   
*<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**  $f \in_{\circ} \mathfrak{A}(\text{Arr})$   
**and**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$   
**and**  $b \in_{\circ} \mathfrak{A}(\text{Obj})$   
**and**  $\mathfrak{F}(\text{ArrMap})(f) : \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(b)$   
**shows**  $f : a \mapsto_{\mathfrak{A}} b$   
 $\langle \text{proof} \rangle$

**lemma** (in *is-iso-dghm*) *iso-dghm-HomCod-is-arr-conv*:  
**assumes**  $f \in_{\circ} \mathfrak{B}(\text{Arr})$   
**and**  $a \in_{\circ} \mathfrak{B}(\text{Obj})$   
**and**  $b \in_{\circ} \mathfrak{B}(\text{Obj})$   
**and**  $\text{inv-dghm } \mathfrak{F}(\text{ArrMap})(f) : \text{inv-dghm } \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{A}} \text{inv-dghm } \mathfrak{F}(\text{ObjMap})(b)$   
**shows**  $f : a \mapsto_{\mathfrak{B}} b$   
 $\langle \text{proof} \rangle$

### 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**  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} = \text{dghm-id } \mathfrak{A}$   
**and**  $\mathfrak{F} \circ_{DGHM} \mathfrak{G} = \text{dghm-id } \mathfrak{B}$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{\mapsto_{DG.iso\alpha}} \mathfrak{B}$   
 $\langle \text{proof} \rangle$

**lemma** *is-iso-dghm-is-iso-arr*:  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto_{\mapsto_{DG.iso\alpha}} \mathfrak{B}$   
**shows** [*dghm-cs-intros*]:  $\text{inv-dghm } \mathfrak{F} : \mathfrak{B} \mapsto_{\mapsto_{DG.iso\alpha}} \mathfrak{A}$   
**and** [*dghm-cs-simps*]:  $\text{inv-dghm } \mathfrak{F} \circ_{DGHM} \mathfrak{F} = \text{dghm-id } \mathfrak{A}$   
**and** [*dghm-cs-simps*]:  $\mathfrak{F} \circ_{DGHM} \text{inv-dghm } \mathfrak{F} = \text{dghm-id } \mathfrak{B}$   
 $\langle \text{proof} \rangle$

#### Further properties

**lemma** (in *is-iso-dghm*) *iso-inv-dghm-ObjMap-dghm-ObjMap-app*[*dghm-cs-simps*]:  
**assumes**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$   
**shows**  $\text{inv-dghm } \mathfrak{F}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(a)) = a$   
 $\langle \text{proof} \rangle$

**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**  $\text{inv-dghm } \mathfrak{F}(\text{ArrMap})(\mathfrak{F}(\text{ArrMap})(f)) = f$   
 $\langle \text{proof} \rangle$

**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**  $\langle \approx_{DG} \rangle$  50)

**sublocale** *iso-digraph*  $\subseteq HomDom$ : *digraph*  $\alpha \mathfrak{A} + HomCod$ : *digraph*  $\alpha \mathfrak{B}$   
 $\langle proof \rangle$

Rules.

**lemma** *iso-digraphI'*:

**assumes**  $\exists \mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B}$

**shows**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$

$\langle proof \rangle$

**lemma** *iso-digraphI*:

**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B}$

**shows**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$

$\langle proof \rangle$

**lemma** *iso-digraphD[dest]*:

**assumes**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$

**shows**  $\exists \mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B}$

$\langle proof \rangle$

**lemma** *iso-digraphE[elim]*:

**assumes**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$

**obtains**  $\mathfrak{F}$  **where**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.iso\alpha} \mathfrak{B}$

$\langle proof \rangle$

## A digraph isomorphism is an equivalence relation

**lemma** *iso-digraph-refl*:

**assumes** *digraph*  $\alpha \mathfrak{A}$

**shows**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{A}$

$\langle proof \rangle$

**lemma** *iso-digraph-sym[sym]*:

**assumes**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$

**shows**  $\mathfrak{B} \approx_{DG\alpha} \mathfrak{A}$

$\langle proof \rangle$

**lemma** *iso-digraph-trans[trans]*:

**assumes**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{B}$  **and**  $\mathfrak{B} \approx_{DG\alpha} \mathfrak{C}$

**shows**  $\mathfrak{A} \approx_{DG\alpha} \mathfrak{C}$

$\langle proof \rangle$

### 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*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +  
*HomDom*: *digraph*  $\alpha$   $\mathfrak{A}$  +  
*HomCod*: *digraph*  $\alpha$   $\mathfrak{B}$   
**for**  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +  
**assumes** *tm-dghm-ObjMap-in-Vset*[*dg-small-cs-intros*]:  $\mathfrak{F}(\text{ObjMap}) \in_{\circ} \text{Vset } \alpha$   
**and** *tm-dghm-ArrMap-in-Vset*[*dg-small-cs-intros*]:  $\mathfrak{F}(\text{ArrMap}) \in_{\circ} \text{Vset } \alpha$

**syntax** *is-tm-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ : } - \mapsto \mapsto_{DG.tm1} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *is-tm-dghm*  $\equiv$  *is-tm-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} \equiv \text{CONST } is\text{-tm-dghm } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-tm-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-cn-tm-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F} \equiv \mathfrak{F} : \text{op-dg } \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}$

**syntax** *is-cn-tm-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ : } - \text{ }_{DG.tm} \mapsto \mapsto 1 - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *is-cn-tm-dghm*  $\equiv$  *is-cn-tm-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{DG.tm} \mapsto \mapsto \alpha \mathfrak{B} \rightarrow \text{CONST } is\text{-cn-tm-dghm } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** *all-tm-dghms* ::  $V \Rightarrow V$

**where** *all-tm-dghms*  $\alpha \equiv \text{set } \{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}\}$

**abbreviation** *small-tm-dghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *small-tm-dghms*  $\alpha \mathfrak{A} \mathfrak{B} \equiv \text{set } \{\mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}\}$

**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*  $\sqsubseteq$  *HomDom.tiny-digraph*  $\alpha$   $\mathfrak{A}$

*<proof>*

**lemmas** (**in** *is-tm-dghm*)

*tm-dghm-HomDom-is-tiny-digraph* = *HomDom.tiny-digraph-axioms*

lemmas  $[dg\text{-small-cs-intros}] = is\text{-tm-dghm.tm-dghm-HomDom-is-tiny-digraph}$

Further rules.

**lemma**  $is\text{-tm-dghmI}'$ :

assumes  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG\alpha} \mathfrak{B}$   
 and  $[simp]: \mathfrak{F}(\mathcal{O}bjMap) \in_0 Vset\ \alpha$   
 and  $[simp]: \mathfrak{F}(\mathcal{A}rrMap) \in_0 Vset\ \alpha$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}$

$\langle proof \rangle$

Size.

**lemma**  $small\text{-all-tm-dghms}[simp]: small\ \{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}\}$

$\langle proof \rangle$

### Opposite digraph homomorphism with tiny maps

**lemma** (in  $is\text{-tm-dghm}$ )  $is\text{-tm-dghm-op}: op\text{-dghm}\ \mathfrak{F} : op\text{-dg}\ \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} op\text{-dg}\ \mathfrak{B}$

$\langle proof \rangle$

**lemma** (in  $is\text{-tm-dghm}$ )  $is\text{-tm-dghm-op}'[dg\text{-op-intros}]$ :

assumes  $\mathfrak{A}' = op\text{-dg}\ \mathfrak{A}$  and  $\mathfrak{B}' = op\text{-dg}\ \mathfrak{B}$  and  $\alpha' = \alpha$   
 shows  $op\text{-dghm}\ \mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.tm\alpha'} \mathfrak{B}'$

$\langle proof \rangle$

lemmas  $is\text{-tm-dghm-op}[dg\text{-op-intros}] = is\text{-tm-dghm.is-tm-dghm-op}'$

### Composition of a digraph homomorphism with tiny maps

**lemma**  $dghm\text{-comp-is-tm-dghm}[dg\text{-small-cs-intros}]$ :

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  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C}$

$\langle proof \rangle$

### Finite digraphs and digraph homomorphisms with tiny maps

**lemma** (in  $is\text{-dghm}$ )  $dghm\text{-is-tm-dghm-if-HomDom-finite-digraph}$ :

assumes  $finite\text{-digraph}\ \alpha\ \mathfrak{A}$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}$

$\langle proof \rangle$

### Constant digraph homomorphism

**lemma**  $dghm\text{-const-is-tm-dghm}$ :

assumes  $tiny\text{-digraph}\ \alpha\ \mathfrak{C}$  and  $digraph\ \alpha\ \mathfrak{D}$  and  $f : a \mapsto_{\mathfrak{D}} a$   
 shows  $dghm\text{-const}\ \mathfrak{C}\ \mathfrak{D}\ a\ f : \mathfrak{C} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{D}$

$\langle proof \rangle$

**lemma**  $dghm\text{-const-is-tm-dghm}'[dg\text{-small-cs-intros}]$ :

assumes  $tiny\text{-digraph}\ \alpha\ \mathfrak{C}$   
 and  $digraph\ \alpha\ \mathfrak{D}$   
 and  $f : a \mapsto_{\mathfrak{D}} a$   
 and  $\mathfrak{C}' = \mathfrak{C}$   
 and  $\mathfrak{D}' = \mathfrak{D}$

shows  $dghm\text{-const}\ \mathfrak{C}\ \mathfrak{D}\ a\ f : \mathfrak{C}' \mapsto \mapsto_{DG.tm\alpha} \mathfrak{D}'$

$\langle proof \rangle$

### 3.5.2 Tiny digraph homomorphism

#### Definition and elementary properties

**locale** *is-tiny-dghm* =

*is-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +

*HomDom*: *tiny-digraph*  $\alpha$   $\mathfrak{A}$  +

*HomCod*: *tiny-digraph*  $\alpha$   $\mathfrak{B}$

for  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$

**syntax** *-is-tiny-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ : / } - \mapsto \mapsto_{DG.tiny^1} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-tiny-dghm*  $\equiv$  *is-tiny-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tiny\alpha} \mathfrak{B} \equiv \text{CONST } is-tiny-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-tiny-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *is-cn-tiny-dghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F} \equiv \mathfrak{F} : op-dg \mathfrak{A} \mapsto \mapsto_{DG.tiny\alpha} \mathfrak{B}$

**syntax** *-is-cn-tiny-dghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ : / } - \text{ }_{DG.tiny} \mapsto \mapsto_1 - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-cn-tiny-dghm*  $\equiv$  *is-cn-tiny-dghm*

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{DG.tiny} \mapsto \mapsto_{\alpha} \mathfrak{B} \rightarrow \text{CONST } is-cn-tiny-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** *all-tiny-dghms* ::  $V \Rightarrow V$

where *all-tiny-dghms*  $\alpha \equiv \text{set } \{ \mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tiny\alpha} \mathfrak{B} \}$

**abbreviation** *small-dghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *small-dghms*  $\alpha \mathfrak{A} \mathfrak{B} \equiv \text{set } \{ \mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG.tiny\alpha} \mathfrak{B} \}$

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*]:

$\mathfrak{F}(\text{ObjMap}) \in_{\circ} \text{Vset } \alpha$

*<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*]:

$\mathfrak{F}(\text{ArrMap}) \in_{\circ} \text{Vset } \alpha$

*<proof>*

**lemmas** [*dg-small-cs-intros*] = *is-tiny-dghm.tiny-dghm-ArrMap-in-Vset*

**lemma** (in *is-tiny-dghm*) *tiny-dghm-in-Vset*:  $\mathfrak{F} \in_{\circ} \text{Vset } \alpha$

*<proof>*

**sublocale** *is-tiny-dghm*  $\subseteq$  *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 { $\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{B}$ }*

*<proof>*

**lemma** *tiny-dghms-vsubset-Vset[simp]:*

*set { $\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{B}$ }  $\subseteq_o$  *Vset*  $\alpha$*

*<proof>*

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

**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_o \beta$

**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG.tiny\beta} \mathfrak{B}$

*<proof>*

### Opposite tiny digraph homomorphism

**lemma** (in *is-tiny-dghm*) *is-tiny-dghm-op:*

*op-dghm  $\mathfrak{F} : op-dg \mathfrak{A} \mapsto_{DG.tiny\alpha} op-dg \mathfrak{B}$*

*<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  $\mathfrak{F} : \mathfrak{A}' \mapsto_{DG.tiny\alpha'} \mathfrak{B}'$*

*<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**  $\mathfrak{G} : \mathfrak{B} \mapsto_{DG.tiny\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{B}$

**shows**  $\mathfrak{G} \circ_{DGHM} \mathfrak{F} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{C}$

*<proof>*

### Tiny constant digraph homomorphism

**lemma** *dghm-const-is-tiny-dghm:*

**assumes** *tiny-digraph*  $\alpha \mathfrak{C}$  **and** *tiny-digraph*  $\alpha \mathfrak{D}$  **and**  $f : a \mapsto_{\mathfrak{D}} a$

**shows** *dghm-const*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{C} \mapsto_{DG.tiny\alpha} \mathfrak{D}$

*<proof>*

**lemma** *dghm-const-is-tiny-dghm'[dg-small-cs-intros]:*

**assumes** *tiny-digraph*  $\alpha \mathfrak{C}$

**and** *tiny-digraph*  $\alpha \mathfrak{D}$

**and**  $f : a \mapsto_{\mathfrak{D}} a$

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

**and**  $\mathfrak{D}' = \mathfrak{D}$

**shows** *dghm-const*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{C}' \mapsto_{DG.tiny\alpha} \mathfrak{D}'$

*<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  $\alpha$  such that  $\omega < \alpha$ . Such transformations of digraph homomorphisms are referred to either as  $\alpha$ -transformations of digraph homomorphisms or transformations of  $\alpha$ -digraph homomorphisms.

locale *is-tdghm* =

$Z$   $\alpha$  +

*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-usv*: *usv* ( $\mathfrak{N}(\mathit{NTMap})$ )

**and** *tdghm-NTMap-vdomain*[*dg-cs-simps*]:  $\mathcal{D}_\circ$  ( $\mathfrak{N}(\mathit{NTMap})$ ) =  $\mathfrak{A}(\mathit{Obj})$

**and** *tdghm-NTDom*[*dg-cs-simps*]:  $\mathfrak{N}(\mathit{NTDom})$  =  $\mathfrak{F}$

**and** *tdghm-NTCod*[*dg-cs-simps*]:  $\mathfrak{N}(\mathit{NTCod})$  =  $\mathfrak{G}$

**and** *tdghm-NTDGDom*[*dg-cs-simps*]:  $\mathfrak{N}(\mathit{NTDGDom})$  =  $\mathfrak{A}$

**and** *tdghm-NTDGCod*[*dg-cs-simps*]:  $\mathfrak{N}(\mathit{NTDGCod})$  =  $\mathfrak{B}$

**and** *tdghm-NTMap-is-arr*:

$a \in_\circ \mathfrak{A}(\mathit{Obj}) \implies \mathfrak{N}(\mathit{NTMap})(\mathit{a}) : \mathfrak{F}(\mathit{ObjMap})(\mathit{a}) \mapsto_{\mathfrak{B}} \mathfrak{G}(\mathit{ObjMap})(\mathit{a})$

syntax *is-tdghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \mathit{bool}$

$\langle \langle - \cdot / - \mapsto_{DGHM} - \cdot / - \mapsto_{DG^1} - \rangle \rangle$  [51, 51, 51, 51, 51] 51

syntax-consts *is-tdghm*  $\equiv$  *is-tdghm*

translations  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B} \equiv$

*CONST is-tdghm*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$   $\mathfrak{G}$   $\mathfrak{N}$

abbreviation *all-tdghms* ::  $V \Rightarrow V$

**where** *all-tdghms*  $\alpha \equiv \mathit{set} \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B} \}$

abbreviation *tdghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *tdghms*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B} \equiv \mathit{set} \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B} \}$

abbreviation *these-tdghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *these-tdghms*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$   $\mathfrak{G} \equiv \mathit{set} \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B} \}$

sublocale *is-tdghm*  $\subseteq$  *NTMap*: *usv*  $\langle \mathfrak{N}(\mathit{NTMap}) \rangle$

**rewrites**  $\mathcal{D}_\circ$  ( $\mathfrak{N}(\mathit{NTMap})$ ) =  $\mathfrak{A}(\mathit{Obj})$

$\langle \mathit{proof} \rangle$

**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**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$   
**and**  $A = \mathfrak{F}(\text{ObjMap})(\downarrow a)$   
**and**  $B = \mathfrak{G}(\text{ObjMap})(\downarrow a)$   
**shows**  $\mathfrak{N}(\text{NTMap})(\downarrow a) : A \mapsto_{\mathfrak{B}} B$   
 $\langle \text{proof} \rangle$

**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**  $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG\alpha'} \mathfrak{B}'$   
 $\langle \text{proof} \rangle$

**mk-ide rf** *is-tdghm-def*[*unfolded is-tdghm-axioms-def*]  
 $[\text{intro } \text{is-tdghm}I]$   
 $[\text{dest } \text{is-tdghm}D[\text{dest}]]$   
 $[\text{elim } \text{is-tdghm}E[\text{elim}]]$

**lemmas** [*dg-cs-intros*] =  
*is-tdghmD(3,4)*

Elementary properties.

**lemma** *tdghm-eqI*:  
**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG\alpha'} \mathfrak{B}'$   
**and**  $\mathfrak{N}(\text{NTMap}) = \mathfrak{N}'(\text{NTMap})$   
**and**  $\mathfrak{F} = \mathfrak{F}'$   
**and**  $\mathfrak{G} = \mathfrak{G}'$   
**and**  $\mathfrak{A} = \mathfrak{A}'$   
**and**  $\mathfrak{B} = \mathfrak{B}'$   
**shows**  $\mathfrak{N} = \mathfrak{N}'$   
 $\langle \text{proof} \rangle$

**lemma** (in *is-tdghm*) *tdghm-def*:  
 $\mathfrak{N} = [\mathfrak{N}(\text{NTMap}), \mathfrak{N}(\text{NTDom}), \mathfrak{N}(\text{NTCod}), \mathfrak{N}(\text{NTDGDom}), \mathfrak{N}(\text{NTDGCod})]_{\circ}$   
 $\langle \text{proof} \rangle$

**lemma** (in *is-tdghm*) *tdghm-NTMap-app-in-Arr*[*dg-cs-intros*]:  
**assumes**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$   
**shows**  $\mathfrak{N}(\text{NTMap})(\downarrow a) \in_{\circ} \mathfrak{B}(\text{Arr})$   
 $\langle \text{proof} \rangle$

**lemmas** [*dg-cs-intros*] = *is-tdghm.tdghm-NTMap-app-in-Arr*

**lemma** (in *is-tdghm*) *tdghm-NTMap-vrange-vifunion*:  
 $\mathcal{R}_{\circ}(\mathfrak{N}(\text{NTMap})) \subseteq_{\circ} (\cup_{\circ} a \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{F}(\text{ObjMap}))) \cup_{\circ} b \in_{\circ} \mathcal{R}_{\circ}(\mathfrak{G}(\text{ObjMap})). \text{Hom } \mathfrak{B} \text{ } a \text{ } b$   
 $\langle \text{proof} \rangle$

**lemma** (in *is-tdghm*) *tdghm-NTMap-vrange*:  $\mathcal{R}_\circ (\mathfrak{N}(\text{NTMap})) \subseteq_\circ \mathfrak{B}(\text{Arr})$   
 ⟨proof⟩

Size.

**lemma** (in *is-tdghm*) *tdghm-NTMap-uset-Vset*:  $\mathfrak{N}(\text{NTMap}) \subseteq_\circ \text{Vset } \alpha$   
 ⟨proof⟩

**lemma** (in *is-tdghm*) *tdghm-NTMap-in-Vset*:  
 assumes  $\alpha \in_\circ \beta$   
 shows  $\mathfrak{N}(\text{NTMap}) \in_\circ \text{Vset } \beta$   
 ⟨proof⟩

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

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

**lemma** *small-all-tdghms[simp]*:  
 small  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG} \alpha \mathfrak{B}\}$   
 ⟨proof⟩

**lemma** *small-tdghms[simp]*: small  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG} \alpha \mathfrak{B}\}$   
 ⟨proof⟩

**lemma** *small-these-tdghms[simp]*: small  $\{\mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG} \alpha \mathfrak{B}\}$   
 ⟨proof⟩

Further elementary results.

**lemma** *these-tdghms-iff*:  
 $\mathfrak{N} \in_\circ \text{these-tdghms } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \iff \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG} \alpha \mathfrak{B}$   
 ⟨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*  $\mathfrak{N} =$   
 [  $\mathfrak{N}(\text{NTMap})$ ,  
 $\text{op-dghm } (\mathfrak{N}(\text{NTCod}))$ ,  
 $\text{op-dghm } (\mathfrak{N}(\text{NTDom}))$ ,  
 $\text{op-dg } (\mathfrak{N}(\text{NTDGDom}))$ ,  
 $\text{op-dg } (\mathfrak{N}(\text{NTDGCod}))$   
 ]<sub>o</sub>

Components.

**lemma** *op-tdghm-components[dg-op-simps]*:  
 shows *op-tdghm*  $\mathfrak{N}(\text{NTMap}) = \mathfrak{N}(\text{NTMap})$   
 and *op-tdghm*  $\mathfrak{N}(\text{NTDom}) = \text{op-dghm } (\mathfrak{N}(\text{NTCod}))$

**and**  $op\text{-}tdghm \mathfrak{N}(\mathcal{NTCod}) = op\text{-}dghm (\mathfrak{N}(\mathcal{NTDom}))$   
**and**  $op\text{-}tdghm \mathfrak{N}(\mathcal{NTDGDom}) = op\text{-}dg (\mathfrak{N}(\mathcal{NTDGDom}))$   
**and**  $op\text{-}tdghm \mathfrak{N}(\mathcal{NTDGCod}) = op\text{-}dg (\mathfrak{N}(\mathcal{NTDGCod}))$   
 ⟨proof⟩

### Further properties

**lemma** (in *is-tdghm*) *is-tdghm-op*:

$op\text{-}tdghm \mathfrak{N} : op\text{-}dghm \mathfrak{G} \mapsto_{DGHM} op\text{-}dghm \mathfrak{F} : op\text{-}dg \mathfrak{A} \mapsto_{DG\alpha} op\text{-}dg \mathfrak{B}$   
 ⟨proof⟩

**lemma** (in *is-tdghm*) *is-tdghm-op'*[*dg-op-intros*]:

**assumes**  $\mathfrak{G}' = op\text{-}dghm \mathfrak{G}$   
**and**  $\mathfrak{F}' = op\text{-}dghm \mathfrak{F}$   
**and**  $\mathfrak{A}' = op\text{-}dg \mathfrak{A}$   
**and**  $\mathfrak{B}' = op\text{-}dg \mathfrak{B}$   
**shows**  $op\text{-}tdghm \mathfrak{N} : \mathfrak{G}' \mapsto_{DGHM} \mathfrak{F}' : \mathfrak{A}' \mapsto_{DG\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $is\text{-}tdghm\text{-}op[*dg-op-intros*] = is\text{-}tdghm.is\text{-}tdghm\text{-}op'$

**lemma** (in *is-tdghm*) *tdghm-op-tdghm-op-tdghm*[*dg-op-simps*]:

$op\text{-}tdghm (op\text{-}tdghm \mathfrak{N}) = \mathfrak{N}$   
 ⟨proof⟩

**lemmas**  $tdghm\text{-}op\text{-}tdghm\text{-}op\text{-}tdghm[*dg-op-simps*] = is\text{-}tdghm.tdghm\text{-}op\text{-}tdghm\text{-}op\text{-}tdghm$

**lemma** *eq-op-tdghm-iff*:

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG\alpha} \mathfrak{B}'$   
**shows**  $op\text{-}tdghm \mathfrak{N} = op\text{-}tdghm \mathfrak{N}' \iff \mathfrak{N} = \mathfrak{N}'$   
 ⟨proof⟩

## 3.6.4 Composition of a transformation of digraph homomorphisms and a digraph homomorphism

### Definition and elementary properties

**definition**  $tdghm\text{-}dghm\text{-}comp :: V \Rightarrow V \Rightarrow V$  (infixl ⟨*TDGHM-DGHM*⟩ 55)

**where**  $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} =$

$[$   
 $(\lambda a \in \circ \mathfrak{H}(\mathcal{HomDom})(\mathcal{Obj}). \mathfrak{N}(\mathcal{NTMap})(\mathfrak{H}(\mathcal{ObjMap})(a))),$   
 $\mathfrak{N}(\mathcal{NTDom}) \circ_{DGHM} \mathfrak{H},$   
 $\mathfrak{N}(\mathcal{NTCod}) \circ_{DGHM} \mathfrak{H},$   
 $\mathfrak{H}(\mathcal{HomDom}),$   
 $\mathfrak{N}(\mathcal{NTDGCod})$   
 $]$ .

Components.

**lemma** *tdghm-dghm-comp-components*:

**shows**  $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{NTMap}) =$   
 $(\lambda a \in \circ \mathfrak{H}(\mathcal{HomDom})(\mathcal{Obj}). \mathfrak{N}(\mathcal{NTMap})(\mathfrak{H}(\mathcal{ObjMap})(a)))$   
**and** [*dg-shared-cs-simps*, *dg-cs-simps*]:  
 $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{NTDom}) = \mathfrak{N}(\mathcal{NTDom}) \circ_{DGHM} \mathfrak{H}$   
**and** [*dg-shared-cs-simps*, *dg-cs-simps*]:  
 $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{NTCod}) = \mathfrak{N}(\mathcal{NTCod}) \circ_{DGHM} \mathfrak{H}$   
**and** [*dg-shared-cs-simps*, *dg-cs-simps*]:

$(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{N}TDGDom) = \mathfrak{H}(\mathcal{H}omDom)$   
**and**  $[dg-shared-cs-simps, dg-cs-simps]:$   
 $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{N}TDGCod) = \mathfrak{N}(\mathcal{N}TDGCod)$   
 $\langle proof \rangle$

### Transformation map

**mk-VLambda**  $tdghm-dghm-comp-components(1)$   
 $|vsu\ tdghm-dghm-comp-NTMap-vsuv[dg-shared-cs-intros, dg-cs-intros]|$

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

**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**  $\mathcal{R}_\circ ((\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H})(\mathcal{N}TMap)) \subseteq_\circ \mathfrak{C}(\mathcal{A}rr)$   
 $\langle proof \rangle$

### Opposite of the composition of a transformation of digraph homomorphisms and a digraph homomorphism

**lemma**  $op-tdghm-tdghm-dghm-comp[dg-op-simps]:$   
 $op-tdghm (\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H}) = op-tdghm \mathfrak{N} \circ_{TDGHM-DGHM} op-dghm \mathfrak{H}$   
 $\langle proof \rangle$

### 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**  $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F} \circ_{DGHM} \mathfrak{H} \mapsto_{DGHM} \mathfrak{G} \circ_{DGHM} \mathfrak{H} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$   
 $\langle proof \rangle$

**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**  $\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$   
 $\langle proof \rangle$

### Further properties

**lemma**  $tdghm-dghm-comp-tdghm-dghm-comp-assoc:$   
**assumes**  $\mathfrak{N} : \mathfrak{H} \mapsto_{DGHM} \mathfrak{H}' : \mathfrak{C} \mapsto\mapsto_{DG\alpha} \mathcal{D}$   
**and**  $\mathfrak{G} : \mathfrak{B} \mapsto\mapsto_{DG\alpha} \mathfrak{C}$   
**and**  $\mathfrak{F} : \mathfrak{A} \mapsto\mapsto_{DG\alpha} \mathfrak{B}$   
**shows**  $(\mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{G}) \circ_{TDGHM-DGHM} \mathfrak{F} = \mathfrak{N} \circ_{TDGHM-DGHM} (\mathfrak{G} \circ_{DGHM} \mathfrak{F})$   
 $\langle proof \rangle$

**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**  $[dg\text{-}cs\text{-}simps] = is\text{-}tdghm.tdghm\text{-}tdghm\text{-}dghm\text{-}comp\text{-}dghm\text{-}id$

### 3.6.5 Composition of a digraph homomorphism and a transformation of digraph homomorphisms

#### Definition and elementary properties

**definition**  $dghm\text{-}tdghm\text{-}comp :: V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{DGHM-TDGHM} \rangle$  55)

where  $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} =$

[  
 $(\lambda a \in \mathfrak{N}(NTDGD\text{Dom}) (Obj). \mathfrak{H}(ArrMap) (\mathfrak{N}(NTMap) (a))),$   
 $\mathfrak{H} \circ_{DGHM} \mathfrak{N}(NTDom),$   
 $\mathfrak{H} \circ_{DGHM} \mathfrak{N}(NTCod),$   
 $\mathfrak{N}(NTDGD\text{Dom}),$   
 $\mathfrak{H}(HomCod)$   
 ]<sub>o</sub>

Components.

**lemma**  $dghm\text{-}tdghm\text{-}comp\text{-}components:$

**shows**  $(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTMap) =$

$(\lambda a \in \mathfrak{N}(NTDGD\text{Dom}) (Obj). \mathfrak{H}(ArrMap) (\mathfrak{N}(NTMap) (a)))$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, dg\text{-}cs\text{-}simps]:$

$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTDom) = \mathfrak{H} \circ_{DGHM} \mathfrak{N}(NTDom)$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, dg\text{-}cs\text{-}simps]:$

$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTCod) = \mathfrak{H} \circ_{DGHM} \mathfrak{N}(NTCod)$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, dg\text{-}cs\text{-}simps]:$

$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTDGD\text{Dom}) = \mathfrak{N}(NTDGD\text{Dom})$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, dg\text{-}cs\text{-}simps]:$

$(\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTDGCod) = \mathfrak{H}(HomCod)$

*<proof>*

#### Transformation map

**mk-VLambda**  $dghm\text{-}tdghm\text{-}comp\text{-}components(1)$

$|vsu\ dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}vsu[dg\text{-}shared\text{-}cs\text{-}intros, dg\text{-}cs\text{-}intros]|$

**mk-VLambda** (**in**  $is\text{-}tdghm$ )

$dghm\text{-}tdghm\text{-}comp\text{-}components(1)$  [**where**  $\mathfrak{N}=\mathfrak{N}$ , *unfolded*  $tdghm\text{-}NTDGD\text{Dom}$ ]

$|vdomain\ dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}vdomain|$

$|app\ dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}app|$

**lemmas**  $[dg\text{-}cs\text{-}simps] =$

$is\text{-}tdghm.dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}vdomain$

$is\text{-}tdghm.dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}app$

**lemma**  $dghm\text{-}tdghm\text{-}comp\text{-}NTMap\text{-}vrangle:$

**assumes**  $\mathfrak{H} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$

**shows**  $\mathcal{R}_o ((\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N})(NTMap)) \subseteq_o \mathfrak{C}(Arr)$

*<proof>*

#### Opposite of the composition of a digraph homomorphism and a transformation of digraph homomorphisms

**lemma**  $op\text{-}tdghm\text{-}dghm\text{-}tdghm\text{-}comp[dg\text{-}op\text{-}simps]:$

$op\text{-}tdghm (\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N}) = op\text{-}dghm \mathfrak{H} \circ_{DGHM-TDGHM} op\text{-}tdghm \mathfrak{N}$

*<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**  $\mathfrak{H} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$  **and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$

**shows**  $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{H} \circ_{DGHM} \mathfrak{F} \mapsto_{DGHM} \mathfrak{H} \circ_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{C}$

*<proof>*

**lemma** *dghm-tdghm-comp-is-tdghm'[dg-cs-intros]*:

**assumes**  $\mathfrak{H} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$

**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$

**and**  $\mathfrak{F}' = \mathfrak{H} \circ_{DGHM} \mathfrak{F}$

**and**  $\mathfrak{G}' = \mathfrak{H} \circ_{DGHM} \mathfrak{G}$

**shows**  $\mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{C}$

*<proof>*

### Further properties

**lemma** *dghm-comp-dghm-tdghm-comp-assoc*:

**assumes**  $\mathfrak{N} : \mathfrak{H} \mapsto_{DGHM} \mathfrak{H}' : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$

**and**  $\mathfrak{F} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$

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

**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**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : \mathfrak{B} \mapsto_{DG\alpha} \mathfrak{C}$

**and**  $\mathfrak{H} : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{D}$

**and**  $\mathfrak{K} : \mathfrak{A} \mapsto_{DG\alpha} \mathfrak{B}$

**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* =

$Z \alpha +$

$NTDom: is-tm-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$

$NTCod: is-tm-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{G} +$

$is-tdghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

for  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

syntax *-is-tm-tdghm* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$   
 $\langle \langle (- \cdot / - \mapsto_{DGHM.tm} - \cdot / - \mapsto_{DG.tm} -) \rangle [51, 51, 51, 51, 51] 51 \rangle$

syntax-consts *-is-tm-tdghm*  $\equiv is-tm-tdghm$

translations  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \Rightarrow$   
 $CONST is-tm-tdghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

abbreviation *all-tm-tdghms* ::  $V \Rightarrow V$

where *all-tm-tdghms*  $\alpha \equiv$

$set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

abbreviation *tm-tdghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *tm-tdghms*  $\alpha \mathfrak{A} \mathfrak{B} \equiv$

$set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

abbreviation *these-tm-tdghms* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

where *these-tm-tdghms*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \equiv$

$set \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

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  $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG.tm} \mathfrak{B}'$   
 $\langle proof \rangle$

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*:  $\mathfrak{N}(\mathfrak{NTMap}) \in_o Vset \alpha$   
 $\langle proof \rangle$

lemma *small-all-tm-tdghms*[*simp*]:

$small \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

$\langle proof \rangle$

lemma *small-tm-tdghms*[*simp*]:

$small \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

$\langle proof \rangle$

lemma *small-these-tm-tdghms*[*simp*]:

$small \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tm} \mathfrak{B} \}$

$\langle proof \rangle$

Further elementary results.

**lemma** *these-tm-tdghms-iff*:

$$\begin{aligned} \mathfrak{N} \in_0 \text{ these-tm-tdghms } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} &\leftrightarrow \\ \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} & \\ \langle \text{proof} \rangle & \end{aligned}$$

### Opposite transformation of digraph homomorphisms with tiny maps

**lemma** (in *is-tm-tdghm*) *is-tm-tdghm-op*:

$$\begin{aligned} \text{op-tdghm } \mathfrak{N} : \text{op-dghm } \mathfrak{G} \mapsto_{DGHM.tm} \text{op-dghm } \mathfrak{F} : \text{op-dg } \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \text{op-dg } \mathfrak{B} & \\ \langle \text{proof} \rangle & \end{aligned}$$

**lemma** (in *is-tm-tdghm*) *is-tm-tdghm-op'*[*dg-op-intros*]:

$$\begin{aligned} \text{assumes } \mathfrak{G}' = \text{op-dghm } \mathfrak{G} & \\ \text{and } \mathfrak{F}' = \text{op-dghm } \mathfrak{F} & \\ \text{and } \mathfrak{A}' = \text{op-dg } \mathfrak{A} & \\ \text{and } \mathfrak{B}' = \text{op-dg } \mathfrak{B} & \\ \text{shows } \text{op-tdghm } \mathfrak{N} : \mathfrak{G}' \mapsto_{DGHM.tm} \mathfrak{F}' : \mathfrak{A}' \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B}' & \\ \langle \text{proof} \rangle & \end{aligned}$$

**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*:

$$\begin{aligned} \text{assumes } \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{B} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} \text{ and } \mathfrak{H} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} & \\ \text{shows } \mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F} \circ_{DGHM} \mathfrak{H} \mapsto_{DGHM.tm} \mathfrak{G} \circ_{DGHM} \mathfrak{H} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \langle \text{proof} \rangle & \end{aligned}$$

**lemma** *tdghm-dghm-comp-is-tm-tdghm'*[*dg-small-cs-intros*]:

$$\begin{aligned} \text{assumes } \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{B} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \text{and } \mathfrak{H} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} & \\ \text{and } \mathfrak{F}' = \mathfrak{F} \circ_{DGHM} \mathfrak{H} & \\ \text{and } \mathfrak{G}' = \mathfrak{G} \circ_{DGHM} \mathfrak{H} & \\ \text{shows } \mathfrak{N} \circ_{TDGHM-DGHM} \mathfrak{H} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \langle \text{proof} \rangle & \end{aligned}$$

### Composition of a digraph homomorphism with tiny maps and a transformation of digraph homomorphisms with tiny maps

**lemma** *dghm-tdghm-comp-is-tm-tdghm*:

$$\begin{aligned} \text{assumes } \mathfrak{H} : \mathfrak{B} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} \text{ and } \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} & \\ \text{shows } \mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{H} \circ_{DGHM} \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{H} \circ_{DGHM} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \langle \text{proof} \rangle & \end{aligned}$$

**lemma** *dghm-tdghm-comp-is-tm-tdghm'*[*dg-small-cs-intros*]:

$$\begin{aligned} \text{assumes } \mathfrak{H} : \mathfrak{B} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \text{and } \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tm} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{B} & \\ \text{and } \mathfrak{F}' = \mathfrak{H} \circ_{DGHM} \mathfrak{F} & \\ \text{and } \mathfrak{G}' = \mathfrak{H} \circ_{DGHM} \mathfrak{G} & \\ \text{shows } \mathfrak{H} \circ_{DGHM-TDGHM} \mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \mathfrak{C} & \\ \langle \text{proof} \rangle & \end{aligned}$$

## 3.7.2 Transformation of homomorphisms of tiny digraphs

### Definition and elementary properties

locale *is-tiny-tdghm* =

$Z \alpha +$   
 $NTDom: is-tiny-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$   
 $NTCod: is-tiny-dghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{G} +$   
 $is-tdghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$   
**for**  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

**syntax**  $-is-tiny-tdghm :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$   
 $(\langle (- / - \mapsto_{DGHM.tiny} - / - \mapsto_{DG.tiny} -) \rangle [51, 51, 51, 51, 51] 51)$   
**syntax-consts**  $-is-tiny-tdghm \equiv is-tiny-tdghm$   
**translations**  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \equiv$   
 $CONST is-tiny-tdghm \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

**abbreviation**  $all-tiny-tdghms :: V \Rightarrow V$   
**where**  $all-tiny-tdghms \alpha \equiv$   
 $set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$

**abbreviation**  $tiny-tdghms :: V \Rightarrow V \Rightarrow V \Rightarrow V$   
**where**  $tiny-tdghms \alpha \mathfrak{A} \mathfrak{B} \equiv$   
 $set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$

**abbreviation**  $these-tiny-tdghms :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$   
**where**  $these-tiny-tdghms \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \equiv$   
 $set \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$

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**  $is-tiny-tdghm \subseteq is-tm-tdghm$   
 $\langle proof \rangle$

**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  $is-tiny-tdghm$ )  $tiny-tdghm-in-Vset: \mathfrak{N} \in_o Vset \alpha$   
 $\langle proof \rangle$

**lemma**  $small-all-tiny-tdghms[simp]:$   
 $small \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$   
 $\langle proof \rangle$

**lemma**  $small-tiny-tdghms[simp]:$   
 $small \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$   
 $\langle proof \rangle$

**lemma**  $small-these-tiny-tdghms[simp]:$   
 $small \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny} \mathfrak{B} \}$   
 $\langle proof \rangle$

**lemma** *tiny-tdghms-vsubset-Vset[simp]*:  
 set  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{B}\} \subseteq_o Vset \alpha$   
 (is  $\langle set \ ?tdghms \subseteq_o \ - \rangle$ )  
 $\langle proof \rangle$

**lemma** (in *is-tdghm*) *tdghm-is-tiny-tdghm-if-ge-Limit*:  
 assumes  $Z \beta$  and  $\alpha \in_o \beta$   
 shows  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny\beta} \mathfrak{B}$   
 $\langle proof \rangle$

Further elementary results.

**lemma** *these-tiny-tdghms-iff*:  
 $\mathfrak{N} \in_o these-tiny-tdghms \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \longleftrightarrow$   
 $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{DG.tiny\alpha} \mathfrak{B}$   
 $\langle proof \rangle$

### Opposite transformation of homomorphisms of tiny digraphs

**lemma** (in *is-tiny-tdghm*) *is-tm-tdghm-op*: *op-tdghm*  $\mathfrak{N} :$   
 $op-dghm \mathfrak{G} \mapsto_{DGHM.tiny} op-dghm \mathfrak{F} : op-dg \mathfrak{A} \mapsto_{DG.tiny\alpha} op-dg \mathfrak{B}$   
 $\langle proof \rangle$

**lemma** (in *is-tiny-tdghm*) *is-tiny-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*  $\mathfrak{N} : \mathfrak{G}' \mapsto_{DGHM.tiny} \mathfrak{F}' : \mathfrak{A}' \mapsto_{DG.tiny\alpha} \mathfrak{B}'$   
 $\langle proof \rangle$

**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*  $I \mathfrak{A} =$

[  
 ( $\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Obj})$ ),  
 ( $\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})$ ),  
 ( $\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})). (\lambda i \in_{\circ} I. \mathfrak{A} i(\text{Dom})(f(i)))$ ),  
 ( $\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})). (\lambda i \in_{\circ} I. \mathfrak{A} i(\text{Cod})(f(i)))$ )  
 ]<sub>o</sub>

**syntax** *-PDIGRAPH* :: *pttrn*  $\Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

( $\langle \langle 3 \prod_{DG-\epsilon_{\circ}} \cdot / \cdot \rangle \rangle [0, 0, 10] 10$ )

**syntax-consts** *-PDIGRAPH*  $\hat{=} dg-prod$

**translations**  $\prod_{DG} i \in_{\circ} I. \mathfrak{A} \hat{=} CONST dg-prod I (\lambda i. \mathfrak{A})$

Components.

**lemma** *dg-prod-components*:

**shows** ( $\prod_{DG} i \in_{\circ} I. \mathfrak{A} i(\text{Obj})$ ) = ( $\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Obj})$ )

**and** ( $\prod_{DG} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})$ ) = ( $\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})$ )

**and** ( $\prod_{DG} i \in_{\circ} I. \mathfrak{A} i(\text{Dom})$ ) =

( $\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})). (\lambda i \in_{\circ} I. \mathfrak{A} i(\text{Dom})(f(i)))$ )

**and** ( $\prod_{DG} i \in_{\circ} I. \mathfrak{A} i(\text{Cod})$ ) =

( $\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})). (\lambda i \in_{\circ} I. \mathfrak{A} i(\text{Cod})(f(i)))$ )

*<proof>*

### 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*]:  $i \in_{\circ} I \implies digraph \alpha (\mathfrak{A} i)$

**and** *pdg-index-in-Vset*[*dg-cs-intros*]:  $I \in_{\circ} Vset \alpha$

Rules.

**lemma** (**in** *pdigraph-base*) *pdigraph-base-axioms'*[*dg-prod-cs-intros*]:

**assumes**  $\alpha' = \alpha$  **and**  $I' = I$

**shows** *pdigraph-base*  $\alpha' I' \mathfrak{A}$

*<proof>*

**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** ( $\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Obj})$ )  $\in_{\circ} Vset \beta$

*<proof>*

**lemma** (in *pdigraph-base*) *pdg-Arr-in-Vset*:  
**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $(\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i(\text{Arr})) \in_{\circ} \text{Vset } \beta$   
*<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-ubset-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**  $\wedge i. i \in_{\circ} I \implies a(i) \in_{\circ} \mathfrak{A} i(\text{Obj})$   
**shows**  $a \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Obj})$   
*<proof>*

**lemma** *dg-prod-ObjD*:  
**assumes**  $a \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Obj})$   
**shows** *vsv a* **and**  $\mathcal{D}_{\circ} a = I$  **and**  $\wedge i. i \in_{\circ} I \implies a(i) \in_{\circ} \mathfrak{A} i(\text{Obj})$   
*<proof>*

**lemma** *dg-prod-ObjE*:  
**assumes**  $a \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Obj})$   
**obtains** *vsv a* **and**  $\mathcal{D}_{\circ} a = I$  **and**  $\wedge i. i \in_{\circ} I \implies a(i) \in_{\circ} \mathfrak{A} i(\text{Obj})$   
*<proof>*

**lemma** *dg-prod-Obj-cong*:  
**assumes**  $g \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Obj})$   
**and**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Obj})$   
**and**  $\wedge i. i \in_{\circ} I \implies g(i) = f(i)$   
**shows**  $g = f$   
*<proof>*

## Arrow

**lemma** *dg-prod-ArrI*:  
**assumes** *vsv f* **and**  $\mathcal{D}_{\circ} f = I$  **and**  $\wedge i. i \in_{\circ} I \implies f(i) \in_{\circ} \mathfrak{A} i(\text{Arr})$   
**shows**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Arr})$   
*<proof>*

**lemma** *dg-prod-ArrD*:  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Arr})$   
**shows** *vsv f* **and**  $\mathcal{D}_{\circ} f = I$  **and**  $\wedge i. i \in_{\circ} I \implies f(i) \in_{\circ} \mathfrak{A} i(\text{Arr})$   
*<proof>*

**lemma** *dg-prod-ArrE*:  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Arr})$   
**obtains** *vsv f* **and**  $\mathcal{D}_{\circ} f = I$  **and**  $\wedge i. i \in_{\circ} I \implies f(i) \in_{\circ} \mathfrak{A} i(\text{Arr})$   
*<proof>*

**lemma** *dg-prod-Arr-cong*:  
**assumes**  $g \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)(\text{Arr})$

**and**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Arr)$   
**and**  $\wedge i. i \in_{\circ} I \implies g(\downarrow i) = f(\downarrow i)$   
**shows**  $g = f$   
 $\langle proof \rangle$

### Domain

**mk-VLambda**  $dg\text{-prod-components}(3)$   
 $| vsv\ dg\text{-prod-Dom-vsuv}[dg\text{-cs-intros}] |$   
 $| vdomain\ dg\text{-prod-Dom-vdomain}[folded\ dg\text{-prod-components},\ dg\text{-cs-simps}] |$   
 $| app\ dg\text{-prod-Dom-app}[folded\ dg\text{-prod-components}] |$

**lemma** (in  $pdigraph\text{-base}$ )  $dg\text{-prod-Dom-app-in-Obj}[dg\text{-cs-intros}]$ :  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Arr)$   
**shows**  $(\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Dom) (\downarrow f) \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Obj)$   
 $\langle proof \rangle$

**lemma**  $dg\text{-prod-Dom-app-component-app}[dg\text{-cs-simps}]$ :  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Arr)$  **and**  $i \in_{\circ} I$   
**shows**  $(\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Dom) (\downarrow f) (\downarrow i) = \mathfrak{A} i (\downarrow Dom) (\downarrow f(\downarrow i))$   
 $\langle proof \rangle$

### Codomain

**mk-VLambda**  $dg\text{-prod-components}(4)$   
 $| vsv\ dg\text{-prod-Cod-vsuv}[dg\text{-cs-intros}] |$   
 $| vdomain\ dg\text{-prod-Cod-vdomain}[folded\ dg\text{-prod-components},\ dg\text{-cs-simps}] |$   
 $| app\ dg\text{-prod-Cod-app}[folded\ dg\text{-prod-components}] |$

**lemma** (in  $pdigraph\text{-base}$ )  $dg\text{-prod-Cod-app-in-Obj}[dg\text{-cs-intros}]$ :  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Arr)$   
**shows**  $(\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Cod) (\downarrow f) \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Obj)$   
 $\langle proof \rangle$

**lemma**  $dg\text{-prod-Cod-app-component-app}[dg\text{-cs-simps}]$ :  
**assumes**  $f \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Arr)$  **and**  $i \in_{\circ} I$   
**shows**  $(\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\downarrow Cod) (\downarrow f) (\downarrow i) = \mathfrak{A} i (\downarrow Cod) (\downarrow f(\downarrow i))$   
 $\langle proof \rangle$

### A product $\alpha$ -digraph is a tiny $\beta$ -digraph

**lemma** (in  $pdigraph\text{-base}$ )  $pdg\text{-tiny-digraph-dg-prod}$ :  
**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $tiny\text{-digraph} \beta (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)$   
 $\langle proof \rangle$

**lemma** (in  $pdigraph\text{-base}$ )  $pdg\text{-tiny-digraph-dg-prod}'$ :  
 $tiny\text{-digraph} (\alpha + \omega) (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i)$   
 $\langle proof \rangle$

### Arrow with a domain and a codomain

**lemma** (in  $pdigraph\text{-base}$ )  $dg\text{-prod-is-arrI}$ :  
**assumes**  $vsu\ f$   
**and**  $\mathcal{D}_{\circ} f = I$   
**and**  $vsu\ a$   
**and**  $\mathcal{D}_{\circ} a = I$   
**and**  $vsu\ b$

**and**  $\mathcal{D}_\circ b = I$   
**and**  $\bigwedge i. i \in_\circ I \implies f(|i) : a(|i) \mapsto_{\mathfrak{A} i} b(|i)$   
**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**  $\bigwedge i. i \in_\circ I \implies f(|i) : a(|i) \mapsto_{\mathfrak{A} i} b(|i)$   
 ⟨*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**  $\bigwedge i. i \in_\circ I \implies f(|i) : a(|i) \mapsto_{\mathfrak{A} i} b(|i)$   
 ⟨*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*:  $J \subseteq_\circ I \implies (\prod_{DG} i \in_\circ J. \mathfrak{A} i)(|Obj) \subseteq_\circ Vset \alpha$   
**and** *pdg-Hom-vifunion-in-Vset*:  
 [[  
    $J \subseteq_\circ I$ ;  
    $A \subseteq_\circ (\prod_{DG} i \in_\circ J. \mathfrak{A} i)(|Obj)$ ;  
    $B \subseteq_\circ (\prod_{DG} i \in_\circ J. \mathfrak{A} i)(|Obj)$ ;  
    $A \in_\circ Vset \alpha$ ;  
    $B \in_\circ Vset \alpha$   
 ]]  $\implies (\bigcup_\circ a \in_\circ A. \bigcup_\circ b \in_\circ B. Hom (\prod_{DG} i \in_\circ J. \mathfrak{A} i) a b) \in_\circ Vset \alpha$

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**  $(\cup_{\circ} a \in_{\circ} A. \cup_{\circ} b \in_{\circ} B. Hom (\prod_{DG i \in_{\circ} I. \mathfrak{A} i} a b) \in_{\circ} Vset \alpha$   
*<proof>*

**lemma** (in *pdigraph*) *pdg-vsubset-index-pdigraph*:  
**assumes**  $J \subseteq_{\circ} I$   
**shows** *pdigraph*  $\alpha J \mathfrak{A}$   
*<proof>*

**A product  $\alpha$ -digraph is an  $\alpha$ -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  $\subseteq$  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-uset* *subset* *assms*,  
     *unfolded vunion-complement*[*OF* *subset*]  
   ]  
 )

## Results

**lemma** *dg-prod-vunion-Obj-in-Obj*:  
**assumes** *vdisjnt* *J* *K*  
**and**  $b \in_{\circ} (\prod_{DG} j \in_{\circ} J. \mathfrak{A} j) (\text{Obj})$   
**and**  $c \in_{\circ} (\prod_{DG} k \in_{\circ} K. \mathfrak{A} k) (\text{Obj})$   
**shows**  $b \cup_{\circ} c \in_{\circ} (\prod_{DG} i \in_{\circ} J \cup_{\circ} K. \mathfrak{A} i) (\text{Obj})$   
*<proof>*

**lemma** *dg-prod-vdiff-vunion-Obj-in-Obj*:  
**assumes**  $J \subseteq_{\circ} I$   
**and**  $b \in_{\circ} (\prod_{DG} k \in_{\circ} I -_{\circ} J. \mathfrak{A} k) (\text{Obj})$   
**and**  $c \in_{\circ} (\prod_{DG} j \in_{\circ} J. \mathfrak{A} j) (\text{Obj})$   
**shows**  $b \cup_{\circ} c \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\text{Obj})$   
*<proof>*

**lemma** *dg-prod-vunion-Arr-in-Arr*:  
**assumes** *vdisjnt* *J* *K*  
**and**  $b \in_{\circ} (\prod_{DG} j \in_{\circ} J. \mathfrak{A} j) (\text{Arr})$   
**and**  $c \in_{\circ} (\prod_{DG} k \in_{\circ} K. \mathfrak{A} k) (\text{Arr})$   
**shows**  $b \cup_{\circ} c \in_{\circ} (\prod_{DG} i \in_{\circ} J \cup_{\circ} K. \mathfrak{A} i) (\text{Arr})$   
*<proof>*

**lemma** *dg-prod-vdiff-vunion-Arr-in-Arr*:  
**assumes**  $J \subseteq_{\circ} I$   
**and**  $b \in_{\circ} (\prod_{DG} k \in_{\circ} I -_{\circ} J. \mathfrak{A} k) (\text{Arr})$   
**and**  $c \in_{\circ} (\prod_{DG} j \in_{\circ} J. \mathfrak{A} j) (\text{Arr})$   
**shows**  $b \cup_{\circ} c \in_{\circ} (\prod_{DG} i \in_{\circ} I. \mathfrak{A} i) (\text{Arr})$   
*<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**  $dghm\text{-proj} :: V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V \langle \pi_{DG} \rangle$

**where**  $\pi_{DG} I \mathfrak{A} i =$   
 $[$   
 $(\lambda a \in \circ ((\prod_{DG} i \in \circ I. \mathfrak{A} i) \langle Obj \rangle)). a \langle i \rangle,$   
 $(\lambda f \in \circ ((\prod_{DG} i \in \circ I. \mathfrak{A} i) \langle Arr \rangle)). f \langle i \rangle,$   
 $(\prod_{DG} i \in \circ I. \mathfrak{A} i),$   
 $\mathfrak{A} i$   
 $]$

Components.

**lemma**  $dghm\text{-proj-components}$ :

**shows**  $\pi_{DG} I \mathfrak{A} i \langle ObjMap \rangle = (\lambda a \in \circ ((\prod_{DG} i \in \circ I. \mathfrak{A} i) \langle Obj \rangle)). a \langle i \rangle$   
**and**  $\pi_{DG} I \mathfrak{A} i \langle ArrMap \rangle = (\lambda f \in \circ ((\prod_{DG} i \in \circ I. \mathfrak{A} i) \langle Arr \rangle)). f \langle i \rangle$   
**and**  $\pi_{DG} I \mathfrak{A} i \langle HomDom \rangle = (\prod_{DG} i \in \circ I. \mathfrak{A} i)$   
**and**  $\pi_{DG} I \mathfrak{A} i \langle HomCod \rangle = \mathfrak{A} i$   
 $\langle proof \rangle$

Object map.

**mk-VLambda**  $dghm\text{-proj-components}(1)$   
 $[vsu\ dghm\text{-proj-ObjMap-vsuv}[dg\text{-cs-intros}]$   
 $[vdomain\ dghm\text{-proj-ObjMap-vdomain}[dg\text{-cs-simps}]$   
 $[app\ dghm\text{-proj-ObjMap-app}[dg\text{-cs-simps}]$

**lemma** (in  $pdigraph$ )  $dghm\text{-proj-ObjMap-vrange}$ :

**assumes**  $i \in \circ I$   
**shows**  $\mathcal{R}_\circ (\pi_{DG} I \mathfrak{A} i \langle ObjMap \rangle) \subseteq_\circ \mathfrak{A} i \langle Obj \rangle$   
 $\langle proof \rangle$

Arrow map.

**mk-VLambda**  $dghm\text{-proj-components}(2)$   
 $[vsu\ dghm\text{-proj-ArrMap-vsuv}[dg\text{-cs-intros}]$   
 $[vdomain\ dghm\text{-proj-ArrMap-vdomain}[dg\text{-cs-simps}]$   
 $[app\ dghm\text{-proj-ArrMap-app}[dg\text{-cs-simps}]$

**lemma** (in  $pdigraph$ )  $dghm\text{-proj-ArrMap-vrange}$ :

**assumes**  $i \in \circ I$   
**shows**  $\mathcal{R}_\circ (\pi_{DG} I \mathfrak{A} i \langle ArrMap \rangle) \subseteq_\circ \mathfrak{A} i \langle Arr \rangle$   
 $\langle proof \rangle$

#### A projection digraph homomorphism is a digraph homomorphism

**lemma** (in  $pdigraph$ )  $pdg\text{-dghm-proj-is-dghm}$ :

**assumes**  $i \in \circ I$   
**shows**  $\pi_{DG} I \mathfrak{A} i : (\prod_{DG} i \in \circ I. \mathfrak{A} i) \mapsto_{DG\alpha} \mathfrak{A} i$   
 $\langle proof \rangle$

**lemma** (in  $pdigraph$ )  $pdg\text{-dghm-proj-is-dghm}'$ :

**assumes**  $i \in \circ I$  **and**  $\mathfrak{C} = (\prod_{DG} i \in \circ I. \mathfrak{A} i)$  **and**  $\mathfrak{D} = \mathfrak{A} i$   
**shows**  $\pi_{DG} I \mathfrak{A} i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{D}$   
 $\langle proof \rangle$

**lemmas**  $[dg\text{-cs-intros}] = pdigraph.pdg\text{-dghm-proj-is-dghm}'$

### 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  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi =$

$$\begin{aligned} & [ \\ & (\lambda a \in_o \mathfrak{C}(\mathfrak{Obj}). (\lambda i \in_o I. \varphi i(\mathfrak{ObjMap})(\mathfrak{!}a))), \\ & (\lambda f \in_o \mathfrak{C}(\mathfrak{Arr}). (\lambda i \in_o I. \varphi i(\mathfrak{ArrMap})(\mathfrak{!}f))), \\ & \mathfrak{C}, \\ & (\prod_{DG} i \in_o I. \mathfrak{A} i) \\ & ]_o \end{aligned}$$

Components.

**lemma**  $dghm-up-components$ :

**shows**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ObjMap}) = (\lambda a \in_o \mathfrak{C}(\mathfrak{Obj}). (\lambda i \in_o I. \varphi i(\mathfrak{ObjMap})(\mathfrak{!}a)))$

**and**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ArrMap}) = (\lambda f \in_o \mathfrak{C}(\mathfrak{Arr}). (\lambda i \in_o I. \varphi i(\mathfrak{ArrMap})(\mathfrak{!}f)))$

**and**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{HomDom}) = \mathfrak{C}$

**and**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{HomCod}) = (\prod_{DG} i \in_o I. \mathfrak{A} i)$

$\langle proof \rangle$

#### Object map

**mk-VLambda**  $dghm-up-components(1)$

$|vsu dghm-up-ObjMap-vsuv[dg-cs-intros]|$

$|vdomain dghm-up-ObjMap-vdomain[dg-cs-simps]|$

$|app dghm-up-ObjMap-app|$

**lemma**  $dghm-up-ObjMap-vrange$ :

**assumes**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$

**shows**  $\mathcal{R}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ObjMap})) \subseteq_o (\prod_{DG} i \in_o I. \mathfrak{A} i)(\mathfrak{Obj})$

$\langle proof \rangle$

**lemma**  $dghm-up-ObjMap-app-vdomain[dg-cs-simps]$ :

**assumes**  $a \in_o \mathfrak{C}(\mathfrak{Obj})$

**shows**  $\mathcal{D}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ObjMap})(\mathfrak{!}a)) = I$

$\langle proof \rangle$

**lemma**  $dghm-up-ObjMap-app-component[dg-cs-simps]$ :

**assumes**  $a \in_o \mathfrak{C}(\mathfrak{Obj})$  **and**  $i \in_o I$

**shows**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ObjMap})(\mathfrak{!}a)(\mathfrak{!}i) = \varphi i(\mathfrak{ObjMap})(\mathfrak{!}a)$

$\langle proof \rangle$

**lemma**  $dghm-up-ObjMap-app-vrange$ :

**assumes**  $a \in_o \mathfrak{C}(\mathfrak{Obj})$  **and**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$

**shows**  $\mathcal{R}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{ObjMap})(\mathfrak{!}a)) \subseteq_o (\bigcup_o i \in_o I. \mathfrak{A} i)(\mathfrak{Obj})$

$\langle proof \rangle$

#### Arrow map

**mk-VLambda**  $dghm-up-components(2)$

$|vsu dghm-up-ArrMap-vsuv[dg-cs-intros]|$

$|vdomain dghm-up-ArrMap-vdomain[dg-cs-simps]|$

$|app dghm-up-ArrMap-app|$

**lemma** (in  $pdigraph$ )  $dghm-up-ArrMap-vrange$ :

**assumes**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(ArrMap)) \subseteq_o (\prod_{DG i \in_o I. \mathfrak{A} i}(Arr))$   
*<proof>*

**lemma** *dghm-up-ArrMap-vrange:*

**assumes**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(ArrMap)) \subseteq_o (\prod_{DG i \in_o I. \mathfrak{A} i}(Arr))$   
*<proof>*

**lemma** *dghm-up-ArrMap-app-vdomain[dg-cs-simps]:*

**assumes**  $a \in_o \mathfrak{C}(Arr)$   
**shows**  $\mathcal{D}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(ArrMap)(a)) = I$   
*<proof>*

**lemma** *dghm-up-ArrMap-app-component[dg-cs-simps]:*

**assumes**  $a \in_o \mathfrak{C}(Arr)$  **and**  $i \in_o I$   
**shows**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi(ArrMap)(a)(i) = \varphi i(ArrMap)(a)$   
*<proof>*

**lemma** *dghm-up-ArrMap-app-vrange:*

**assumes**  $a \in_o \mathfrak{C}(Arr)$  **and**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o (dghm-up I \mathfrak{A} \mathfrak{C} \varphi(ArrMap)(a)) \subseteq_o (\bigcup_o i \in_o I. \mathfrak{A} i(Arr))$   
*<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**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$   
**shows**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi : \mathfrak{C} \mapsto_{DG\alpha} (\prod_{DG i \in_o I. \mathfrak{A} i})$   
*<proof>*

**Further properties**

**lemma** (in *pdigraph*) *pdg-dghm-comp-dghm-proj-dghm-up:*

**assumes** *digraph*  $\alpha \mathfrak{C}$   
**and**  $\wedge i. i \in_o I \implies \varphi i : \mathfrak{C} \mapsto_{DG\alpha} \mathfrak{A} i$   
**and**  $i \in_o I$   
**shows**  $\varphi i = \pi_{DG} I \mathfrak{A} i \circ_{DGHM} dghm-up I \mathfrak{A} \mathfrak{C} \varphi$   
*<proof>*

**lemma** (in *pdigraph*) *pdg-dghm-up-eq-dghm-proj:*

**assumes**  $\mathfrak{F} : \mathfrak{C} \mapsto_{DG\alpha} (\prod_{DG i \in_o I. \mathfrak{A} i})$   
**and**  $\wedge i. i \in_o I \implies \varphi i = \pi_{DG} I \mathfrak{A} i \circ_{DGHM} \mathfrak{F}$   
**shows**  $dghm-up I \mathfrak{A} \mathfrak{C} \varphi = \mathfrak{F}$   
*<proof>*

### 3.8.9 Singleton digraph

**Object**

**lemma** *dg-singleton-ObjI:*

**assumes**  $A = set \{(j, a)\}$  **and**  $a \in_o \mathfrak{C}(Obj)$   
**shows**  $A \in_o (\prod_{DG i \in_o set \{j\}. \mathfrak{C}}(Obj))$   
*<proof>*

**lemma** *dg-singleton-ObjE:*

**assumes**  $A \in_o (\prod_{DG i \in_o set \{j\}. \mathfrak{C}}(Obj))$   
**obtains**  $a$  **where**  $A = set \{(j, a)\}$  **and**  $a \in_o \mathfrak{C}(Obj)$

*<proof>*

### Arrow

**lemma** *dg-singleton-ArrI*:

**assumes**  $F = \text{set } \{\langle j, a \rangle\}$  **and**  $a \in_{\circ} \mathfrak{C}(\text{Arr})$

**shows**  $F \in_{\circ} (\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C})(\text{Arr})$

*<proof>*

**lemma** *dg-singleton-ArrE*:

**assumes**  $F \in_{\circ} (\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C})(\text{Arr})$

**obtains**  $a$  **where**  $F = \text{set } \{\langle j, a \rangle\}$  **and**  $a \in_{\circ} \mathfrak{C}(\text{Arr})$

*<proof>*

### Singleton digraph is a digraph

**lemma** (**in** *digraph*) *dg-finite-pdigraph-dg-singleton*:

**assumes**  $j \in_{\circ} V \text{set } \alpha$

**shows** *finite-pdigraph*  $\alpha$  ( $\text{set } \{j\}$ ) ( $\lambda i. \mathfrak{C}$ )

*<proof>*

**lemma** (**in** *digraph*) *dg-digraph-dg-singleton*:

**assumes**  $j \in_{\circ} V \text{set } \alpha$

**shows** *digraph*  $\alpha$  ( $\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C}$ )

*<proof>*

### Arrow with a domain and a codomain

**lemma** (**in** *digraph*) *dg-singleton-is-arrI*:

**assumes**  $j \in_{\circ} V \text{set } \alpha$  **and**  $f : a \mapsto_{\mathfrak{C}} b$

**shows**  $\text{set } \{\langle j, f \rangle\} : \text{set } \{\langle j, a \rangle\} \mapsto (\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C}) \text{set } \{\langle j, b \rangle\}$

*<proof>*

**lemma** (**in** *digraph*) *dg-singleton-is-arrD*:

**assumes**  $\text{set } \{\langle j, f \rangle\} : \text{set } \{\langle j, a \rangle\} \mapsto (\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C}) \text{set } \{\langle j, b \rangle\}$

**and**  $j \in_{\circ} V \text{set } \alpha$

**shows**  $f : a \mapsto_{\mathfrak{C}} b$

*<proof>*

**lemma** (**in** *digraph*) *dg-singleton-is-arrE*:

**assumes**  $\text{set } \{\langle j, f \rangle\} : \text{set } \{\langle j, a \rangle\} \mapsto (\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C}) \text{set } \{\langle j, b \rangle\}$

**and**  $j \in_{\circ} V \text{set } \alpha$

**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*  $j \mathfrak{C} =$

[  
 $(\lambda a \in_{\circ} \mathfrak{C}(\text{Obj}). \text{set } \{\langle j, a \rangle\}),$   
 $(\lambda f \in_{\circ} \mathfrak{C}(\text{Arr}). \text{set } \{\langle j, f \rangle\}),$   
 $\mathfrak{C},$   
 $(\prod_{DG} j \in_{\circ} \text{set } \{j\}. \mathfrak{C})$   
 ]<sub>o</sub>

Components.

**lemma** *dghm-singleton-components*:

**shows**  $dghm-singleton\ j\ \mathfrak{C}(\mathit{ObjMap}) = (\lambda a \in_{\circ} \mathfrak{C}(\mathit{Obj}).\ set\ \{\{j, a\}\})$   
**and**  $dghm-singleton\ j\ \mathfrak{C}(\mathit{ArrMap}) = (\lambda f \in_{\circ} \mathfrak{C}(\mathit{Arr}).\ set\ \{\{j, f\}\})$   
**and**  $dghm-singleton\ j\ \mathfrak{C}(\mathit{HomDom}) = \mathfrak{C}$   
**and**  $dghm-singleton\ j\ \mathfrak{C}(\mathit{HomCod}) = (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})$   
*<proof>*

### Object map

**mk-VLambda**  $dghm-singleton-components(1)$   
 $|vsv\ dghm-singleton-ObjMap-vsuv[ dg-cs-intros ]|$   
 $|vdomain\ dghm-singleton-ObjMap-vdomain[ dg-cs-simps ]|$   
 $|app\ dghm-singleton-ObjMap-app[ dg-prod-cs-simps ]|$

**lemma**  $dghm-singleton-ObjMap-vrange[ dg-cs-simps ]:$   
 $\mathcal{R}_{\circ}\ (dghm-singleton\ j\ \mathfrak{C}(\mathit{ObjMap})) = (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})(\mathit{Obj})$   
*<proof>*

### Arrow map

**mk-VLambda**  $dghm-singleton-components(2)$   
 $|vsv\ dghm-singleton-ArrMap-vsuv[ dg-cs-intros ]|$   
 $|vdomain\ dghm-singleton-ArrMap-vdomain[ dg-cs-simps ]|$   
 $|app\ dghm-singleton-ArrMap-app[ dg-prod-cs-simps ]|$

**lemma**  $dghm-singleton-ArrMap-vrange[ dg-cs-simps ]:$   
 $\mathcal{R}_{\circ}\ (dghm-singleton\ j\ \mathfrak{C}(\mathit{ArrMap})) = (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})(\mathit{Arr})$   
*<proof>*

## Singleton digraph homomorphism is an isomorphism of digraphs

**lemma** (*in digraph*)  $dg-dghm-singleton-is-dghm:$   
**assumes**  $j \in_{\circ} Vset\ \alpha$   
**shows**  $dghm-singleton\ j\ \mathfrak{C} : \mathfrak{C} \mapsto_{DG} is_{\circ} \alpha\ (\prod_{DG} j \in_{\circ} set\ \{j\}.\ \mathfrak{C})$   
*<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*  $\langle \times_{DG} \rangle$  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**  $\alpha\ \mathfrak{A}\ \mathfrak{B}$   
**assumes**  $\mathfrak{A}: digraph\ \alpha\ \mathfrak{A}$  **and**  $\mathfrak{B}: digraph\ \alpha\ \mathfrak{B}$   
**begin**

**interpretation**  $\mathcal{Z}\ \alpha\ \langle proof \rangle$   
**interpretation**  $\mathfrak{A}: digraph\ \alpha\ \mathfrak{A}\ \langle proof \rangle$   
**interpretation**  $\mathfrak{B}: digraph\ \alpha\ \mathfrak{B}\ \langle proof \rangle$

**lemma**  $finite-pdigraph-dg-prod-2: finite-pdigraph\ \alpha\ (2_{\mathbb{N}})\ (if2\ \mathfrak{A}\ \mathfrak{B})$   
*<proof>*

**interpretation**  $finite-pdigraph\ \alpha\ \langle 2_{\mathbb{N}} \rangle\ \langle if2\ \mathfrak{A}\ \mathfrak{B} \rangle$

*<proof>*

**lemma** *digraph-dg-prod-2[ dg-cs-intros ]*: *digraph*  $\alpha$  ( $\mathfrak{A} \times_{DG} \mathfrak{B}$ )

*<proof>*

**end**

### Object

**context**

fixes  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$

assumes  $\mathfrak{A}$ : *digraph*  $\alpha$   $\mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha$   $\mathfrak{B}$

**begin**

**lemma** *dg-prod-2-ObjI*:

assumes  $a \in_{\circ} \mathfrak{A}(\text{Obj})$  and  $b \in_{\circ} \mathfrak{B}(\text{Obj})$

shows  $[a, b]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$

*<proof>*

**lemma** *dg-prod-2-ObjI'*[*dg-prod-cs-intros*]:

assumes  $ab = [a, b]_{\circ}$  and  $a \in_{\circ} \mathfrak{A}(\text{Obj})$  and  $b \in_{\circ} \mathfrak{B}(\text{Obj})$

shows  $ab \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$

*<proof>*

**lemma** *dg-prod-2-ObjE*:

assumes  $ab \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$

obtains  $a$   $b$  where  $ab = [a, b]_{\circ}$  and  $a \in_{\circ} \mathfrak{A}(\text{Obj})$  and  $b \in_{\circ} \mathfrak{B}(\text{Obj})$

*<proof>*

**end**

### Arrow

**context**

fixes  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$

assumes  $\mathfrak{A}$ : *digraph*  $\alpha$   $\mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha$   $\mathfrak{B}$

**begin**

**lemma** *dg-prod-2-ArrI*:

assumes  $g \in_{\circ} \mathfrak{A}(\text{Arr})$  and  $f \in_{\circ} \mathfrak{B}(\text{Arr})$

shows  $[g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$

*<proof>*

**lemma** *dg-prod-2-ArrI'*[*dg-prod-cs-intros*]:

assumes  $gf = [g, f]_{\circ}$  and  $g \in_{\circ} \mathfrak{A}(\text{Arr})$  and  $f \in_{\circ} \mathfrak{B}(\text{Arr})$

shows  $[g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$

*<proof>*

**lemma** *dg-prod-2-ArrE*:

assumes  $gf \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$

obtains  $g$   $f$  where  $gf = [g, f]_{\circ}$  and  $g \in_{\circ} \mathfrak{A}(\text{Arr})$  and  $f \in_{\circ} \mathfrak{B}(\text{Arr})$

*<proof>*

**end**

### Arrow with a domain and a codomain

**context**

fixes  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$

assumes  $\mathfrak{A}$ : *digraph*  $\alpha$   $\mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha$   $\mathfrak{B}$   
begin

interpretation  $\mathcal{Z} \alpha$  *<proof>*

interpretation  $\mathfrak{A}$ : *digraph*  $\alpha$   $\mathfrak{A}$  *<proof>*

interpretation  $\mathfrak{B}$ : *digraph*  $\alpha$   $\mathfrak{B}$  *<proof>*

interpretation *finite-pdigraph*  $\alpha$   $\langle 2_{\mathbb{N}} \rangle$  *<if2>*  $\mathfrak{A}$   $\mathfrak{B}$   
*<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  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$

assumes  $\mathfrak{A}$ : *digraph*  $\alpha$   $\mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha$   $\mathfrak{B}$

begin

lemma *dg-prod-2-Dom-usv*:  $usv ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Dom}))$

*<proof>*

lemma *dg-prod-2-Dom-vdomain* [*dg-cs-simps*]:

$\mathcal{D}_{\circ} ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Dom})) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$

*<proof>*

lemma *dg-prod-2-Dom-app* [*dg-prod-cs-simps*]:

assumes  $[g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$

shows  $(\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Dom})(\downarrow g, f)_{\bullet} = [\mathfrak{A}(\text{Dom})(\downarrow g), \mathfrak{B}(\text{Dom})(\downarrow f)]_{\circ}$

*<proof>*

lemma *dg-prod-2-Dom-vrange*:  $\mathcal{R}_{\circ} ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Dom})) \sqsubseteq_{\circ} (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$

*<proof>*

end

**Codomain****context**fixes  $\alpha \mathfrak{A} \mathfrak{B}$ assumes  $\mathfrak{A}$ : *digraph*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{B}$ **begin**

**lemma** *dg-prod-2-Cod-vs-v*:  $vsv ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Cod}))$   
*<proof>*

**lemma** *dg-prod-2-Cod-vdomain*[*dg-cs-simps*]:  
 $\mathcal{D}_\circ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Cod})) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$   
*<proof>*

**lemma** *dg-prod-2-Cod-app*[*dg-prod-cs-simps*]:  
 assumes  $[g, f]_\circ \in_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$   
 shows  $(\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Cod})(g, f)_\bullet = [\mathfrak{A}(\text{Cod})(g), \mathfrak{B}(\text{Cod})(f)]_\circ$ .  
*<proof>*

**lemma** *dg-prod-2-Cod-vrange*:  $\mathcal{R}_\circ ((\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Cod})) \subseteq_\circ (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$   
*<proof>*

**end****Opposite product digraph****context**fixes  $\alpha \mathfrak{A} \mathfrak{B}$ assumes  $\mathfrak{A}$ : *digraph*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{B}$ **begin****interpretation**  $\mathfrak{A}$ : *digraph*  $\alpha \mathfrak{A}$  *<proof>***interpretation**  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{B}$  *<proof>*

**lemma** *dg-prod-2-op-dg-dg-Obj*[*dg-op-simps*]:  
 $(op\text{-}dg \mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj}) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$   
*<proof>*

**lemma** *dg-prod-2-dg-op-dg-Obj*[*dg-op-simps*]:  
 $(\mathfrak{A} \times_{DG} op\text{-}dg \mathfrak{B})(\text{Obj}) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Obj})$   
*<proof>*

**lemma** *dg-prod-2-op-dg-dg-Arr*[*dg-op-simps*]:  
 $(op\text{-}dg \mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr}) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$   
*<proof>*

**lemma** *dg-prod-2-dg-op-dg-Arr*[*dg-op-simps*]:  
 $(\mathfrak{A} \times_{DG} op\text{-}dg \mathfrak{B})(\text{Arr}) = (\mathfrak{A} \times_{DG} \mathfrak{B})(\text{Arr})$   
*<proof>*

**end****context**fixes  $\alpha \mathfrak{A} \mathfrak{B}$ assumes  $\mathfrak{A}$ : *digraph*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{B}$ **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**  $dghm\text{-proj}\text{-fst} :: V \Rightarrow V \Rightarrow V \langle \pi_{DG.1} \rangle$

where  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} = dghm\text{-proj} (2_N) (if2 \mathfrak{A} \mathfrak{B}) 0$

**definition**  $dghm\text{-proj}\text{-snd} :: V \Rightarrow V \Rightarrow V \langle \pi_{DG.2} \rangle$

where  $\pi_{DG.2} \mathfrak{A} \mathfrak{B} = dghm\text{-proj} (2_N) (if2 \mathfrak{A} \mathfrak{B}) (1_N)$

#### 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**  $dghm\text{-proj}\text{-fst}\text{-ObjMap}\text{-app}[dg\text{-cs}\text{-simps}]$ :

assumes  $[a, b]_o \in_o (\mathfrak{A} \times_{DG} \mathfrak{B}) (Obj)$

shows  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} (ObjMap) ([a, b]) \bullet = a$

*<proof>*

**lemma**  $dghm\text{-proj}\text{-snd}\text{-ObjMap}\text{-app}[dg\text{-cs}\text{-simps}]$ :

assumes  $[a, b]_o \in_o (\mathfrak{A} \times_{DG} \mathfrak{B}) (Obj)$

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**  $dghm\text{-proj}\text{-fst}\text{-ArrMap}\text{-app}[dg\text{-cs}\text{-simps}]$ :

assumes  $[g, f]_o \in_o (\mathfrak{A} \times_{DG} \mathfrak{B}) (Arr)$

shows  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} (ArrMap) ([g, f]) \bullet = g$

*<proof>*

**lemma**  $dghm\text{-proj}\text{-snd}\text{-ArrMap}\text{-app}[dg\text{-cs}\text{-simps}]$ :

assumes  $[g, f]_o \in_o (\mathfrak{A} \times_{DG} \mathfrak{B}) (Arr)$

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**  $dghm\text{-proj}\text{-fst}\text{-HomDom}$ :  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} (HomDom) = \mathfrak{A} \times_{DG} \mathfrak{B}$

*<proof>*

**lemma**  $dghm\text{-proj}\text{-fst}\text{-HomCod}$ :  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} (HomCod) = \mathfrak{A}$

*<proof>*

**lemma**  $dghm\text{-proj}\text{-snd}\text{-HomDom}$ :  $\pi_{DG.2} \mathfrak{A} \mathfrak{B} (HomDom) = \mathfrak{A} \times_{DG} \mathfrak{B}$

*<proof>*

**lemma** *dghm-proj-snd-HomCod*:  $\pi_{DG.2} \mathfrak{A} \mathfrak{B} (\text{HomCod}) = \mathfrak{B}$

*<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*  $\alpha \langle 2_{\mathbb{N}} \rangle \langle \text{if2 } \mathfrak{A} \mathfrak{B} \rangle$

*<proof>*

**lemma** *dghm-proj-fst-is-dghm*:

**assumes**  $i \in_{\circ} I$

**shows**  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} : \mathfrak{A} \times_{DG} \mathfrak{B} \mapsto_{DG} \alpha \mathfrak{A}$

*<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**  $\pi_{DG.1} \mathfrak{A} \mathfrak{B} : \mathfrak{C} \mapsto_{DG} \alpha \mathfrak{D}$

*<proof>*

**lemma** *dghm-proj-snd-is-dghm*:

**assumes**  $i \in_{\circ} I$

**shows**  $\pi_{DG.2} \mathfrak{A} \mathfrak{B} : \mathfrak{A} \times_{DG} \mathfrak{B} \mapsto_{DG} \alpha \mathfrak{B}$

*<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**  $\pi_{DG.2} \mathfrak{A} \mathfrak{B} : \mathfrak{C} \mapsto_{DG} \alpha \mathfrak{D}$

*<proof>*

**end**

### 3.8.13 Product of three digraphs

**definition** *dg-prod-3* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \langle \langle (- \times_{DG3} - \times_{DG3} -) \rangle [81, 81, 81] 80 \rangle$

**where**  $\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C} = (\prod_{DG} i \in_{\circ} 3_{\mathbb{N}}. \text{if3 } \mathfrak{A} \mathfrak{B} \mathfrak{C} i)$

### 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**  $\mathcal{Z} \alpha \langle \text{proof} \rangle$

**interpretation**  $\mathfrak{A}$ : *digraph*  $\alpha \mathfrak{A} \langle \text{proof} \rangle$

**interpretation**  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{B} \langle \text{proof} \rangle$

**interpretation**  $\mathfrak{B}$ : *digraph*  $\alpha \mathfrak{C} \langle \text{proof} \rangle$

**lemma** *finite-pdigraph-dg-prod-3*:

*finite-pdigraph*  $\alpha (3_{\mathbb{N}}) (\text{if3 } \mathfrak{A} \mathfrak{B} \mathfrak{C})$

*<proof>*

**interpretation** *finite-pdigraph*  $\alpha \langle 3_{\mathbb{N}} \rangle \langle \text{if3 } \mathfrak{A} \mathfrak{B} \mathfrak{C} \rangle$

*<proof>*

**lemma** *digraph-dg-prod-3[ dg-cs-intros ]*: *digraph*  $\alpha$  ( $\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C}$ )

*<proof>*

**end**

**Object**

**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**

**lemma** *dg-prod-3-ObjI*:

**assumes**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$  **and**  $b \in_{\circ} \mathfrak{B}(\text{Obj})$  **and**  $c \in_{\circ} \mathfrak{C}(\text{Obj})$

**shows**  $[a, b, c]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Obj})$

*<proof>*

**lemma** *dg-prod-3-ObjI'*[*dg-prod-cs-intros*]:

**assumes**  $abc = [a, b, c]_{\circ}$  **and**  $a \in_{\circ} \mathfrak{A}(\text{Obj})$  **and**  $b \in_{\circ} \mathfrak{B}(\text{Obj})$  **and**  $c \in_{\circ} \mathfrak{C}(\text{Obj})$

**shows**  $abc \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Obj})$

*<proof>*

**lemma** *dg-prod-3-ObjE*:

**assumes**  $abc \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Obj})$

**obtains**  $a$   $b$   $c$

**where**  $abc = [a, b, c]_{\circ}$

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

**and**  $b \in_{\circ} \mathfrak{B}(\text{Obj})$

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

*<proof>*

**end**

**Arrow**

**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**

**lemma** *dg-prod-3-ArrI*:

**assumes**  $h \in_{\circ} \mathfrak{A}(\text{Arr})$  **and**  $g \in_{\circ} \mathfrak{B}(\text{Arr})$  **and**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$

**shows**  $[h, g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Arr})$

*<proof>*

**lemma** *dg-prod-3-ArrI'*[*dg-prod-cs-intros*]:

**assumes**  $hgf = [h, g, f]_{\circ}$

**and**  $h \in_{\circ} \mathfrak{A}(\text{Arr})$

**and**  $g \in_{\circ} \mathfrak{B}(\text{Arr})$

**and**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$

**shows**  $[h, g, f]_{\circ} \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Arr})$

*<proof>*

**lemma** *dg-prod-3-ArrE*:

**assumes**  $hgf \in_{\circ} (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(\text{Arr})$

**obtains**  $h$   $g$   $f$

where  $hgf = [h, g, f]_o$   
 and  $h \in_o \mathfrak{A}(Arr)$   
 and  $g \in_o \mathfrak{B}(Arr)$   
 and  $f \in_o \mathfrak{C}(Arr)$

*<proof>*

end

### Arrow with a domain and a codomain

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  $\mathfrak{C}$ : *digraph*  $\alpha \mathfrak{C}$  *<proof>*

interpretation *finite-pdigraph*  $\alpha \langle 3_{\mathbb{N}} \rangle \langle if3 \mathfrak{A} \mathfrak{B} \mathfrak{C} \rangle$   
*<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  $[f, f', f'']_o : [a, a', a'']_o \mapsto_{\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C}} [b, b', b'']_o$

*<proof>*

lemma *dg-prod-3-is-arrI'* [*dg-prod-cs-intros*]:

assumes  $F = [f, f', f'']_o$

and  $A = [a, a', a'']_o$

and  $B = [b, b', b'']_o$

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  $F = [f, f', f'']_o$

and  $A = [a, a', a'']_o$

and  $B = [b, b', b'']_o$

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  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{C}$

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**  $\mathfrak{C}$ : *digraph*  $\alpha$   $\mathfrak{C}$  *<proof>*

**lemma** *dg-prod-3-Dom-usv*:  $usv ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Dom))$   
*<proof>*

**lemma** *dg-prod-3-Dom-vdomain*[*dg-cs-simps*]:  
 $\mathcal{D}_o ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Dom)) = (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Arr)$   
*<proof>*

**lemma** *dg-prod-3-Dom-app*[*dg-prod-cs-simps*]:  
**assumes**  $[f, f', f'']_o \in_o (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Arr)$   
**shows**  $(\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Dom)(f, f', f'')_\bullet =$   
 $[\mathfrak{A}(Dom)(f), \mathfrak{B}(Dom)(f'), \mathfrak{C}(Dom)(f'')]_o$   
*<proof>*

**lemma** *dg-prod-3-Dom-vrange*:  
 $\mathcal{R}_o ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Dom)) \subseteq_o (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Obj)$   
*<proof>*

**end**

## Codomain

**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**  $\mathfrak{C}$ : *digraph*  $\alpha$   $\mathfrak{C}$  *<proof>*

**lemma** *dg-prod-3-Cod-usv*:  $usv ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Cod))$   
*<proof>*

**lemma** *dg-prod-3-Cod-vdomain*[*dg-cs-simps*]:  
 $\mathcal{D}_o ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Cod)) = (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Arr)$   
*<proof>*

**lemma** *dg-prod-3-Cod-app*[*dg-prod-cs-simps*]:  
**assumes**  $[f, f', f'']_o \in_o (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Arr)$   
**shows**  
 $(\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Cod)(f, f', f'')_\bullet =$   
 $[\mathfrak{A}(Cod)(f), \mathfrak{B}(Cod)(f'), \mathfrak{C}(Cod)(f'')]_o$   
*<proof>*

**lemma** *dg-prod-3-Cod-vrange*:  
 $\mathcal{R}_o ((\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Cod)) \subseteq_o (\mathfrak{A} \times_{DG3} \mathfrak{B} \times_{DG3} \mathfrak{C})(Obj)$   
*<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*  $\alpha$   $\mathfrak{B}$  + *dg: digraph*  $\alpha$   $\mathfrak{C}$  for  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$  +  
**assumes** *subdg-Obj-vsubset*[*dg-sub-fw-cs-intros*]:  
 $a \in_{\circ} \mathfrak{B}(\text{Obj}) \implies a \in_{\circ} \mathfrak{C}(\text{Obj})$   
**and** *subdg-is-arr-vsubset*[*dg-sub-fw-cs-intros*]:  
 $f : a \mapsto_{\mathfrak{B}} b \implies f : a \mapsto_{\mathfrak{C}} b$

**abbreviation** *is-subdigraph* ( $\langle - / \subseteq_{DG} - \rangle$ ) [51, 51] 50  
**where**  $\mathfrak{B} \subseteq_{DG} \alpha \mathfrak{C} \equiv \text{subdigraph } \alpha \mathfrak{B} \mathfrak{C}$

**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**  $\mathfrak{B}' \subseteq_{DG} \alpha' \mathfrak{C}$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *subdigraph*) *subdigraph-axioms''*[*dg-cs-intros*]:  
**assumes**  $\alpha' = \alpha$  **and**  $\mathfrak{C}' = \mathfrak{C}$   
**shows**  $\mathfrak{B} \subseteq_{DG} \alpha' \mathfrak{C}'$   
 $\langle \text{proof} \rangle$

**mk-ide rf** *subdigraph-def*[*unfolded subdigraph-axioms-def*]  
 $| \text{intro } \text{subdigraph} I |$   
 $| \text{dest } \text{subdigraph} D [ \text{dest} ] |$   
 $| \text{elim } \text{subdigraph} E [ \text{elim}! ] |$

**lemmas** [*dg-sub-cs-intros*] = *subdigraphD*(1,2)

The opposite subdigraph.

**lemma** (**in** *subdigraph*) *subdg-subdigraph-op-dg-op-dg: op-dg*  $\mathfrak{B} \subseteq_{DG} \alpha \text{ op-dg } \mathfrak{C}$   
 $\langle \text{proof} \rangle$

**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**  $a \in_{\circ} \mathfrak{B}(\text{Obj})$   
**shows**  $a \in_{\circ} \mathfrak{C}(\text{Obj})$   
 $\langle \text{proof} \rangle$

**lemmas**  $[dg\text{-sub-fw-cs-intros}] = \text{subdigraph.subdg-objD}$

**lemma** (**in** *subdigraph*) *subdg-arrD* $[dg\text{-sub-fw-cs-intros}]$ :  
**assumes**  $f \in_{\circ} \mathfrak{B}(\text{Arr})$   
**shows**  $f \in_{\circ} \mathfrak{C}(\text{Arr})$   
 $\langle \text{proof} \rangle$

**lemmas**  $[dg\text{-sub-fw-cs-intros}] = \text{subdigraph.subdg-arrD}$

**lemma** (**in** *subdigraph*) *subdg-dom-simp*:  
**assumes**  $f \in_{\circ} \mathfrak{B}(\text{Arr})$   
**shows**  $\mathfrak{B}(\text{Dom})(f) = \mathfrak{C}(\text{Dom})(f)$   
 $\langle \text{proof} \rangle$

**lemmas**  $[dg\text{-sub-fw-cs-intros}] = \text{subdigraph.subdg-dom-simp}$

**lemma** (**in** *subdigraph*) *subdg-cod-simp*:  
**assumes**  $f \in_{\circ} \mathfrak{B}(\text{Arr})$   
**shows**  $\mathfrak{B}(\text{Cod})(f) = \mathfrak{C}(\text{Cod})(f)$   
 $\langle \text{proof} \rangle$

**lemmas**  $[dg\text{-sub-fw-cs-intros}] = \text{subdigraph.subdg-cod-simp}$

**lemma** (**in** *subdigraph*) *subdg-is-arrD*:  
**assumes**  $f : a \mapsto_{\mathfrak{B}} b$   
**shows**  $f : a \mapsto_{\mathfrak{C}} b$   
 $\langle \text{proof} \rangle$

**lemmas**  $[dg\text{-sub-fw-cs-intros}] = \text{subdigraph.subdg-is-arrD}$

### The subdigraph relation is a partial order

**lemma** *subdg-refl*:  
**assumes**  $\text{digraph } \alpha \mathfrak{A}$   
**shows**  $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{A}$   
 $\langle \text{proof} \rangle$

**lemma** *subdg-trans* $[trans]$ :  
**assumes**  $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{B}$  and  $\mathfrak{B} \subseteq_{DG\alpha} \mathfrak{C}$   
**shows**  $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{C}$   
 $\langle \text{proof} \rangle$

**lemma** *subdg-antisym*:  
**assumes**  $\mathfrak{A} \subseteq_{DG\alpha} \mathfrak{B}$  and  $\mathfrak{B} \subseteq_{DG\alpha} \mathfrak{A}$   
**shows**  $\mathfrak{A} = \mathfrak{B}$   
 $\langle \text{proof} \rangle$

### 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*  $\mathfrak{B} \mathfrak{C} = [\text{vid-on } (\mathfrak{B}(\text{Obj})), \text{vid-on } (\mathfrak{B}(\text{Arr})), \mathfrak{B}, \mathfrak{C}]_{\circ}$

Components.

**lemma** *dghm-inc-components*:

**shows**  $dghm-inc \mathfrak{B} \mathfrak{C}(\text{ObjMap}) = vid-on (\mathfrak{B}(\text{Obj}))$   
**and**  $dghm-inc \mathfrak{B} \mathfrak{C}(\text{ArrMap}) = vid-on (\mathfrak{B}(\text{Arr}))$   
**and**  $[dg-cs-simps]: dghm-inc \mathfrak{B} \mathfrak{C}(\text{HomDom}) = \mathfrak{B}$   
**and**  $[dg-cs-simps]: dghm-inc \mathfrak{B} \mathfrak{C}(\text{HomCod}) = \mathfrak{C}$   
*<proof>*

**Object map**

**mk-VLambda** *dghm-inc-components(1)[folded VLambda-vid-on]*  
 $|vsu \text{ dghm-inc-ObjMap-vs}[dg-cs-intros]|$   
 $|vdomain \text{ dghm-inc-ObjMap-vdomain}[dg-cs-simps]|$   
 $|app \text{ dghm-inc-ObjMap-app}[dg-cs-simps]|$

**Arrow map**

**mk-VLambda** *dghm-inc-components(2)[folded VLambda-vid-on]*  
 $|vsu \text{ dghm-inc-ArrMap-vs}[dg-cs-intros]|$   
 $|vdomain \text{ dghm-inc-ArrMap-vdomain}[dg-cs-simps]|$   
 $|app \text{ dghm-inc-ArrMap-app}[dg-cs-simps]|$

**Canonical inclusion digraph homomorphism associated with a subdigraph**

**sublocale**  $subdigraph \subseteq inc: is-ft-dghm \alpha \mathfrak{B} \mathfrak{C} \langle dghm-inc \mathfrak{B} \mathfrak{C} \rangle$   
*<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*)  $subdg-dghm-inc-op-dg-is-dghm[dg-sub-cs-intros]:$   
 $dghm-inc (op-dg \mathfrak{B}) (op-dg \mathfrak{C}) : op-dg \mathfrak{B} \mapsto_{DG.f_{faithful\alpha}} op-dg \mathfrak{C}$   
*<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**  $fl-subdg-is-fl-dghm-inc: dghm-inc \mathfrak{B} \mathfrak{C} : \mathfrak{B} \mapsto_{DG.f_{full\alpha}} \mathfrak{C}$

**abbreviation**  $is-fl-subdigraph (\langle -/ \subseteq_{DG.f_{full\alpha}} - \rangle [51, 51] 50)$   
**where**  $\mathfrak{B} \subseteq_{DG.f_{full\alpha}} \mathfrak{C} \equiv fl-subdigraph \alpha \mathfrak{B} \mathfrak{C}$

**sublocale**  $fl-subdigraph \subseteq inc: is-fl-dghm \alpha \mathfrak{B} \mathfrak{C} \langle dghm-inc \mathfrak{B} \mathfrak{C} \rangle$   
*<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**  $A \in_{\circ} \mathfrak{B}(\text{Obj})$  **and**  $B \in_{\circ} \mathfrak{B}(\text{Obj})$   
**shows**  $\text{Hom } \mathfrak{B} A B = \text{Hom } \mathfrak{C} A B$   
 ⟨proof⟩

### 3.9.5 Wide subdigraph

#### Definition and elementary properties

See [3]<sup>3</sup>.

**locale** *wide-subdigraph* = *subdigraph* +  
**assumes** *wide-subdg-Obj*[*dg-sub-bw-cs-intros*]:  $a \in_{\circ} \mathfrak{C}(\text{Obj}) \implies a \in_{\circ} \mathfrak{B}(\text{Obj})$

**abbreviation** *is-wide-subdigraph* ( $\langle \langle - / \subseteq_{DG.wide1} - \rangle \rangle$ ) [51, 51] 50  
**where**  $\mathfrak{B} \subseteq_{DG.wide\alpha} \mathfrak{C} \equiv \text{wide-subdigraph } \alpha \mathfrak{B} \mathfrak{C}$

**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*]:

<sup>3</sup><https://ncatlab.org/nlab/show/wide+subcategory>

$\mathfrak{B}(\text{Obj}) = \mathfrak{C}(\text{Obj})$   
 ⟨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  $\alpha$   $\mathfrak{A}$   
 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**  $dg-0 = [0, 0, 0, 0]$ .

Components.

**lemma**  $dg-0$ -components:  
**shows**  $dg-0(\text{Obj}) = 0$   
**and**  $dg-0(\text{Arr}) = 0$   
**and**  $dg-0(\text{Dom}) = 0$   
**and**  $dg-0(\text{Cod}) = 0$   
 $\langle proof \rangle$

#### 0 is a digraph

**lemma** (in  $\mathcal{Z}$ )  $digraph$ - $dg-0$ [ $dg$ -cs-intros]:  $digraph \alpha \ dg-0$   
 $\langle proof \rangle$

**lemmas** [ $dg$ -cs-intros] =  $\mathcal{Z}.digraph$ - $dg-0$

#### Opposite of the digraph 0

**lemma**  $op$ - $dg$ - $dg-0$ [ $dg$ -op-simps]:  $op$ - $dg \ (dg-0) = dg-0$   
 $\langle proof \rangle$

#### Arrow with a domain and a codomain

**lemma**  $dg-0$ -is-arr-iff[ $simp$ ]:  $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-0} \mathfrak{B} \longleftrightarrow False$   
 $\langle proof \rangle$

#### A digraph without objects is empty

**lemma** (in  $digraph$ )  $dg$ - $dg-0$ -if-Obj-0:  
**assumes**  $\mathfrak{C}(\text{Obj}) = 0$   
**shows**  $\mathfrak{C} = dg-0$   
 $\langle proof \rangle$

### 3.10.3 Empty digraph homomorphism

#### Definition and elementary properties

**definition**  $dghm-0 :: V \Rightarrow V$   
**where**  $dghm-0 \ \mathfrak{A} = [0, 0, dg-0, \mathfrak{A}]$ .

Components.

**lemma**  $dghm-0$ -components:  
**shows**  $dghm-0 \ \mathfrak{A}(\text{ObjMap}) = 0$

**and**  $dghm-0 \mathfrak{A}(\text{ArrMap}) = 0$   
**and**  $dghm-0 \mathfrak{A}(\text{HomDom}) = dg-0$   
**and**  $dghm-0 \mathfrak{A}(\text{HomCod}) = \mathfrak{A}$   
 ⟨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**  $dghm-0-ObjMap-vsuv[dg-cs-intros]$ :  $vsuv (dghm-0 \mathfrak{C}(\text{ObjMap}))$   
 ⟨proof⟩

### Arrow map

**lemma**  $dghm-0-ArrMap-vsuv[dg-cs-intros]$ :  $vsuv (dghm-0 \mathfrak{C}(\text{ArrMap}))$   
 ⟨proof⟩

### Empty digraph homomorphism is a faithful digraph homomorphism

**lemma** (in  $\mathcal{Z}$ )  $dghm-0-is-ft-dghm$ :  
**assumes**  $digraph \alpha \mathfrak{A}$   
**shows**  $dghm-0 \mathfrak{A} : dg-0 \mapsto \mapsto_{DG.f\text{faithful}\alpha} \mathfrak{A}$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ )  $dghm-0-is-ft-dghm'[dghm-cs-intros]$ :  
**assumes**  $digraph \alpha \mathfrak{A}$   
**and**  $\mathfrak{B}' = \mathfrak{A}$   
**and**  $\mathfrak{A}' = dg-0$   
**shows**  $dghm-0 \mathfrak{A} : \mathfrak{A}' \mapsto \mapsto_{DG.f\text{faithful}\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $[dghm-cs-intros] = \mathcal{Z}.dghm-0-is-ft-dghm'$

**lemma** (in  $\mathcal{Z}$ )  $dghm-0-is-dghm$ :  
**assumes**  $digraph \alpha \mathfrak{A}$   
**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**  $digraph \alpha \mathfrak{A}$   
**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  $tdghm-0 \mathfrak{C} = [0, dghm-0 \mathfrak{C}, dghm-0 \mathfrak{C}, dg-0, \mathfrak{C}]_o$ .

Components.

**lemma**  $tdghm-0$ -components:

shows  $tdghm-0 \mathfrak{C}(NTMap) = 0$

and  $[dg\text{-}cs\text{-}simps]: tdghm-0 \mathfrak{C}(NTDom) = dghm-0 \mathfrak{C}$

and  $[dg\text{-}cs\text{-}simps]: tdghm-0 \mathfrak{C}(NTCod) = dghm-0 \mathfrak{C}$

and  $[dg\text{-}cs\text{-}simps]: tdghm-0 \mathfrak{C}(NTDGDom) = dg-0$

and  $[dg\text{-}cs\text{-}simps]: tdghm-0 \mathfrak{C}(NTDGCod) = \mathfrak{C}$

$\langle proof \rangle$

Duality.

**lemma**  $op\text{-}tdghm\text{-}tdghm-0$ :  $op\text{-}tdghm (tdghm-0 \mathfrak{C}) = tdghm-0 (op\text{-}dg \mathfrak{C})$

$\langle proof \rangle$

#### Transformation map

**lemma**  $tdghm-0\text{-}NTMap\text{-}vsv[dg\text{-}cs\text{-}intros]$ :  $vsv (tdghm-0 \mathfrak{C}(NTMap))$

$\langle proof \rangle$

**lemma**  $tdghm-0\text{-}NTMap\text{-}vdomain[dg\text{-}cs\text{-}simps]$ :  $\mathcal{D}_o (tdghm-0 \mathfrak{C}(NTMap)) = 0$

$\langle proof \rangle$

**lemma**  $tdghm-0\text{-}NTMap\text{-}vrangle[dg\text{-}cs\text{-}simps]$ :  $\mathcal{R}_o (tdghm-0 \mathfrak{C}(NTMap)) = 0$

$\langle proof \rangle$

#### Empty transformation of digraph homomorphisms is a transformation of digraph homomorphisms

**lemma** (in *digraph*)  $dg\text{-}tdghm-0\text{-}is\text{-}tdghmI$ :

$tdghm-0 \mathfrak{C} : dghm-0 \mathfrak{C} \mapsto_{DGHM} dghm-0 \mathfrak{C} : dg-0 \mapsto_{DG\alpha} \mathfrak{C}$

$\langle proof \rangle$

**lemma** (in *digraph*)  $dg\text{-}tdghm-0\text{-}is\text{-}tdghmI'$   $[dg\text{-}cs\text{-}intros]$ :

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  $tdghm-0 \mathfrak{C} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG\alpha} \mathfrak{B}'$

$\langle proof \rangle$

**lemmas**  $[dg\text{-}cs\text{-}intros] = digraph.dg\text{-}tdghm-0\text{-}is\text{-}tdghmI'$

**lemma**  $is\text{-}tdghm\text{-}is\text{-}tdghm-0\text{-}if\text{-}dg-0$ :

assumes  $\mathfrak{N} : \mathfrak{F} \mapsto_{DGHM} \mathfrak{G} : dg-0 \mapsto_{DG\alpha} \mathfrak{C}$

shows  $\mathfrak{N} = tdghm-0 \mathfrak{C}$  and  $\mathfrak{F} = dghm-0 \mathfrak{C}$  and  $\mathfrak{G} = dghm-0 \mathfrak{C}$

$\langle proof \rangle$

### 3.10.5 10: digraph with one object and no arrows

#### Definition and elementary properties

**definition**  $dg-10 :: V \Rightarrow V$

where  $dg-10 \mathbf{a} = [set \{a\}, 0, 0, 0]$ .

Components.

**lemma** *dg-10-components*:

**shows**  $dg-10 \mathbf{a}(Obj) = set \{a\}$

**and**  $dg-10 \mathbf{a}(Arr) = 0$

**and**  $dg-10 \mathbf{a}(Dom) = 0$

**and**  $dg-10 \mathbf{a}(Cod) = 0$

*<proof>*

**10 is a digraph**

**lemma** (in  $\mathcal{Z}$ ) *digraph-dg-10*:

**assumes**  $a \in_o Vset \alpha$

**shows** *digraph*  $\alpha$  ( $dg-10 \mathbf{a}$ )

*<proof>*

**Arrow with a domain and a codomain**

**lemma** *dg-10-is-arr-iff*:  $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-10 \mathbf{a}} \mathfrak{B} \longleftrightarrow False$

*<proof>*

### 3.10.6 1: digraph with one object and one arrow

**Definition and elementary properties**

**definition**  $dg-1 :: V \Rightarrow V \Rightarrow V$

**where**  $dg-1 \mathbf{a} \mathbf{f} = [set \{a\}, set \{f\}, set \{(f, a)\}, set \{(f, a)\}]$ .

Components.

**lemma** *dg-1-components*:

**shows**  $dg-1 \mathbf{a} \mathbf{f}(Obj) = set \{a\}$

**and**  $dg-1 \mathbf{a} \mathbf{f}(Arr) = set \{f\}$

**and**  $dg-1 \mathbf{a} \mathbf{f}(Dom) = set \{(f, a)\}$

**and**  $dg-1 \mathbf{a} \mathbf{f}(Cod) = set \{(f, a)\}$

*<proof>*

**1 is a digraph**

**lemma** (in  $\mathcal{Z}$ ) *digraph-dg-1*:

**assumes**  $a \in_o Vset \alpha$  **and**  $f \in_o Vset \alpha$

**shows** *digraph*  $\alpha$  ( $dg-1 \mathbf{a} \mathbf{f}$ )

*<proof>*

**Arrow with a domain and a codomain**

**lemma** *dg-1-is-arrI*:

**assumes**  $a = a$  **and**  $b = a$  **and**  $f = f$

**shows**  $f : a \mapsto_{dg-1 \mathbf{a} \mathbf{f}} b$

*<proof>*

**lemma** *dg-1-is-arrD*:

**assumes**  $f : a \mapsto_{dg-1 \mathbf{a} \mathbf{f}} b$

**shows**  $a = a$  **and**  $b = a$  **and**  $f = f$

*<proof>*

**lemma** *dg-1-is-arrE*:

**assumes**  $f : a \mapsto_{dg-1 \mathbf{a} \mathbf{f}} b$

**obtains**  $a = \mathbf{a}$  and  $b = \mathbf{a}$  and  $f = \mathbf{f}$   
(proof)

**lemma**  $dg-1$ -is-arr-iff:  $f : a \mapsto_{dg-1} \mathbf{a} \mathbf{f} b \leftrightarrow (a = \mathbf{a} \wedge b = \mathbf{a} \wedge f = \mathbf{f})$   
(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,  $\alpha$ -*GRPH* is defined as a digraph of digraphs and digraph homomorphisms in  $V_\alpha$ .

**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*  $\alpha =$

[  
 set { $\mathfrak{C}$ . *digraph*  $\alpha$   $\mathfrak{C}$ },  
 all-dghms  $\alpha$ ,  
 ( $\lambda \mathfrak{F} \in_\circ \text{all-dghms } \alpha. \mathfrak{F}(\text{HomDom})$ ),  
 ( $\lambda \mathfrak{F} \in_\circ \text{all-dghms } \alpha. \mathfrak{F}(\text{HomCod})$ )  
 ]<sub>o</sub>

Components.

**lemma** *dg-GRPH-components*:

**shows** *dg-GRPH*  $\alpha(\text{Obj}) = \text{set } \{\mathfrak{C}. \text{digraph } \alpha \mathfrak{C}\}$

**and** *dg-GRPH*  $\alpha(\text{Arr}) = \text{all-dghms } \alpha$

**and** *dg-GRPH*  $\alpha(\text{Dom}) = (\lambda \mathfrak{F} \in_\circ \text{all-dghms } \alpha. \mathfrak{F}(\text{HomDom}))$

**and** *dg-GRPH*  $\alpha(\text{Cod}) = (\lambda \mathfrak{F} \in_\circ \text{all-dghms } \alpha. \mathfrak{F}(\text{HomCod}))$

*<proof>*

#### 3.11.3 Object

**lemma** *dg-GRPH-ObjI*:

**assumes** *digraph*  $\alpha \mathfrak{A}$

**shows**  $\mathfrak{A} \in_\circ \text{dg-GRPH } \alpha(\text{Obj})$

*<proof>*

**lemma** *dg-GRPH-ObjD*:

**assumes**  $\mathfrak{A} \in_\circ \text{dg-GRPH } \alpha(\text{Obj})$

**shows** *digraph*  $\alpha \mathfrak{A}$

*<proof>*

**lemma** *dg-GRPH-ObjE*:

**assumes**  $\mathfrak{A} \in_\circ \text{dg-GRPH } \alpha(\text{Obj})$

**obtains** *digraph*  $\alpha \mathfrak{A}$

*<proof>*

**lemma** *dg-GRPH-Obj-iff[GRPH-cs-simps]*:

$\mathfrak{A} \in_\circ \text{dg-GRPH } \alpha(\text{Obj}) \longleftrightarrow \text{digraph } \alpha \mathfrak{A}$

*<proof>*

#### 3.11.4 Domain

**mk-VLambda** *dg-GRPH-components*(3)

|*vsu dg-GRPH-Dom-vsuv[GRPH-cs-intros]*|

|*vdomain dg-GRPH-Dom-vdomain[GRPH-cs-simps]*|

|*app dg-GRPH-Dom-app[GRPH-cs-simps]*|

**lemma** *dg-GRPH-Dom-vrange*:  $\mathcal{R}_\circ (dg-GRPH \ \alpha (Dom)) \sqsubseteq_\circ dg-GRPH \ \alpha (Obj)$   
 ⟨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*:  $\mathcal{R}_\circ (dg-GRPH \ \alpha (Cod)) \sqsubseteq_\circ dg-GRPH \ \alpha (Obj)$   
 ⟨proof⟩

### 3.11.6 GRPH is a digraph

**lemma** (in  $\mathcal{Z}$ ) *tiny-digraph-dg-GRPH*:  
 assumes  $\mathcal{Z} \ \beta$  and  $\alpha \in_\circ \beta$   
 shows *tiny-digraph*  $\beta$  (*dg-GRPH*  $\alpha$ )  
 ⟨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  $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-GRPH \ \alpha} \mathfrak{B}$   
 ⟨proof⟩

**lemma** *dg-GRPH-is-arrD*:  
 assumes  $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-GRPH \ \alpha} \mathfrak{B}$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG} \alpha \ \mathfrak{B}$   
 ⟨proof⟩

**lemma** *dg-GRPH-is-arrE*:  
 assumes  $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-GRPH \ \alpha} \mathfrak{B}$   
 obtains  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG} \alpha \ \mathfrak{B}$   
 ⟨proof⟩

**lemma** *dg-GRPH-is-arr-iff*[*GRPH-cs-simps*]:  
 $\mathfrak{F} : \mathfrak{A} \mapsto_{dg-GRPH \ \alpha} \mathfrak{B} \longleftrightarrow \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{DG} \alpha \ \mathfrak{B}$   
 ⟨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,  $\alpha$ -*Rel* is defined as a digraph of sets and binary relations in  $V_\alpha$ .

**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**  $f(\backslash ArrVal)(\backslash a) = g(\backslash ArrVal)(\backslash a)$

*<proof>*

### 3.12.3 Arrow for *Rel*

#### Definition and elementary properties

**locale** *arr-Rel* =  $\mathcal{Z}$   $\alpha$  + *vfsequence*  $T$  + *ArrVal*: *vbrelation*  $\langle T(\backslash ArrVal) \rangle$  **for**  $\alpha$   $T$  +

**assumes** *arr-Rel-length*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:

*vcard*  $T = 3_{\mathbb{N}}$

**and** *arr-Rel-ArrVal-vdomain*:  $\mathcal{D}_\circ(T(\backslash ArrVal)) \subseteq_\circ T(\backslash ArrDom)$

**and** *arr-Rel-ArrVal-vrange*:  $\mathcal{R}_\circ(T(\backslash ArrVal)) \subseteq_\circ T(\backslash ArrCod)$

**and** *arr-Rel-ArrDom-in-Vset*:  $T(\backslash ArrDom) \in_\circ Vset\ \alpha$

**and** *arr-Rel-ArrCod-in-Vset*:  $T(\backslash ArrCod) \in_\circ Vset\ \alpha$

**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**  $[f, A, B]_\circ(\backslash ArrVal) = f$

**and**  $[f, A, B]_\circ(\backslash ArrDom) = A$

**and**  $[f, A, B]_\circ(\backslash ArrCod) = B$

*<proof>*

Rules.

**lemma** (**in** *arr-Rel*) *arr-Rel-axioms'*[*dg-cs-intros*, *dg-Rel-cs-intros*]:

**assumes**  $\alpha' = \alpha$

**shows** *arr-Rel*  $\alpha'$   $T$

*<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**  $arr-Rel\ \alpha\ [r, a, b]_\circ$   
 $\langle proof \rangle$

Elementary properties.

**lemma** *arr-Rel-eqI*:

**assumes**  $arr-Rel\ \alpha\ S$   
**and**  $arr-Rel\ \alpha\ T$   
**and**  $S(\downarrow ArrVal) = T(\downarrow ArrVal)$   
**and**  $S(\downarrow ArrDom) = T(\downarrow ArrDom)$   
**and**  $S(\downarrow ArrCod) = T(\downarrow ArrCod)$   
**shows**  $S = T$   
 $\langle proof \rangle$

**lemma** (in *arr-Rel*) *arr-Rel-def*:  $T = [T(\downarrow ArrVal), T(\downarrow ArrDom), T(\downarrow ArrCod)]_\circ$ .

$\langle proof \rangle$

Size.

**lemma** (in *arr-Rel*) *arr-Rel-ArrVal-in-Vset*:  $T(\downarrow ArrVal) \in_\circ Vset\ \alpha$

$\langle proof \rangle$

**lemma** (in *arr-Rel*) *arr-Rel-in-Vset*:  $T \in_\circ Vset\ \alpha$

$\langle proof \rangle$

**lemma** *small-arr-Rel[simp]*:  $small\ \{T.\ arr-Rel\ \alpha\ T\}$

$\langle proof \rangle$

Other elementary properties.

**lemma** *set-Collect-arr-Rel[simp]*:

$x \in_\circ set\ (Collect\ (arr-Rel\ \alpha)) \leftrightarrow arr-Rel\ \alpha\ x$   
 $\langle proof \rangle$

**lemma** (in *arr-Rel*) *arr-Rel-ArrVal-ubset-ArrDom-ArrCod*:

$T(\downarrow ArrVal) \subseteq_\circ T(\downarrow ArrDom) \times_\circ T(\downarrow ArrCod)$   
 $\langle proof \rangle$

## Composition

See Chapter I-7 in [39].

**definition** *comp-Rel* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{Rel} \rangle$  55)

**where**  $comp-Rel\ S\ T = [S(\downarrow ArrVal) \circ_\circ T(\downarrow ArrVal), T(\downarrow ArrDom), S(\downarrow ArrCod)]_\circ$ .

Components.

**lemma** *comp-Rel-components*:

**shows**  $(S \circ_{Rel} T)(\downarrow ArrVal) = S(\downarrow ArrVal) \circ_\circ T(\downarrow ArrVal)$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]$ :  
 $(S \circ_{Rel} T)(\downarrow ArrDom) = T(\downarrow ArrDom)$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]$ :  
 $(S \circ_{Rel} T)(\downarrow ArrCod) = S(\downarrow ArrCod)$

$\langle proof \rangle$

Elementary properties.

**lemma** *comp-Rel-vsuv*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:

$vsu (S \circ_{Rel} T)$

$\langle proof \rangle$

**lemma** *arr-Rel-comp-Rel*[*dg-Rel-cs-intros*]:

**assumes** *arr-Rel*  $\alpha$  *S* **and** *arr-Rel*  $\alpha$  *T*

**shows** *arr-Rel*  $\alpha$  ( $S \circ_{Rel} T$ )

$\langle proof \rangle$

**lemma** *arr-Rel-comp-Rel-assoc*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:

$(H \circ_{Rel} G) \circ_{Rel} F = H \circ_{Rel} (G \circ_{Rel} F)$

$\langle proof \rangle$

### Inclusion arrow

The definition of the inclusion arrow is based on the concept of the inclusion map, e.g., see [5]<sup>4</sup>

**definition** *incl-Rel*  $A B = [vid-on A, A, B]_o$ .

Components.

**lemma** *incl-Rel-components*:

**shows** *incl-Rel*  $A B(\text{ArrVal}) = vid-on A$

**and** [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *incl-Rel*  $A B(\text{ArrDom}) = A$

**and** [*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]: *incl-Rel*  $A B(\text{ArrCod}) = B$

$\langle proof \rangle$

Arrow value.

**lemma** *incl-Rel-ArrVal-vsuv*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:

$vsu (incl-Rel A B(\text{ArrVal}))$

$\langle proof \rangle$

**lemma** *incl-Rel-ArrVal-vdomain*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:

$\mathcal{D}_o (incl-Rel A B(\text{ArrVal})) = A$

$\langle proof \rangle$

**lemma** *incl-Rel-ArrVal-app*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:

**assumes**  $a \in_o A$

**shows** *incl-Rel*  $A B(\text{ArrVal})(a) = a$

$\langle proof \rangle$

Elementary properties.

**lemma** *incl-Rel-vfsequence*[*dg-Rel-shared-cs-intros*, *dg-Rel-cs-intros*]:

*vfsequence* (*incl-Rel*  $A B$ )

$\langle proof \rangle$

**lemma** *incl-Rel-vcard*[*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*]:

*vcard* (*incl-Rel*  $A B$ ) =  $3_{\mathbb{N}}$

$\langle proof \rangle$

**lemma** (**in**  $\mathcal{Z}$ ) *arr-Rel-incl-RelI*:

**assumes**  $A \in_o Vset \alpha$  **and**  $B \in_o Vset \alpha$  **and**  $A \subseteq_o B$

**shows** *arr-Rel*  $\alpha$  (*incl-Rel*  $A B$ )

$\langle proof \rangle$

---

<sup>4</sup>[https://en.wikipedia.org/wiki/Inclusion\\_map](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**  $id-Rel A(\downarrow Arr Val) = vid-on A$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]: id-Rel A(\downarrow Arr Dom) = A$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]: id-Rel A(\downarrow Arr Cod) = A$   
 $\langle proof \rangle$

Elementary properties.

**lemma**  $id-Rel-vfsequence[dg-Rel-shared-cs-intros, dg-Rel-cs-intros]$ :  
 $vfsequence (id-Rel A)$   
 $\langle proof \rangle$

**lemma**  $id-Rel-vcard[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]$ :  
 $vcard (id-Rel A) = 3_{\mathbb{N}}$   
 $\langle proof \rangle$

**lemma**  $(in Z) arr-Rel-id-RelI$ :  
**assumes**  $A \in_{\circ} Vset \alpha$   
**shows**  $arr-Rel \alpha (id-Rel A)$   
 $\langle proof \rangle$

**lemma**  $id-Rel-ArrVal-app[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]$ :  
**assumes**  $a \in_{\circ} A$   
**shows**  $id-Rel A(\downarrow Arr Val)(\downarrow a) = a$   
 $\langle proof \rangle$

**lemma**  $arr-Rel-comp-Rel-id-Rel-left[dg-Rel-cs-simps]$ :  
**assumes**  $arr-Rel \alpha F$  **and**  $F(\downarrow Arr Cod) = A$   
**shows**  $id-Rel A \circ_{Rel} F = F$   
 $\langle proof \rangle$

**lemma**  $arr-Rel-comp-Rel-id-Rel-right[dg-Rel-cs-simps]$ :  
**assumes**  $arr-Rel \alpha F$  **and**  $F(\downarrow Arr Dom) = A$   
**shows**  $F \circ_{Rel} id-Rel A = F$   
 $\langle proof \rangle$

## 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**  $converse-Rel :: V \Rightarrow V \langle (-^1_{Rel}) \rangle [1000] 999$   
**where**  $converse-Rel T = [(T(\downarrow Arr Val))^{-1}_{\circ}, T(\downarrow Arr Cod), T(\downarrow Arr Dom)]_{\circ}$ .

**lemma**  $converse-Rel-components$ :  
**shows**  $T^{-1}_{Rel}(\downarrow Arr Val) = (T(\downarrow Arr Val))^{-1}_{\circ}$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]: T^{-1}_{Rel}(\downarrow Arr Dom) = T(\downarrow Arr Cod)$   
**and**  $[dg-Rel-shared-cs-simps, dg-Rel-cs-simps]: T^{-1}_{Rel}(\downarrow Arr Cod) = T(\downarrow Arr Dom)$   
 $\langle proof \rangle$

Elementary properties.

**lemma** (in *arr-Rel*) *arr-Rel-converse-Rel*:  $arr-Rel \alpha (T^{-1}_{Rel})$   
 ⟨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**  $arr-Rel \alpha F$  **and**  $arr-Rel \alpha G$   
**shows**  $F^{-1}_{Rel} = G^{-1}_{Rel} \leftrightarrow F = G$   
 ⟨proof⟩

**lemma** *arr-Rel-converse-Rel-comp-Rel*[*dg-Rel-cs-simps*]:  
**assumes**  $arr-Rel \alpha G$  **and**  $arr-Rel \alpha F$   
**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**  $c \in_{\circ} Vset \alpha$   
**shows**  $arr-Rel \alpha ((id-Rel c)^{-1}_{Rel})$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *arr-Rel-converse-Rel-id-Rel-eq-id-Rel*[  
*dg-Rel-shared-cs-simps*, *dg-Rel-cs-simps*  
 ]:  
**assumes**  $c \in_{\circ} Vset \alpha$   
**shows**  $(id-Rel c)^{-1}_{Rel} = id-Rel c$   
 ⟨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-v11*[*dg-Rel-cs-simps*]:  
**assumes**  $arr-Rel \alpha T$   
**and**  $\mathcal{D}_{\circ} (T(\downarrow ArrVal)) = A$   
**and**  $T(\downarrow ArrDom) = A$   
**and**  $v11 (T(\downarrow ArrVal))$   
**and**  $A \in_{\circ} Vset \alpha$   
**shows**  $T^{-1}_{Rel} \circ_{Rel} T = id-Rel A$   
 ⟨proof⟩

**lemma** *arr-Rel-comp-Rel-converse-Rel-right-if-v11*[*dg-Rel-cs-simps*]:  
**assumes**  $arr-Rel \alpha T$   
**and**  $\mathcal{R}_{\circ} (T(\downarrow ArrVal)) = A$   
**and**  $T(\downarrow ArrCod) = A$   
**and**  $v11 (T(\downarrow ArrVal))$   
**and**  $A \in_{\circ} Vset \alpha$   
**shows**  $T \circ_{Rel} T^{-1}_{Rel} = id-Rel A$   
 ⟨proof⟩

### 3.12.4 *Rel* as a digraph

#### Definition and elementary properties

**definition**  $dg-Rel :: V \Rightarrow V$

where  $dg-Rel \alpha =$

$$\begin{aligned} & [ \\ & \quad Vset \alpha, \\ & \quad set \{T. arr-Rel \alpha T\}, \\ & \quad (\lambda T \epsilon_o set \{T. arr-Rel \alpha T\}. T(\!|ArrDom\!)), \\ & \quad (\lambda T \epsilon_o set \{T. arr-Rel \alpha T\}. T(\!|ArrCod\!)) \\ & ]_o \end{aligned}$$

Components.

**lemma**  $dg-Rel-components$ :

**shows**  $dg-Rel \alpha(\!|Obj\!) = Vset \alpha$

**and**  $dg-Rel \alpha(\!|Arr\!) = set \{T. arr-Rel \alpha T\}$

**and**  $dg-Rel \alpha(\!|Dom\!) = (\lambda T \epsilon_o set \{T. arr-Rel \alpha T\}. T(\!|ArrDom\!))$

**and**  $dg-Rel \alpha(\!|Cod\!) = (\lambda T \epsilon_o set \{T. arr-Rel \alpha T\}. T(\!|ArrCod\!))$

$\langle proof \rangle$

#### Object

**lemma**  $dg-Rel-Obj-iff$ :  $x \epsilon_o dg-Rel \alpha(\!|Obj\!) \longleftrightarrow x \epsilon_o Vset \alpha$

$\langle proof \rangle$

#### Arrow

**lemma**  $dg-Rel-Arr-iff$ [ $dg-Rel-cs-simps$ ]:  $x \epsilon_o dg-Rel \alpha(\!|Arr\!) \longleftrightarrow arr-Rel \alpha x$

$\langle proof \rangle$

#### 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**  $dg-Rel-Dom-vrange$ :  $\mathcal{R}_o (dg-Rel \alpha(\!|Dom\!)) \subseteq_o dg-Rel \alpha(\!|Obj\!)$

$\langle proof \rangle$

#### 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**  $dg-Rel-Cod-vrange$ :  $\mathcal{R}_o (dg-Rel \alpha(\!|Cod\!)) \subseteq_o dg-Rel \alpha(\!|Obj\!)$

$\langle proof \rangle$

#### Arrow with a domain and a codomain

Rules.

**lemma**  $dg-Rel-is-arrI$ [ $dg-Rel-cs-intros$ ]:

**assumes**  $arr-Rel \alpha S$  **and**  $S(\!|ArrDom\!) = A$  **and**  $S(\!|ArrCod\!) = B$

**shows**  $S : A \mapsto dg-Rel \alpha B$

$\langle proof \rangle$

**lemma** *dg-Rel-is-arrD*:

**assumes**  $S : A \mapsto_{dg-Rel} \alpha B$   
**shows**  $arr-Rel \alpha S$   
**and**  $[dg-cs-simps]: S(\downarrow ArrDom) = A$   
**and**  $[dg-cs-simps]: S(\downarrow ArrCod) = B$   
 $\langle proof \rangle$

**lemma** *dg-Rel-is-arrE*:

**assumes**  $S : A \mapsto_{dg-Rel} \alpha B$   
**obtains**  $arr-Rel \alpha S$  **and**  $S(\downarrow ArrDom) = A$  **and**  $S(\downarrow ArrCod) = B$   
 $\langle proof \rangle$

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**  $incl-Rel A B : A \mapsto_{dg-Rel} \alpha B$

$\langle proof \rangle$

**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**  $incl-Rel A B : A' \mapsto_{dg-Rel} \alpha B'$

$\langle proof \rangle$

**lemmas**  $[dg-Rel-cs-intros] = \mathcal{Z}.dg-Rel-incl-Rel-is-arr'$

**lemma** *dg-Rel-is-arr-ArrValE*:

**assumes**  $T : A \mapsto_{dg-Rel} \alpha B$  **and**  $ab \in_{\circ} T(\downarrow ArrVal)$   
**obtains**  $a b$   
**where**  $ab = \langle a, b \rangle$  **and**  $a \in_{\circ} \mathcal{D}_{\circ}(T(\downarrow ArrVal))$  **and**  $b \in_{\circ} \mathcal{R}_{\circ}(T(\downarrow ArrVal))$

$\langle proof \rangle$

**Rel is a digraph**

**lemma** (in  $\mathcal{Z}$ ) *dg-Rel-Hom-vifunior-in-Vset*:

**assumes**  $X \in_{\circ} Vset \alpha$  **and**  $Y \in_{\circ} Vset \alpha$   
**shows**  $(\cup_{\circ} A \in_{\circ} X. \cup_{\circ} B \in_{\circ} Y. Hom(dg-Rel \alpha) A B) \in_{\circ} Vset \alpha$

$\langle proof \rangle$

**lemma** (in  $\mathcal{Z}$ ) *digraph-dg-Rel: digraph  $\alpha$  (dg-Rel  $\alpha$ )*

$\langle proof \rangle$

### 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]<sup>5</sup>. This reference also contains the majority of the results that are presented in this subsection.

**Definition and elementary properties**

**definition** *dg-hm-dag-Rel* ::  $V \Rightarrow V$  ( $\langle \dagger_{DG.Rel} \rangle$ )

<sup>5</sup><https://ncatlab.org/nlab/show/Rel>

**where**  $\dagger_{DG.Rel} \alpha =$   
 $[$   
 $\quad vid-on (dg-Rel \alpha(\!|Obj\!|)),$   
 $\quad VLambda (dg-Rel \alpha(\!|Arr\!|)) \textit{converse-Rel},$   
 $\quad op-dg (dg-Rel \alpha),$   
 $\quad dg-Rel \alpha$   
 $]$ .

Components.

**lemma** *dghm-dag-Rel-components*:

**shows**  $\dagger_{DG.Rel} \alpha(\!|ObjMap\!|) = vid-on (dg-Rel \alpha(\!|Obj\!|))$   
**and**  $\dagger_{DG.Rel} \alpha(\!|ArrMap\!|) = VLambda (dg-Rel \alpha(\!|Arr\!|)) \textit{converse-Rel}$   
**and**  $\dagger_{DG.Rel} \alpha(\!|HomDom\!|) = op-dg (dg-Rel \alpha)$   
**and**  $\dagger_{DG.Rel} \alpha(\!|HomCod\!|) = dg-Rel \alpha$   
*<proof>*

### Object map

**mk-VLambda** *dghm-dag-Rel-components*(1)[*folded VLambda-vid-on*]  
 $|vsv \textit{dghm-dag-Rel-ObjMap-vsuv} [dg-Rel-cs-intros]|$   
 $|vdomain$   
 $\quad \textit{dghm-dag-Rel-ObjMap-vdomain} [unfolded \textit{dg-Rel-components}, \textit{dg-Rel-cs-simps}]$   
 $|$   
 $|app \textit{dghm-dag-Rel-ObjMap-app} [unfolded \textit{dg-Rel-components}, \textit{dg-Rel-cs-simps}]|$

**lemma** *dghm-dag-Rel-ObjMap-vrange*[*dg-cs-simps*]:  $\mathcal{R}_o (\dagger_{DG.Rel} \alpha(\!|ObjMap\!|)) = Vset \alpha$   
*<proof>*

### Arrow map

**mk-VLambda** *dghm-dag-Rel-components*(2)  
 $|vsv \textit{dghm-dag-Rel-ArrMap-vsuv} [dg-Rel-cs-intros]|$   
 $|vdomain \textit{dghm-dag-Rel-ArrMap-vdomain} [dg-Rel-cs-simps]|$   
 $|app \textit{dghm-dag-Rel-ArrMap-app} [unfolded \textit{dg-Rel-cs-simps}, \textit{dg-Rel-cs-simps}]|$

**lemma** *dghm-dag-Rel-ArrMap-app-vdomain*[*dg-cs-simps*]:  
**assumes**  $T : A \mapsto dg-Rel \alpha B$   
**shows**  $\mathcal{D}_o (\dagger_{DG.Rel} \alpha(\!|ArrMap\!|)(\!|T\!|)(\!|ArrVal\!|)) = \mathcal{R}_o (T(\!|ArrVal\!|))$   
*<proof>*

**lemma** *dghm-dag-Rel-ArrMap-app-vrange*[*dg-cs-simps*]:  
**assumes**  $T : A \mapsto dg-Rel \alpha B$   
**shows**  $\mathcal{R}_o (\dagger_{DG.Rel} \alpha(\!|ArrMap\!|)(\!|T\!|)(\!|ArrVal\!|)) = \mathcal{D}_o (T(\!|ArrVal\!|))$   
*<proof>*

**lemma** *dghm-dag-Rel-ArrMap-app-iff*[*dg-cs-simps*]:  
**assumes**  $T : A \mapsto dg-Rel \alpha B$   
**shows**  $\langle a, b \rangle \in_o \dagger_{DG.Rel} \alpha(\!|ArrMap\!|)(\!|T\!|)(\!|ArrVal\!|) \longleftrightarrow \langle b, a \rangle \in_o T(\!|ArrVal\!|)$   
*<proof>*

### Further properties

**lemma** *dghm-dag-Rel-ArrMap-vrange*[*dg-Rel-cs-simps*]:  
 $\mathcal{R}_o (\dagger_{DG.Rel} \alpha(\!|ArrMap\!|)) = dg-Rel \alpha(\!|Arr\!|)$   
*<proof>*

**lemma** *dghm-dag-Rel-ArrMap-app-is-arr*:  
**assumes**  $T : b \mapsto dg-Rel \alpha a$

**shows**

$$\dagger_{DG.Rel} \alpha (ArrMap) (T) : \dagger_{DG.Rel} \alpha (ObjMap) (a) \mapsto_{dg-Rel} \alpha \dagger_{DG.Rel} \alpha (ObjMap) (b)$$

*<proof>*

### Canonical dagger for *Rel* is a digraph isomorphism

**lemma (in  $\mathcal{Z}$ )** *dghm-dag-Rel-is-iso-dghm*:

$$\dagger_{DG.Rel} \alpha : op-dg (dg-Rel \alpha) \mapsto_{DG.iso} dg-Rel \alpha$$

*<proof>*

### Further properties of the canonical dagger

**lemma (in  $\mathcal{Z}$ )** *dghm-cn-comp-dghm-dag-Rel-dghm-dag-Rel*:

$$\dagger_{DG.Rel} \alpha \circ_{DGHM} \dagger_{DG.Rel} \alpha = dghm-id (dg-Rel \alpha)$$

*<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]<sup>6</sup>). 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,  $\alpha$ -*Par* is defined as a digraph of sets and partial functions in  $V_\alpha$

**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* =  $\mathcal{Z} \ \alpha + \text{vfsequence } T + \text{ArrVal}: \text{vsv } \langle T(\text{ArrVal}) \rangle$  **for**  $\alpha \ T +$

**assumes** *arr-Par-length*[*dg-Rel-shared-cs-simps*, *dg-Par-cs-simps*]:

*vcard*  $T = 3_{\mathbb{N}}$

**and** *arr-Par-ArrVal-vdomain*:  $\mathcal{D}_\circ (T(\text{ArrVal})) \subseteq_\circ T(\text{ArrDom})$

**and** *arr-Par-ArrVal-vrange*:  $\mathcal{R}_\circ (T(\text{ArrVal})) \subseteq_\circ T(\text{ArrCod})$

**and** *arr-Par-ArrDom-in-Vset*:  $T(\text{ArrDom}) \in_\circ \text{Vset } \alpha$

**and** *arr-Par-ArrCod-in-Vset*:  $T(\text{ArrCod}) \in_\circ \text{Vset } \alpha$

Elementary properties.

**sublocale** *arr-Par*  $\subseteq$  *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*  $\alpha' \ T$

*<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** *vsu*  $r$

**and**  $\mathcal{D}_\circ r \subseteq_\circ a$

**and**  $\mathcal{R}_\circ r \subseteq_\circ b$

**and**  $a \in_\circ \text{Vset } \alpha$

**and**  $b \in_\circ \text{Vset } \alpha$

**shows** *arr-Par*  $\alpha [r, a, b]_\circ$

*<proof>*

**lemma** *arr-Par-arr-RelI*:

**assumes** *arr-Rel*  $\alpha \ T$  **and** *vsu*  $(T(\text{ArrVal}))$

**shows** *arr-Par*  $\alpha \ T$

*<proof>*

---

<sup>6</sup><https://ncatlab.org/nlab/show/partial+function>

**lemma** *arr-Par-arr-RelD*:  
**assumes** *arr-Par*  $\alpha$  *T*  
**shows** *arr-Rel*  $\alpha$  *T* **and** *vsv* ( $T(\downarrow ArrVal)$ )  
*<proof>*

**lemma** *arr-Par-arr-RelE*:  
**assumes** *arr-Par*  $\alpha$  *T*  
**obtains** *arr-Rel*  $\alpha$  *T* **and** *vsv* ( $T(\downarrow ArrVal)$ )  
*<proof>*

Further properties.

**lemma** *arr-Par-eqI*:  
**assumes** *arr-Par*  $\alpha$  *S*  
**and** *arr-Par*  $\alpha$  *T*  
**and**  $S(\downarrow ArrVal) = T(\downarrow ArrVal)$   
**and**  $S(\downarrow ArrDom) = T(\downarrow ArrDom)$   
**and**  $S(\downarrow ArrCod) = T(\downarrow ArrCod)$   
**shows**  $S = T$   
*<proof>*

**lemma** *small-arr-Par[simp]*: *small*  $\{T. \text{arr-Par } \alpha T\}$   
*<proof>*

**lemma** *set-Collect-arr-Par[simp]*:  
 $T \in_{\circ} \text{set} (\text{Collect} (\text{arr-Par } \alpha)) \longleftrightarrow \text{arr-Par } \alpha T$   
*<proof>*

## Composition

**abbreviation** (*input*) *comp-Par* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{Par} \rangle$  55)  
**where** *comp-Par*  $\equiv$  *comp-Rel*

**lemma** *arr-Par-comp-Par[dg-Par-cs-intros]*:  
**assumes** *arr-Par*  $\alpha$  *S* **and** *arr-Par*  $\alpha$  *T*  
**shows** *arr-Par*  $\alpha$  ( $S \circ_{Par} T$ )  
*<proof>*

**lemma** *arr-Par-comp-Par-ArrVal-app*:  
**assumes** *arr-Par*  $\alpha$  *S*  
**and** *arr-Par*  $\alpha$  *T*  
**and**  $x \in_{\circ} \mathcal{D}_{\circ} (T(\downarrow ArrVal))$   
**and**  $T(\downarrow ArrVal)(x) \in_{\circ} \mathcal{D}_{\circ} (S(\downarrow ArrVal))$   
**shows**  $(S \circ_{Par} T)(\downarrow ArrVal)(x) = S(\downarrow ArrVal)(T(\downarrow ArrVal)(x))$   
*<proof>*

## Inclusion

**abbreviation** (*input*) *incl-Par* ::  $V \Rightarrow V \Rightarrow V$   
**where** *incl-Par*  $\equiv$  *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*  $\alpha$  (*incl-Par*  $A$   $B$ )  
*<proof>*

## Identity

**abbreviation** (*input*) *id-Par* ::  $V \Rightarrow V$

where  $id\text{-}Par \equiv id\text{-}Rel$

**lemma** (in  $Z$ )  $arr\text{-}Par\text{-}id\text{-}ParI$ :

assumes  $A \in_0 Vset \alpha$

shows  $arr\text{-}Par \alpha (id\text{-}Par A)$

$\langle proof \rangle$

**lemma**  $arr\text{-}Par\text{-}comp\text{-}Par\text{-}id\text{-}Par\text{-}left[dg\text{-}Par\text{-}cs\text{-}simps]$ :

assumes  $arr\text{-}Par \alpha f$  and  $f(\downarrow ArrCod) = A$

shows  $id\text{-}Par A \circ_{Par} f = f$

$\langle proof \rangle$

**lemma**  $arr\text{-}Par\text{-}comp\text{-}Par\text{-}id\text{-}Par\text{-}right[dg\text{-}Par\text{-}cs\text{-}simps]$ :

assumes  $arr\text{-}Par \alpha f$  and  $f(\downarrow ArrDom) = A$

shows  $f \circ_{Par} id\text{-}Par A = f$

$\langle proof \rangle$

### 3.13.3 $Par$ as a digraph

#### Definition and elementary properties

**definition**  $dg\text{-}Par :: V \Rightarrow V$

where  $dg\text{-}Par \alpha =$

[  
 $Vset \alpha,$   
 $set \{T. arr\text{-}Par \alpha T\},$   
 $(\lambda T \in_0 set \{T. arr\text{-}Par \alpha T\}. T(\downarrow ArrDom)),$   
 $(\lambda T \in_0 set \{T. arr\text{-}Par \alpha T\}. T(\downarrow ArrCod))$   
 ]<sub>o</sub>

Components.

**lemma**  $dg\text{-}Par\text{-}components$ :

shows  $dg\text{-}Par \alpha(\downarrow Obj) = Vset \alpha$

and  $dg\text{-}Par \alpha(\downarrow Arr) = set \{T. arr\text{-}Par \alpha T\}$

and  $dg\text{-}Par \alpha(\downarrow Dom) = (\lambda T \in_0 set \{T. arr\text{-}Par \alpha T\}. T(\downarrow ArrDom))$

and  $dg\text{-}Par \alpha(\downarrow Cod) = (\lambda T \in_0 set \{T. arr\text{-}Par \alpha T\}. T(\downarrow ArrCod))$

$\langle proof \rangle$

#### Object

**lemma**  $dg\text{-}Par\text{-}Obj\text{-}iff$ :  $x \in_0 dg\text{-}Par \alpha(\downarrow Obj) \longleftrightarrow x \in_0 Vset \alpha$

$\langle proof \rangle$

#### Arrow

**lemma**  $dg\text{-}Par\text{-}Arr\text{-}iff[dg\text{-}Par\text{-}cs\text{-}simps]$ :  $x \in_0 dg\text{-}Par \alpha(\downarrow Arr) \longleftrightarrow arr\text{-}Par \alpha x$

$\langle proof \rangle$

#### Domain

**mk-VLambda**  $dg\text{-}Par\text{-}components(3)$

$[vsu dg\text{-}Par\text{-}Dom\text{-}vsu[dg\text{-}Par\text{-}cs\text{-}intros]]$

$[vdomain dg\text{-}Par\text{-}Dom\text{-}vdomain[dg\text{-}Par\text{-}cs\text{-}simps]]$

$[app dg\text{-}Par\text{-}Dom\text{-}app[unfolded set\text{-}Collect\text{-}arr\text{-}Par, dg\text{-}Par\text{-}cs\text{-}simps]]$

**lemma**  $dg\text{-}Par\text{-}Dom\text{-}vrange$ :  $\mathcal{R}_o (dg\text{-}Par \alpha(\downarrow Dom)) \subseteq_0 dg\text{-}Par \alpha(\downarrow Obj)$

$\langle proof \rangle$

**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*:  $\mathcal{R}_\circ (dg-Par \ \alpha \ (\text{Cod})) \sqsubseteq_\circ dg-Par \ \alpha \ (\text{Obj})$   
 ⟨*proof*⟩

**Arrow with a domain and a codomain**

Rules.

**lemma** *dg-Par-is-arrI*:  
**assumes** *arr-Par*  $\alpha$  *S* **and**  $S(\text{ArrDom}) = A$  **and**  $S(\text{ArrCod}) = B$   
**shows**  $S : A \mapsto_{dg-Par \ \alpha} B$   
 ⟨*proof*⟩

**lemmas** [*dg-Par-cs-intros*] = *dg-Par-is-arrI*

**lemma** *dg-Par-is-arrD*:  
**assumes**  $S : A \mapsto_{dg-Par \ \alpha} B$   
**shows** *arr-Par*  $\alpha$  *S*  
**and** [*dg-cs-simps*]:  $S(\text{ArrDom}) = A$   
**and** [*dg-cs-simps*]:  $S(\text{ArrCod}) = B$   
 ⟨*proof*⟩

**lemma** *dg-Par-is-arrE*:  
**assumes**  $S : A \mapsto_{dg-Par \ \alpha} B$   
**obtains** *arr-Par*  $\alpha$  *S* **and**  $S(\text{ArrDom}) = A$  **and**  $S(\text{ArrCod}) = B$   
 ⟨*proof*⟩

Elementary properties.

**lemma** *dg-Par-is-arr-dg-Rel-is-arr*:  
**assumes**  $r : a \mapsto_{dg-Par \ \alpha} b$   
**shows**  $r : a \mapsto_{dg-Rel \ \alpha} b$   
 ⟨*proof*⟩

**lemma** *dg-Par-Hom-vsubset-dg-Rel-Hom*:  
**assumes**  $a \in_\circ dg-Par \ \alpha \ (\text{Obj})$   $b \in_\circ dg-Par \ \alpha \ (\text{Obj})$   
**shows**  $\text{Hom} (dg-Par \ \alpha) a b \sqsubseteq_\circ \text{Hom} (dg-Rel \ \alpha) a b$   
 ⟨*proof*⟩

**lemma** (in  $\mathcal{Z}$ ) *dg-Par-incl-Par-is-arr*:  
**assumes**  $A \in_\circ dg-Par \ \alpha \ (\text{Obj})$  **and**  $B \in_\circ dg-Par \ \alpha \ (\text{Obj})$  **and**  $A \sqsubseteq_\circ B$   
**shows** *incl-Par*  $A B : A \mapsto_{dg-Par \ \alpha} B$   
 ⟨*proof*⟩

**lemma** (in  $\mathcal{Z}$ ) *dg-Par-incl-Par-is-arr*'[*dg-Par-cs-intros*]:  
**assumes**  $A \in_\circ dg-Par \ \alpha \ (\text{Obj})$   
**and**  $B \in_\circ dg-Par \ \alpha \ (\text{Obj})$   
**and**  $A \sqsubseteq_\circ B$   
**and**  $A' = A$   
**and**  $B' = B$   
**shows** *incl-Par*  $A B : A' \mapsto_{dg-Par \ \alpha} B'$   
 ⟨*proof*⟩

**lemmas** [*dg-Par-cs-intros*] =  $\mathcal{Z}.dg\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr'$

**Par is a digraph**

**lemma** (in  $\mathcal{Z}$ ) *dg-Par-Hom-vifunion-in-Vset*:

**assumes**  $X \in_{\circ} Vset\ \alpha$  **and**  $Y \in_{\circ} Vset\ \alpha$

**shows**  $(\bigcup_{\circ} A \in_{\circ} X. \bigcup_{\circ} B \in_{\circ} Y. Hom\ (dg\text{-}Par\ \alpha)\ A\ B) \in_{\circ} Vset\ \alpha$

*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *digraph-dg-Par*: *digraph*  $\alpha$  (*dg-Par*  $\alpha$ )

*<proof>*

**Par is a wide subdigraph of Rel**

**lemma** (in  $\mathcal{Z}$ ) *wide-subdigraph-dg-Par-dg-Rel*: *dg-Par*  $\alpha \subseteq_{DG.wide\alpha} dg\text{-}Rel\ \alpha$

*<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,  $\alpha$ -*Set* is defined as a digraph of sets and binary relations in the set  $V_\alpha$ .

**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* =  $\mathcal{Z} \alpha + \text{vfsequence } T + \text{ArrVal: vsu } \langle T(\text{ArrVal}) \rangle$  **for**  $\alpha T +$

**assumes** *arr-Set-length*[*dg-Rel-shared-cs-simps*, *dg-Set-cs-simps*]:

$\text{vcard } T = 3_{\mathbb{N}}$

**and** *arr-Set-ArrVal-vdomain*[*dg-Rel-shared-cs-simps*, *dg-Set-cs-simps*]:

$\mathcal{D}_\circ (T(\text{ArrVal})) = T(\text{ArrDom})$

**and** *arr-Set-ArrVal-vrange*:  $\mathcal{R}_\circ (T(\text{ArrVal})) \subseteq_\circ T(\text{ArrCod})$

**and** *arr-Set-ArrDom-in-Vset*:  $T(\text{ArrDom}) \in_\circ \text{Vset } \alpha$

**and** *arr-Set-ArrCod-in-Vset*:  $T(\text{ArrCod}) \in_\circ \text{Vset } \alpha$

**lemmas** [*dg-Set-cs-simps*] = *arr-Set.arr-Set-ArrVal-vdomain*

Elementary properties.

**sublocale** *arr-Set*  $\subseteq$  *arr-Par*

*<proof>*

Rules.

**lemma** (**in** *arr-Set*) *arr-Set-axioms'*[*dg-cs-intros*, *dg-Set-cs-intros*]:

**assumes**  $\alpha' = \alpha$

**shows** *arr-Set*  $\alpha' T$

*<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** *vsu r*

**and**  $\mathcal{D}_\circ r = a$

**and**  $\mathcal{R}_\circ r \subseteq_\circ b$

**and**  $a \in_\circ \text{Vset } \alpha$

**and**  $b \in_\circ \text{Vset } \alpha$

**shows** *arr-Set*  $\alpha [r, a, b]_\circ$

*<proof>*

**lemma** *arr-Set-arr-ParI*:

**assumes** *arr-Par*  $\alpha T$  **and**  $\mathcal{D}_\circ (T(\text{ArrVal})) = T(\text{ArrDom})$

**shows** *arr-Set*  $\alpha T$

*<proof>*

**lemma** *arr-Set-arr-ParD*:  
**assumes** *arr-Set*  $\alpha$  *T*  
**shows** *arr-Par*  $\alpha$  *T* **and**  $\mathcal{D}_\circ (T(\downarrow ArrVal)) = T(\downarrow ArrDom)$   
 $\langle proof \rangle$

**lemma** *arr-Set-arr-ParE*:  
**assumes** *arr-Set*  $\alpha$  *T*  
**obtains** *arr-Par*  $\alpha$  *T* **and**  $\mathcal{D}_\circ (T(\downarrow ArrVal)) = T(\downarrow ArrDom)$   
 $\langle proof \rangle$

Further properties.

**lemma** *arr-Set-eqI*:  
**assumes** *arr-Set*  $\alpha$  *S*  
**and** *arr-Set*  $\alpha$  *T*  
**and**  $S(\downarrow ArrVal) = T(\downarrow ArrVal)$   
**and**  $S(\downarrow ArrDom) = T(\downarrow ArrDom)$   
**and**  $S(\downarrow ArrCod) = T(\downarrow ArrCod)$   
**shows**  $S = T$   
 $\langle proof \rangle$

**lemma** *small-arr-Set[simp]*: *small*  $\{T. arr-Set \alpha T\}$   
 $\langle proof \rangle$

**lemma** *set-Collect-arr-Set[simp]*:  
 $T \in_\circ set (Collect (arr-Set \alpha)) \longleftrightarrow arr-Set \alpha T$   
 $\langle proof \rangle$

## Composition

See [39]).

**abbreviation** (*input*) *comp-Set*  $:: V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{Set} \rangle$  55)  
**where** *comp-Set*  $\equiv comp-Rel$

**lemma** *arr-Set-comp-Set[dg-Set-cs-intros]*:  
**assumes** *arr-Set*  $\alpha$  *S* **and** *arr-Set*  $\alpha$  *T* **and**  $\mathcal{R}_\circ (T(\downarrow ArrVal)) \subseteq_\circ \mathcal{D}_\circ (S(\downarrow ArrVal))$   
**shows** *arr-Set*  $\alpha$   $(S \circ_{Set} T)$   
 $\langle proof \rangle$

**lemma** *arr-Set-comp-Set-ArrVal-app*:  
**assumes** *arr-Set*  $\alpha$  *S*  
**and** *arr-Set*  $\alpha$  *T*  
**and**  $x \in_\circ \mathcal{D}_\circ (T(\downarrow ArrVal))$   
**and**  $T(\downarrow ArrVal)(x) \in_\circ \mathcal{D}_\circ (S(\downarrow ArrVal))$   
**shows**  $(S \circ_{Set} T)(\downarrow ArrVal)(x) = S(\downarrow ArrVal)(T(\downarrow ArrVal)(x))$   
 $\langle proof \rangle$

## Inclusion

**abbreviation** (*input*) *incl-Set*  $:: V \Rightarrow V \Rightarrow V$   
**where** *incl-Set*  $\equiv incl-Rel$

**lemma** (**in** *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*  $\alpha$   $(incl-Set A B)$   
 $\langle proof \rangle$

**Identity**

**abbreviation** (*input*)  $id\text{-Set} :: V \Rightarrow V$   
**where**  $id\text{-Set} \equiv id\text{-Rel}$

**lemma** (*in*  $Z$ )  $arr\text{-Set-id-SetI}$ :

**assumes**  $A \in_o Vset \alpha$   
**shows**  $arr\text{-Set} \alpha (id\text{-Set} A)$

*<proof>*

**lemma**  $arr\text{-Set-comp-Set-id-Set-left}[dg\text{-Set-cs-simps}]$ :

**assumes**  $arr\text{-Set} \alpha F$  **and**  $F(\text{ArrCod}) = A$   
**shows**  $id\text{-Set} A \circ_{Set} F = F$

*<proof>*

**lemma**  $arr\text{-Set-comp-Set-id-Set-right}[dg\text{-Set-cs-simps}]$ :

**assumes**  $arr\text{-Set} \alpha F$  **and**  $F(\text{ArrDom}) = A$   
**shows**  $F \circ_{Set} id\text{-Set} A = F$

*<proof>*

**3.14.3 Set as a digraph****Definition and elementary properties**

**definition**  $dg\text{-Set} :: V \Rightarrow V$

**where**  $dg\text{-Set} \alpha =$

[  
 $Vset \alpha,$   
 $set \{T. arr\text{-Set} \alpha T\},$   
 $(\lambda T \in_o set \{T. arr\text{-Set} \alpha T\}. T(\text{ArrDom})),$   
 $(\lambda T \in_o set \{T. arr\text{-Set} \alpha T\}. T(\text{ArrCod}))$   
 ]<sub>o</sub>

Components.

**lemma**  $dg\text{-Set-components}$ :

**shows**  $dg\text{-Set} \alpha(\text{Obj}) = Vset \alpha$   
**and**  $dg\text{-Set} \alpha(\text{Arr}) = set \{T. arr\text{-Set} \alpha T\}$   
**and**  $dg\text{-Set} \alpha(\text{Dom}) = (\lambda T \in_o set \{T. arr\text{-Set} \alpha T\}. T(\text{ArrDom}))$   
**and**  $dg\text{-Set} \alpha(\text{Cod}) = (\lambda T \in_o set \{T. arr\text{-Set} \alpha T\}. T(\text{ArrCod}))$

*<proof>*

**Object**

**lemma**  $dg\text{-Set-Obj-iff}$ :  $x \in_o dg\text{-Set} \alpha(\text{Obj}) \iff x \in_o Vset \alpha$

*<proof>*

**Arrow**

**lemma**  $dg\text{-Set-Arr-iff}[dg\text{-Set-cs-simps}]$ :  $x \in_o dg\text{-Set} \alpha(\text{Arr}) \iff arr\text{-Set} \alpha x$

*<proof>*

**Domain**

**mk-VLambda**  $dg\text{-Set-components}(3)$

$|vsu dg\text{-Set-Dom-vsuv}[dg\text{-Set-cs-intros]|$

$|vdomain dg\text{-Set-Dom-vdomain}[dg\text{-Set-cs-simps]|$

$|app dg\text{-Set-Dom-app}[unfolded set-Collect-arr-Set, dg\text{-Set-cs-simps]|$

**lemma**  $dg\text{-Set-Dom-vrange}$ :  $\mathcal{R}_o (dg\text{-Set} \alpha(\text{Dom})) \sqsubseteq_o dg\text{-Set} \alpha(\text{Obj})$

*<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*:  $\mathcal{R}_\circ (dg-Set \ \alpha \ (Cod)) \sqsubseteq_\circ dg-Set \ \alpha \ (Obj)$   
 ⟨*proof*⟩

**Arrow with a domain and a codomain**

Rules.

**lemma** *dg-Set-is-arrI*[*dg-Set-cs-intros*]:  
**assumes** *arr-Set*  $\alpha$  *S* **and**  $S(\downarrow ArrDom) = A$  **and**  $S(\downarrow ArrCod) = B$   
**shows**  $S : A \mapsto_{dg-Set \ \alpha} B$   
 ⟨*proof*⟩

**lemma** *dg-Set-is-arrD*:  
**assumes**  $S : A \mapsto_{dg-Set \ \alpha} B$   
**shows** *arr-Set*  $\alpha$  *S*  
**and** [*dg-cs-simps*]:  $S(\downarrow ArrDom) = A$   
**and** [*dg-cs-simps*]:  $S(\downarrow ArrCod) = B$   
 ⟨*proof*⟩

**lemma** *dg-Set-is-arrE*:  
**assumes**  $S : A \mapsto_{dg-Set \ \alpha} B$   
**obtains** *arr-Set*  $\alpha$  *S* **and**  $S(\downarrow ArrDom) = A$  **and**  $S(\downarrow ArrCod) = B$   
 ⟨*proof*⟩

**lemma** *dg-Set-ArrVal-vdomain*[*dg-Set-cs-simps, dg-cs-simps*]:  
**assumes**  $T : A \mapsto_{dg-Set \ \alpha} B$   
**shows**  $\mathcal{D}_\circ (T(\downarrow ArrVal)) = A$   
 ⟨*proof*⟩

Elementary properties.

**lemma** *dg-Set-ArrVal-app-vrange*[*dg-Set-cs-intros*]:  
**assumes**  $F : A \mapsto_{dg-Set \ \alpha} B$  **and**  $a \in_\circ A$   
**shows**  $F(\downarrow ArrVal)(\downarrow a) \in_\circ B$   
 ⟨*proof*⟩

**lemma** *dg-Set-is-arr-dg-Par-is-arr*:  
**assumes**  $T : A \mapsto_{dg-Set \ \alpha} B$   
**shows**  $T : A \mapsto_{dg-Par \ \alpha} B$   
 ⟨*proof*⟩

**lemma** *dg-Set-Hom-vsubset-dg-Par-Hom*:  
**assumes**  $a \in_\circ dg-Set \ \alpha \ (Obj)$   $b \in_\circ dg-Set \ \alpha \ (Obj)$   
**shows**  $Hom (dg-Set \ \alpha) \ a \ b \sqsubseteq_\circ Hom (dg-Par \ \alpha) \ a \ b$   
 ⟨*proof*⟩

**lemma** (in  $\mathcal{Z}$ ) *dg-Set-incl-Set-is-arr*:  
**assumes**  $A \in_\circ dg-Set \ \alpha \ (Obj)$  **and**  $B \in_\circ dg-Set \ \alpha \ (Obj)$  **and**  $A \sqsubseteq_\circ B$   
**shows** *incl-Set*  $A \ B : A \mapsto_{dg-Set \ \alpha} B$   
 ⟨*proof*⟩

**lemma** (in  $\mathcal{Z}$ ) *dg-Set-incl-Set-is-arr'*[*dg-cs-intros, dg-Set-cs-intros*]:

**assumes**  $A \in_{\circ} dg\text{-Set } \alpha(\text{Obj})$   
**and**  $B \in_{\circ} dg\text{-Set } \alpha(\text{Obj})$   
**and**  $A \subseteq_{\circ} B$   
**and**  $A' = A$   
**and**  $B' = B$   
**and**  $\mathfrak{C}' = dg\text{-Set } \alpha$   
**shows**  $incl\text{-Set } A B : A' \mapsto_{\mathfrak{C}'} B'$   
 ⟨proof⟩

**lemmas**  $[dg\text{-Set-cs-intros}] = \mathcal{Z}.dg\text{-Set-incl-Set-is-arr}'$

### *Set* is a digraph

**lemma** (in  $\mathcal{Z}$ ) *dg-Set-Hom-vifunior-in-Vset*:  
**assumes**  $X \in_{\circ} Vset \alpha$  **and**  $Y \in_{\circ} Vset \alpha$   
**shows**  $(\bigcup_{\circ} A \in_{\circ} X. \bigcup_{\circ} B \in_{\circ} Y. Hom (dg\text{-Set } \alpha) A B) \in_{\circ} Vset \alpha$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *digraph-dg-Set: digraph*  $\alpha (dg\text{-Set } \alpha)$   
 ⟨proof⟩

### *Set* is a wide subdigraph of *Par*

**lemma** (in  $\mathcal{Z}$ ) *wide-subdigraph-dg-Set-dg-Par*:  $dg\text{-Set } \alpha \subseteq_{DG.wide\alpha} dg\text{-Par } \alpha$   
 ⟨proof⟩

---

# 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*] =  $\mathcal{Z}$ - $\beta$

## 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]<sup>1</sup>. However, it is important to note that the term has other meanings in programming and computer science.

**definition**  $smc-dg :: V \Rightarrow V$

**where**  $smc-dg \mathfrak{C} = [\mathfrak{C}(Obj), \mathfrak{C}(Arr), \mathfrak{C}(Dom), \mathfrak{C}(Cod)]$ .

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**  $smc-dg \mathfrak{C}(Cod) = \mathfrak{C}(Cod)$

$\langle proof \rangle$

Regular definitions.

**lemma**  $smc-dg-is-arr[slicing-simps]: f : a \mapsto_{smc-dg \mathfrak{C}} b \longleftrightarrow f : a \mapsto_{\mathfrak{C}} b$

$\langle proof \rangle$

**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**  $\langle \circ_{A1} \rangle$  55)

**where**  $Comp-app \mathfrak{C} a b \equiv \mathfrak{C}(Comp)(a, b)$ .

**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**  $small-composable-arrs[simp]$ :

$small \{[g, f]_{\circ} \mid g f. \exists a b c. g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b\}$

$\langle proof \rangle$

Rules.

**lemma**  $composable-arrsI[smc-cs-intros]$ :

<sup>1</sup>[https://en.wikipedia.org/wiki/Object\\_slicing](https://en.wikipedia.org/wiki/Object_slicing)

**assumes**  $gf = [g, f]_o$  **and**  $g : b \mapsto_{\mathfrak{C}} c$  **and**  $f : a \mapsto_{\mathfrak{C}} b$   
**shows**  $gf \in_o$  *composable-arrs*  $\mathfrak{C}$   
 ⟨*proof*⟩

**lemma** *composable-arrsE[elim!]*:  
**assumes**  $gf \in_o$  *composable-arrs*  $\mathfrak{C}$   
**obtains**  $gf a b c$  **where**  $gf = [g, f]_o$  **and**  $g : b \mapsto_{\mathfrak{C}} c$  **and**  $f : a \mapsto_{\mathfrak{C}} b$   
 ⟨*proof*⟩

**lemma** *small-composable-arrs[simp]*:  
 $small \{ [g, f]_o \mid g f. g \in_o \mathfrak{C}(Arr) \wedge f \in_o \mathfrak{C}(Arr) \wedge \mathfrak{C}(Dom)(g) = \mathfrak{C}(Cod)(f) \}$   
 ⟨*proof*⟩

**lemma** *dg-composable-arrs'*:  
 $set \{ [g, f]_o \mid g f. g \in_o \mathfrak{C}(Arr) \wedge f \in_o \mathfrak{C}(Arr) \wedge \mathfrak{C}(Dom)(g) = \mathfrak{C}(Cod)(f) \} =$   
*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* =  $\mathcal{Z} \alpha + vfsequence \mathfrak{C} + Comp: vsv \langle \mathfrak{C}(Comp) \rangle$  **for**  $\alpha \mathfrak{C} +$   
**assumes** *smc-length[smc-cs-simps]*:  $vcard \mathfrak{C} = 5_{\mathbb{N}}$   
**and** *smc-digraph[slicing-intros]*:  $digraph \alpha (smc-dg \mathfrak{C})$   
**and** *smc-Comp-vdomain*:  $gf \in_o \mathcal{D}_o (\mathfrak{C}(Comp)) \longleftrightarrow$   
 $(\exists g f b c a. gf = [g, f]_o \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$   
**and** *smc-Comp-is-arr*:  
 $[[ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$   
**and** *smc-Comp-assoc[smc-cs-simps]*:  
 $[[ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies$   
 $(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

**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**  $Z \alpha$

**and** *vfsequence*  $\mathfrak{C}$

**and** *vsv* ( $\mathfrak{C}(\text{Comp})$ )

**and** *vcard*  $\mathfrak{C} = 5_{\mathbb{N}}$

**and** *vsv* ( $\mathfrak{C}(\text{Dom})$ )

**and** *vsv* ( $\mathfrak{C}(\text{Cod})$ )

**and**  $\mathcal{D}_o(\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_o(\mathfrak{C}(\text{Dom})) \subseteq_o \mathfrak{C}(\text{Obj})$

**and**  $\mathcal{D}_o(\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_o(\mathfrak{C}(\text{Cod})) \subseteq_o \mathfrak{C}(\text{Obj})$

**and**  $\wedge gf. gf \in_o \mathcal{D}_o(\mathfrak{C}(\text{Comp})) \longleftrightarrow$

$(\exists g f b c a. gf = [g, f]_o \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$

**and**  $\wedge b c g a f. [[ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$

**and**  $\wedge c d h b g a f. [[ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies$

$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

**and**  $\mathfrak{C}(\text{Obj}) \subseteq_o \text{Vset } \alpha$

**and**  $\wedge A B. [[ A \subseteq_o \mathfrak{C}(\text{Obj}); B \subseteq_o \mathfrak{C}(\text{Obj}); A \in_o \text{Vset } \alpha; B \in_o \text{Vset } \alpha ]] \implies$

$(\bigcup_o a \in_o A. \bigcup_o b \in_o B. \text{Hom } \mathfrak{C} a b) \in_o \text{Vset } \alpha$

**shows** *semicategory*  $\alpha \mathfrak{C}$

*<proof>*

**lemma** *semicategoryD'*:

**assumes** *semicategory*  $\alpha \mathfrak{C}$

**shows**  $Z \alpha$

**and** *vfsequence*  $\mathfrak{C}$

**and** *vsv* ( $\mathfrak{C}(\text{Comp})$ )

**and** *vcard*  $\mathfrak{C} = 5_{\mathbb{N}}$

**and** *vsv* ( $\mathfrak{C}(\text{Dom})$ )

**and** *vsv* ( $\mathfrak{C}(\text{Cod})$ )

**and**  $\mathcal{D}_o(\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_o(\mathfrak{C}(\text{Dom})) \subseteq_o \mathfrak{C}(\text{Obj})$

**and**  $\mathcal{D}_o(\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_o(\mathfrak{C}(\text{Cod})) \subseteq_o \mathfrak{C}(\text{Obj})$

**and**  $\wedge gf. gf \in_o \mathcal{D}_o(\mathfrak{C}(\text{Comp})) \longleftrightarrow$

$(\exists g f b c a. gf = [g, f]_o \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$

**and**  $\wedge b c g a f. [[ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$

**and**  $\wedge c d h b g a f. [[ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies$

$(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

**and**  $\mathfrak{C}(\text{Obj}) \subseteq_o \text{Vset } \alpha$

**and**  $\wedge A B. [[ A \subseteq_o \mathfrak{C}(\text{Obj}); B \subseteq_o \mathfrak{C}(\text{Obj}); A \in_o \text{Vset } \alpha; B \in_o \text{Vset } \alpha ]] \implies$

$(\bigcup_o a \in_o A. \bigcup_o b \in_o B. \text{Hom } \mathfrak{C} a b) \in_o \text{Vset } \alpha$

*<proof>*

**lemma** *semicategoryE'*:

**assumes** *semicategory*  $\alpha \mathfrak{C}$

**obtains**  $Z \alpha$

**and** *vfsequence*  $\mathfrak{C}$

**and** *vsv* ( $\mathfrak{C}(\text{Comp})$ )

**and** *vcard*  $\mathfrak{C} = 5_{\mathbb{N}}$

**and** *vsv* ( $\mathfrak{C}(\text{Dom})$ )

```

and vsv ( $\mathfrak{C}(\text{Cod})$ )
and  $\mathcal{D}_\circ (\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$ 
and  $\mathcal{R}_\circ (\mathfrak{C}(\text{Dom})) \subseteq_\circ \mathfrak{C}(\text{Obj})$ 
and  $\mathcal{D}_\circ (\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$ 
and  $\mathcal{R}_\circ (\mathfrak{C}(\text{Cod})) \subseteq_\circ \mathfrak{C}(\text{Obj})$ 
and  $\wedge gf. gf \in_\circ \mathcal{D}_\circ (\mathfrak{C}(\text{Comp})) \leftrightarrow$ 
  ( $\exists g f b c a. gf = [g, f]_\circ \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b$ )
and  $\wedge b c g a f. \llbracket g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b \rrbracket \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$ 
and  $\wedge c d h b g a f. \llbracket h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b \rrbracket \implies$ 
  ( $h \circ_{A\mathfrak{C}} g$ )  $\circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$ 
and  $\mathfrak{C}(\text{Obj}) \subseteq_\circ \text{Vset } \alpha$ 
and  $\wedge A B. \llbracket A \subseteq_\circ \mathfrak{C}(\text{Obj}); B \subseteq_\circ \mathfrak{C}(\text{Obj}); A \in_\circ \text{Vset } \alpha; B \in_\circ \text{Vset } \alpha \rrbracket \implies$ 
  ( $\bigcup_\circ a \in_\circ A. \bigcup_\circ b \in_\circ B. \text{Hom } \mathfrak{C} a b$ )  $\in_\circ \text{Vset } \alpha$ 
<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**<sup>2</sup>. 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  $\alpha$  <smc-dg  $\mathfrak{C}$ > <proof>

```

```

sublocale Dom: vsv < $\mathfrak{C}(\text{Dom})$ > <proof>

```

```

sublocale Cod: vsv < $\mathfrak{C}(\text{Cod})$ > <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

```

<sup>2</sup><https://lists.cam.ac.uk/pipermail/cl-isabelle-users/2019-September/msg00074.html>

**and** *smc-Cod-vcempty-if-Arr-vcempty* = *dg.dg-Cod-vcempty-if-Arr-vcempty*

**end**

**lemmas** [*smc-cs-intros*] =  
*semicategory.smc-is-arrD(1-3)*  
*semicategory.smc-Hom-in-Vset*

Elementary properties.

**lemma** *smc-eqI*:  
**assumes** *semicategory*  $\alpha$   $\mathfrak{A}$   
**and** *semicategory*  $\alpha$   $\mathfrak{B}$   
**and**  $\mathfrak{A}(\text{Obj}) = \mathfrak{B}(\text{Obj})$   
**and**  $\mathfrak{A}(\text{Arr}) = \mathfrak{B}(\text{Arr})$   
**and**  $\mathfrak{A}(\text{Dom}) = \mathfrak{B}(\text{Dom})$   
**and**  $\mathfrak{A}(\text{Cod}) = \mathfrak{B}(\text{Cod})$   
**and**  $\mathfrak{A}(\text{Comp}) = \mathfrak{B}(\text{Comp})$   
**shows**  $\mathfrak{A} = \mathfrak{B}$   
 $\langle \text{proof} \rangle$

**lemma** *smc-dg-eqI*:  
**assumes** *semicategory*  $\alpha$   $\mathfrak{A}$   
**and** *semicategory*  $\alpha$   $\mathfrak{B}$   
**and**  $\mathfrak{A}(\text{Comp}) = \mathfrak{B}(\text{Comp})$   
**and** *smc-dg*  $\mathfrak{A} = \text{smc-dg } \mathfrak{B}$   
**shows**  $\mathfrak{A} = \mathfrak{B}$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *semicategory*) *smc-def*:  $\mathfrak{C} = [\mathfrak{C}(\text{Obj}), \mathfrak{C}(\text{Arr}), \mathfrak{C}(\text{Dom}), \mathfrak{C}(\text{Cod}), \mathfrak{C}(\text{Comp})]$ .  
 $\langle \text{proof} \rangle$

**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**  $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\text{Comp}))$   
 $\langle \text{proof} \rangle$

**lemmas** [*smc-cs-intros*] = *semicategory.smc-Comp-vdomainI*

**lemma** (**in** *semicategory*) *smc-Comp-vdomainE*[*elim!*]:  
**assumes**  $gf \in_{\circ} \mathcal{D}_{\circ}(\mathfrak{C}(\text{Comp}))$   
**obtains**  $g f a b c$  **where**  $gf = [g, f]_{\circ}$  **and**  $g : b \mapsto_{\mathfrak{C}} c$  **and**  $f : a \mapsto_{\mathfrak{C}} b$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *semicategory*) *smc-Comp-vdomain-is-composable-arrrs*:  
 $\mathcal{D}_{\circ}(\mathfrak{C}(\text{Comp})) = \text{composable-arrrs } \mathfrak{C}$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *semicategory*) *smc-Comp-vrange*:  $\mathcal{R}_{\circ}(\mathfrak{C}(\text{Comp})) \subseteq_{\circ} \mathfrak{C}(\text{Arr})$   
 $\langle \text{proof} \rangle$

**sublocale** *semicategory*  $\subseteq$  *Comp*: *pbinop*  $\langle \mathfrak{C}(\text{Arr}) \rangle$   $\langle \mathfrak{C}(\text{Comp}) \rangle$   
 $\langle \text{proof} \rangle$

Size.

**lemma** (**in** *semicategory*) *smc-Comp-vsubset-Vset*:  $\mathfrak{C}(\text{Comp}) \subseteq_{\circ} \text{Vset } \alpha$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *semicategory*) *smc-semicategory-in-Vset-4*:  $\mathfrak{C} \in_{\circ} \text{Vset } (\alpha + 4_{\mathbb{N}})$

*<proof>*

**lemma** (in *semicategory*) *smc-Comp-in-Vset*:

**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{C}(\text{Comp}) \in_{\circ} \text{Vset } \beta$   
*<proof>*

**lemma** (in *semicategory*) *smc-in-Vset*:

**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{C} \in_{\circ} \text{Vset } \beta$   
*<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*  $\{\mathfrak{C}. \text{semicategory } \alpha \mathfrak{C}\}$

*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *semicategories-in-Vset*:

**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows** *set*  $\{\mathfrak{C}. \text{semicategory } \alpha \mathfrak{C}\} \in_{\circ} \text{Vset } \beta$   
*<proof>*

**lemma** *semicategory-if-semicategory*:

**assumes** *semicategory*  $\beta \mathfrak{C}$   
**and**  $\mathcal{Z} \alpha$   
**and**  $\mathfrak{C}(\text{Obj}) \subseteq_{\circ} \text{Vset } \alpha$   
**and**  $\bigwedge A B. [\![ A \subseteq_{\circ} \mathfrak{C}(\text{Obj}); B \subseteq_{\circ} \mathfrak{C}(\text{Obj}); A \in_{\circ} \text{Vset } \alpha; B \in_{\circ} \text{Vset } \alpha \]\!] \implies$   
 $(\bigcup_{a \in_{\circ} A} a. \bigcup_{b \in_{\circ} B} b. \text{Hom } \mathfrak{C} a b) \in_{\circ} \text{Vset } \alpha$   
**shows** *semicategory*  $\alpha \mathfrak{C}$   
*<proof>*

Further properties.

**lemma** (in *semicategory*) *smc-Comp-venempty-if-Arr-venempty*:

**assumes**  $\mathfrak{C}(\text{Arr}) = 0$   
**shows**  $\mathfrak{C}(\text{Comp}) = 0$   
*<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*  $\mathfrak{C} = [\mathfrak{C}(\text{Obj}), \mathfrak{C}(\text{Arr}), \mathfrak{C}(\text{Cod}), \mathfrak{C}(\text{Dom}), \text{flip } (\mathfrak{C}(\text{Comp}))]$ .

Components.

**lemma** *op-smc-components*:

**shows** [*smc-op-simps*]: *op-smc*  $\mathfrak{C}(\text{Obj}) = \mathfrak{C}(\text{Obj})$   
**and** [*smc-op-simps*]: *op-smc*  $\mathfrak{C}(\text{Arr}) = \mathfrak{C}(\text{Arr})$   
**and** [*smc-op-simps*]: *op-smc*  $\mathfrak{C}(\text{Dom}) = \mathfrak{C}(\text{Cod})$   
**and** [*smc-op-simps*]: *op-smc*  $\mathfrak{C}(\text{Cod}) = \mathfrak{C}(\text{Dom})$   
**and** *op-smc*  $\mathfrak{C}(\text{Comp}) = \text{flip } (\mathfrak{C}(\text{Comp}))$   
*<proof>*

**lemma** *op-smc-component-intros*[*smc-op-intros*]:

**shows**  $a \in_{\circ} \mathfrak{C}(\text{Obj}) \implies a \in_{\circ} \text{op-smc } \mathfrak{C}(\text{Obj})$   
**and**  $f \in_{\circ} \mathfrak{C}(\text{Arr}) \implies f \in_{\circ} \text{op-smc } \mathfrak{C}(\text{Arr})$   
 ⟨*proof*⟩

Slicing.

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

⟨*proof*⟩

Regular definitions.

**lemma** *op-smc-Comp-vdomain*[*smc-op-simps*]:

$\mathcal{D}_{\circ} (\text{op-smc } \mathfrak{C}(\text{Comp})) = (\mathcal{D}_{\circ} (\mathfrak{C}(\text{Comp})))^{-1}$ .  
 ⟨*proof*⟩

**lemma** *op-smc-is-arr*[*smc-op-simps*]:  $f : b \mapsto_{\text{op-smc } \mathfrak{C}} a \iff f : a \mapsto_{\mathfrak{C}} b$

⟨*proof*⟩

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

**lemma** (in *semicategory*) *op-smc-Comp-vrange*[*smc-op-simps*]:

$\mathcal{R}_{\circ} (\text{op-smc } \mathfrak{C}(\text{Comp})) = \mathcal{R}_{\circ} (\mathfrak{C}(\text{Comp}))$   
 ⟨*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**  $g \circ_{A \text{ op-smc } \mathfrak{C}} f = f \circ_{A \mathfrak{C}} g$   
 ⟨*proof*⟩

**lemmas** [*smc-op-simps*] = *semicategory.op-smc-Comp*

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

⟨*proof*⟩

## Further properties

**lemma** (in *semicategory*) *semicategory-op*[*smc-op-intros*]:

*semicategory*  $\alpha$  (*op-smc*  $\mathfrak{C}$ )

⟨*proof*⟩

**lemmas** *semicategory-op*[*smc-op-intros*] = *semicategory.semicategory-op*

**lemma** (in *semicategory*) *smc-op-smc-op-smc*[*smc-op-simps*]:  $\text{op-smc } (\text{op-smc } \mathfrak{C}) = \mathfrak{C}$

⟨*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**  $\text{op-smc } \mathfrak{A} = \text{op-smc } \mathfrak{B} \iff \mathfrak{A} = \mathfrak{B}$

⟨*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* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-monic-arr*  $\mathfrak{C} b c m \leftrightarrow$

$m : b \mapsto_{\mathfrak{C}} c \wedge$

(

$\forall f g a.$

$f : a \mapsto_{\mathfrak{C}} b \longrightarrow g : a \mapsto_{\mathfrak{C}} b \longrightarrow m \circ_{A\mathfrak{C}} f = m \circ_{A\mathfrak{C}} g \longrightarrow f = g$

)

**syntax** *-is-monic-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

( $\langle \cdot : - \mapsto_{\text{mon}1} \cdot \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-monic-arr*  $\hat{=}$  *is-monic-arr*

**translations**  $m : b \mapsto_{\text{mon}\mathfrak{C}} c \hat{=} \text{CONST } \textit{is-monic-arr} \mathfrak{C} b c m$

**definition** *is-epic-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-epic-arr*  $\mathfrak{C} a b e \equiv e : b \mapsto_{\text{monop-smc}\mathfrak{C}} a$

**syntax** *-is-epic-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

( $\langle \cdot : - \mapsto_{\text{epi}1} \cdot \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-epic-arr*  $\hat{=}$  *is-epic-arr*

**translations**  $e : a \mapsto_{\text{epi}\mathfrak{C}} b \hat{=} \text{CONST } \textit{is-epic-arr} \mathfrak{C} a b e$

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**  $\bigwedge f g c. [[ f : b \mapsto_{\mathfrak{C}} c; g : b \mapsto_{\mathfrak{C}} c; f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e ] ] \implies$   
 $f = g$

**shows**  $e : a \mapsto_{\text{epi}\mathfrak{C}} b$   
 ⟨proof⟩

**lemma** *is-epic-arr-is-arr*[*smc-arrow-cs-intros, dest*]:

**assumes**  $e : a \mapsto_{\text{epi}\mathfrak{C}} b$   
**shows**  $e : a \mapsto_{\mathfrak{C}} b$   
 ⟨proof⟩

**lemma** (in *semicategory*) *is-epic-arrD*[*dest*]:

**assumes**  $e : a \mapsto_{\text{epi}\mathfrak{C}} b$   
**shows**  $e : a \mapsto_{\mathfrak{C}} b$   
**and**  $\bigwedge f g c. [\![ f : b \mapsto_{\mathfrak{C}} c; g : b \mapsto_{\mathfrak{C}} c; f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e ]\!] \implies$   
 $f = g$

⟨proof⟩

**lemma** (in *semicategory*) *is-epic-arrE*[*elim!*]:

**assumes**  $e : a \mapsto_{\text{epi}\mathfrak{C}} b$   
**obtains**  $e : a \mapsto_{\mathfrak{C}} b$   
**and**  $\bigwedge f g c. [\![ f : b \mapsto_{\mathfrak{C}} c; g : b \mapsto_{\mathfrak{C}} c; f \circ_{A\mathfrak{C}} e = g \circ_{A\mathfrak{C}} e ]\!] \implies$   
 $f = g$

⟨proof⟩

Elementary properties.

**lemma** (in *semicategory*) *op-smc-is-epic-arr*[*smc-op-simps*]:

$f : b \mapsto_{\text{epi op-smc } \mathfrak{C}} a \iff f : a \mapsto_{\text{mon } \mathfrak{C}} b$   
 ⟨proof⟩

**lemma** (in *semicategory*) *op-smc-is-monic-arr*[*smc-op-simps*]:

$f : b \mapsto_{\text{mon op-smc } \mathfrak{C}} a \iff f : a \mapsto_{\text{epi } \mathfrak{C}} b$   
 ⟨proof⟩

**lemma** (in *semicategory*) *smc-Comp-is-monic-arr*[*smc-arrow-cs-intros*]:

**assumes**  $g : b \mapsto_{\text{mon } \mathfrak{C}} c$  **and**  $f : a \mapsto_{\text{mon } \mathfrak{C}} b$   
**shows**  $g \circ_{A\mathfrak{C}} f : a \mapsto_{\text{mon } \mathfrak{C}} c$

⟨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**  $g : b \mapsto_{\text{epi } \mathfrak{C}} c$  **and**  $f : a \mapsto_{\text{epi } \mathfrak{C}} b$   
**shows**  $g \circ_{A\mathfrak{C}} f : a \mapsto_{\text{epi } \mathfrak{C}} c$

⟨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**  $g \circ_{A\mathfrak{C}} f : a \mapsto_{\text{mon } \mathfrak{C}} c$   
**shows**  $f : a \mapsto_{\text{mon } \mathfrak{C}} b$

⟨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**  $f \circ_{A\mathfrak{C}} g : a \mapsto_{\text{epi } \mathfrak{C}} c$   
**shows**  $f : b \mapsto_{\text{epi } \mathfrak{C}} c$

⟨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**  $-is-idem-arr :: V \Rightarrow V \Rightarrow V \Rightarrow bool$  ( $\langle \cdot : \mapsto_{ide1} \cdot \rangle$  [51, 51] 51)

**syntax-consts**  $-is-idem-arr \equiv is-idem-arr$

**translations**  $f : \mapsto_{ide\mathfrak{C}} b \equiv CONST is-idem-arr \mathfrak{C} b f$

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*)  $op-smc-is-idem-arr[smc-op-simps]$ :

$f : \mapsto_{ide\mathfrak{C}} b \longleftrightarrow f : \mapsto_{ide\mathfrak{C}} b$   
 $\langle proof \rangle$

## 4.2.7 Terminal object and initial object

See Chapter I-5 in [39].

**definition**  $obj-terminal :: V \Rightarrow V \Rightarrow bool$

**where**  $obj-terminal \mathfrak{C} t \longleftrightarrow$   
 $t \in_{\circ} \mathfrak{C}(\mathit{Obj}) \wedge (\forall a. a \in_{\circ} \mathfrak{C}(\mathit{Obj}) \longrightarrow (\exists!f. f : a \mapsto_{\mathfrak{C}} t))$

**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**  $a \in_{\circ} \mathfrak{C}(\mathit{Obj})$  **and**  $\wedge b. b \in_{\circ} \mathfrak{C}(\mathit{Obj}) \Longrightarrow \exists!f. f : a \mapsto_{\mathfrak{C}} b$   
**shows**  $obj-initial \mathfrak{C} a$   
 $\langle proof \rangle$

**lemma**  $obj-initialD[dest]$ :

**assumes**  $obj-initial \mathfrak{C} a$   
**shows**  $a \in_{\circ} \mathfrak{C}(\mathit{Obj})$  **and**  $\wedge b. b \in_{\circ} \mathfrak{C}(\mathit{Obj}) \Longrightarrow \exists!f. f : a \mapsto_{\mathfrak{C}} b$   
 $\langle proof \rangle$

**lemma**  $obj-initialE[elim]$ :

**assumes**  $obj-initial \mathfrak{C} a$   
**obtains**  $a \in_{\circ} \mathfrak{C}(\mathit{Obj})$  **and**  $\wedge b. b \in_{\circ} \mathfrak{C}(\mathit{Obj}) \Longrightarrow \exists!f. f : a \mapsto_{\mathfrak{C}} b$   
 $\langle proof \rangle$

Elementary properties.

**lemma**  $op-smc-obj-initial[smc-op-simps]$ :

$obj-initial (op-smc \mathfrak{C}) = obj-terminal \mathfrak{C}$   
 $\langle proof \rangle$

**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* ::  $V \Rightarrow V \Rightarrow \text{bool}$   
 where *obj-null*  $\mathfrak{C}$   $a \iff \text{obj-initial } \mathfrak{C} \ a \wedge \text{obj-terminal } \mathfrak{C} \ a$

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*  $\mathfrak{C}$ )  $a = \text{obj-null } \mathfrak{C} \ a$   
 ⟨*proof*⟩

### 4.2.9 Zero arrow

**definition** *is-zero-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 where *is-zero-arr*  $\mathfrak{C}$   $a \ b \ h \iff$   
 ( $\exists z \ g \ f. \text{obj-null } \mathfrak{C} \ z \wedge h = g \circ_{A\mathfrak{C}} f \wedge f : a \mapsto_{\mathfrak{C}} z \wedge g : z \mapsto_{\mathfrak{C}} b$ )

**syntax** *-is-zero-arr* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 ( $\langle \cdot : \cdot \mapsto_1 \cdot \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-zero-arr*  $\equiv$  *is-zero-arr*

**translations**  $h : a \mapsto_{0\mathfrak{C}} b \equiv \text{CONST } \text{is-zero-arr } \mathfrak{C} \ a \ b \ h$

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  $\exists z \ g \ f. \text{obj-null } \mathfrak{C} \ z \wedge h = g \circ_{A\mathfrak{C}} f \wedge f : a \mapsto_{\mathfrak{C}} z \wedge g : z \mapsto_{\mathfrak{C}} b$   
 ⟨*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*]:

$f : b \mapsto_{0\text{-}op\text{-}smc} \mathfrak{C} a \iff f : a \mapsto_{0\mathfrak{C}} b$   
 ⟨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* =  $\mathcal{Z} \alpha + \text{vfsequence } \mathfrak{C} + \text{Comp}: \text{vsv } \langle \mathfrak{C}(\text{Comp}) \rangle$  **for**  $\alpha \mathfrak{C} +$

**assumes** *tiny-smc-length*[*smc-cs-simps*]:  $\text{vcard } \mathfrak{C} = 5_{\mathbb{N}}$

**and** *tiny-smc-tiny-digraph*[*slicing-intros*]: *tiny-digraph*  $\alpha$  (*smc-dg*  $\mathfrak{C}$ )

**and** *tiny-smc-Comp-vdomain*:  $gf \in_{\circ} \mathcal{D}_{\circ} (\mathfrak{C}(\text{Comp})) \longleftrightarrow$

$(\exists g f b c a. gf = [g, f]_{\circ} \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$

**and** *tiny-smc-Comp-is-arr*[*smc-cs-intros*]:

$[[ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$

**and** *tiny-smc-assoc*[*smc-cs-simps*]:

$[[ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies$   
 $(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$

**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*  $\alpha' \mathfrak{C}$

*<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*  $\alpha \mathfrak{C}$  **and**  $\mathfrak{C}(\text{Obj}) \in_{\circ} \text{Vset } \alpha$  **and**  $\mathfrak{C}(\text{Arr}) \in_{\circ} \text{Vset } \alpha$

**shows** *tiny-semicategory*  $\alpha \mathfrak{C}$

*<proof>*

**lemma** *tiny-semicategoryI''*:

**assumes**  $\mathcal{Z} \alpha$

**and** *vfsequence*  $\mathfrak{C}$

**and** *vsv* ( $\mathfrak{C}(\text{Comp})$ )

**and**  $\text{vcard } \mathfrak{C} = 5_{\mathbb{N}}$

**and** *vsv* ( $\mathfrak{C}(\text{Dom})$ )

**and** *vsv* ( $\mathfrak{C}(\text{Cod})$ )

**and**  $\mathcal{D}_{\circ} (\mathfrak{C}(\text{Dom})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_{\circ} (\mathfrak{C}(\text{Dom})) \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

**and**  $\mathcal{D}_{\circ} (\mathfrak{C}(\text{Cod})) = \mathfrak{C}(\text{Arr})$

**and**  $\mathcal{R}_{\circ} (\mathfrak{C}(\text{Cod})) \subseteq_{\circ} \mathfrak{C}(\text{Obj})$

**and**  $\wedge gf. gf \in_0 \mathcal{D}_0(\mathfrak{C}(\mathcal{C}(\mathit{Comp}))) \leftrightarrow$   
 $(\exists g f b c a. gf = [g, f]_0 \wedge g : b \mapsto_{\mathfrak{C}} c \wedge f : a \mapsto_{\mathfrak{C}} b)$   
**and**  $\wedge b c g a f. [[ g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies g \circ_{A\mathfrak{C}} f : a \mapsto_{\mathfrak{C}} c$   
**and**  $\wedge c d h b g a f. [[ h : c \mapsto_{\mathfrak{C}} d; g : b \mapsto_{\mathfrak{C}} c; f : a \mapsto_{\mathfrak{C}} b ]] \implies$   
 $(h \circ_{A\mathfrak{C}} g) \circ_{A\mathfrak{C}} f = h \circ_{A\mathfrak{C}} (g \circ_{A\mathfrak{C}} f)$   
**and**  $\mathfrak{C}(\mathit{Obj}) \in_0 \mathit{Vset} \alpha$   
**and**  $\mathfrak{C}(\mathit{Arr}) \in_0 \mathit{Vset} \alpha$   
**shows** *tiny-semicategory*  $\alpha \mathfrak{C}$   
 $\langle \mathit{proof} \rangle$

Slicing.

**context** *tiny-semicategory*

**begin**

**interpretation**  $dg: \mathit{tiny-digraph} \alpha \langle \mathit{smc-dg} \mathfrak{C} \rangle \langle \mathit{proof} \rangle$

**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*  $\subseteq$  *semicategory*

$\langle \mathit{proof} \rangle$

**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*:  $\mathfrak{C}(\mathit{Comp}) \in_0 \mathit{Vset} \alpha$

$\langle \mathit{proof} \rangle$

**lemma** (**in** *tiny-semicategory*) *tiny-smc-in-Vset*:  $\mathfrak{C} \in_0 \mathit{Vset} \alpha$

$\langle \mathit{proof} \rangle$

**lemma** *small-tiny-semicategories*[*simp*]: *small*  $\{\mathfrak{C}. \mathit{tiny-semicategory} \alpha \mathfrak{C}\}$

$\langle \mathit{proof} \rangle$

**lemma** *tiny-semicategories-vsubset-Vset*:

*set*  $\{\mathfrak{C}. \mathit{tiny-semicategory} \alpha \mathfrak{C}\} \subseteq_0 \mathit{Vset} \alpha$

$\langle \mathit{proof} \rangle$

**lemma** (**in** *semicategory*) *smc-tiny-semicategory-if-ge-Limit*:

**assumes**  $Z \beta$  **and**  $\alpha \in_0 \beta$

**shows** *tiny-semicategory*  $\beta \mathfrak{C}$

$\langle \mathit{proof} \rangle$

## Opposite tiny semicategory

**lemma** (**in** *tiny-semicategory*) *tiny-semicategory-op*:

*tiny-semicategory*  $\alpha$  (*op-smc*  $\mathfrak{C}$ )

$\langle \mathit{proof} \rangle$

**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]<sup>3</sup>.

```

locale finite-semicategory =  $\mathcal{Z}$   $\alpha$  + vfsequence  $\mathfrak{C}$  + Comp: vsv  $\langle \mathfrak{C}(\mathfrak{Comp}) \rangle$  for  $\alpha$   $\mathfrak{C}$  +
assumes fin-smc-length[smc-cs-simps]: vcard  $\mathfrak{C} = 5_{\mathbb{N}}$ 
and fin-smc-finite-digraph[slicing-intros]: finite-digraph  $\alpha$  (smc-dg  $\mathfrak{C}$ )
and fin-smc-Comp-vdomain: gf  $\in_{\circ} \mathcal{D}_{\circ} (\mathfrak{C}(\mathfrak{Comp})) \longleftrightarrow$ 
  ( $\exists$  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 ]]  $\implies$  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 ]]  $\implies$ 
  (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  $\alpha'$   $\mathfrak{C}$ 
   $\langle$ proof $\rangle$ 

```

```

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  $\alpha$   $\mathfrak{C}$  and vfinite ( $\mathfrak{C}(\mathfrak{Obj})$ ) and vfinite ( $\mathfrak{C}(\mathfrak{Arr})$ )
shows finite-semicategory  $\alpha$   $\mathfrak{C}$ 
   $\langle$ proof $\rangle$ 

```

```

lemma finite-semicategoryI'':
assumes tiny-semicategory  $\alpha$   $\mathfrak{C}$  and vfinite ( $\mathfrak{C}(\mathfrak{Obj})$ ) and vfinite ( $\mathfrak{C}(\mathfrak{Arr})$ )
shows finite-semicategory  $\alpha$   $\mathfrak{C}$ 
   $\langle$ proof $\rangle$ 

```

Slicing.

```

context finite-semicategory
begin

```

```

interpretation dg: finite-digraph  $\alpha$   $\langle$ smc-dg  $\mathfrak{C}$  $\rangle$   $\langle$ proof $\rangle$ 

```

```

lemmas-with [unfolded slicing-simps]:
  fin-smc-Obj-vfinite[smc-small-cs-intros] = dg.fin-dg-Obj-vfinite

```

<sup>3</sup><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*  $\subseteq$  *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*:  $\mathfrak{C} \in_0 Vset \alpha$   
 ⟨*proof*⟩

Size.

**lemma** *small-finite-semicategories*[*simp*]: *small* { $\mathfrak{C}. finite-semicategory \alpha \mathfrak{C}$ }  
 ⟨*proof*⟩

**lemma** *finite-semicategories-ubset-Vset*:  
*set* { $\mathfrak{C}. finite-semicategory \alpha \mathfrak{C}$ }  $\subseteq_0 Vset \alpha$   
 ⟨*proof*⟩

### Opposite finite semicategory

**lemma** (in *finite-semicategory*) *finite-semicategory-op*:  
*finite-semicategory*  $\alpha$  (*op-smc*  $\mathfrak{C}$ )  
 ⟨*proof*⟩

**lemmas** *finite-semicategory-op*[*smc-op-intros*] =  
*finite-semicategory.fin-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*  $\mathfrak{C}$  =

[ $\mathfrak{C}(\text{ObjMap})$ ,  $\mathfrak{C}(\text{ArrMap})$ , *smc-dg* ( $\mathfrak{C}(\text{HomDom})$ ), *smc-dg* ( $\mathfrak{C}(\text{HomCod})$ )].

Components.

**lemma** *smcf-dghm-components*:

**shows** [*slicing-simps*]: *smcf-dghm*  $\mathfrak{F}(\text{ObjMap})$  =  $\mathfrak{F}(\text{ObjMap})$

**and** [*slicing-simps*]: *smcf-dghm*  $\mathfrak{F}(\text{ArrMap})$  =  $\mathfrak{F}(\text{ArrMap})$

**and** [*slicing-commute*]: *smcf-dghm*  $\mathfrak{F}(\text{HomDom})$  = *smc-dg* ( $\mathfrak{F}(\text{HomDom})$ )

**and** [*slicing-commute*]: *smcf-dghm*  $\mathfrak{F}(\text{HomCod})$  = *smc-dg* ( $\mathfrak{F}(\text{HomCod})$ )

*<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* =

$Z$   $\alpha$  +

*vfsequence*  $\mathfrak{F}$  +

*HomDom*: *semicategory*  $\alpha$   $\mathfrak{A}$  +

*HomCod*: *semicategory*  $\alpha$   $\mathfrak{B}$

**for**  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +

**assumes** *smcf-length*[*smc-cs-simps*]: *vcard*  $\mathfrak{F}$  =  $4_{\mathbb{N}}$

**and** *smcf-is-dghm*[*slicing-intros*]:

*smcf-dghm*  $\mathfrak{F}$  : *smc-dg*  $\mathfrak{A}$   $\mapsto_{DG\alpha}$  *smc-dg*  $\mathfrak{B}$

**and** *smcf-HomDom*[*smc-cs-simps*]:  $\mathfrak{F}(\text{HomDom})$  =  $\mathfrak{A}$

**and** *smcf-HomCod*[*smc-cs-simps*]:  $\mathfrak{F}(\text{HomCod})$  =  $\mathfrak{B}$

**and** *smcf-ArrMap-Comp*[*smc-cs-simps*]:  $[[g : b \mapsto_{\mathfrak{A}} c; f : a \mapsto_{\mathfrak{A}} b]] \implies$

$\mathfrak{F}(\text{ArrMap})(g \circ_{\mathfrak{A}\mathfrak{A}} f) = \mathfrak{F}(\text{ArrMap})(g) \circ_{\mathfrak{A}\mathfrak{B}} \mathfrak{F}(\text{ArrMap})(f)$

**syntax** *-is-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ :/ } - \mapsto_{SMC1} - \rangle \text{ [51, 51, 51] 51} \rangle$

**syntax-consts** *-is-semifunctor*  $\equiv$  *is-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B} \equiv \text{CONST } \textit{is-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-cn-semifunctor*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \equiv \mathfrak{F} : \textit{op-smc } \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**syntax** *-is-cn-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - \text{ :/ } - \text{ }_{SMC} \mapsto_{\mapsto 1} - \rangle \text{ [51, 51, 51] 51} \rangle$

**syntax-consts** *-is-cn-semifunctor*  $\equiv$  *is-cn-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{SMC} \mapsto_{\mapsto \alpha} \mathfrak{B} \equiv \text{CONST } \textit{is-cn-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** *all-smcfs* ::  $V \Rightarrow V$   
**where** *all-smcfs*  $\alpha \equiv \text{set } \{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}\}$

**abbreviation** *smcfs* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$   
**where** *smcfs*  $\alpha \mathfrak{A} \mathfrak{B} \equiv \text{set } \{\mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}\}$

**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**  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$   
**and**  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$   
**shows** *smcf-dghm*  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG\alpha} \mathfrak{B}'$   
*<proof>*

**lemma** *cn-dghm-comp-is-dghm*:  
**assumes**  $\mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$   
**shows** *smcf-dghm*  $\mathfrak{F} : \text{op-dg } (\text{smc-dg } \mathfrak{A}) \mapsto \mapsto_{DG\alpha} \text{smc-dg } \mathfrak{B}$   
*<proof>*

**lemma** *cn-dghm-comp-is-dghm'*[*slicing-intros*]:  
**assumes**  $\mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{A}' = \text{op-dg } (\text{smc-dg } \mathfrak{A})$   
**and**  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$   
**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**  $\mathfrak{F}(\text{HomDom}) = \mathfrak{A}$   
**and**  $\mathfrak{F}(\text{HomCod}) = \mathfrak{B}$   
**and** *vsv* ( $\mathfrak{F}(\text{ObjMap})$ )  
**and** *vsv* ( $\mathfrak{F}(\text{ArrMap})$ )  
**and**  $\mathcal{D}_\circ (\mathfrak{F}(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$   
**and**  $\mathcal{R}_\circ (\mathfrak{F}(\text{ObjMap})) \subseteq_\circ \mathfrak{B}(\text{Obj})$   
**and**  $\mathcal{D}_\circ (\mathfrak{F}(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$   
**and**  $\wedge a b f. f : a \mapsto_{\mathfrak{A}} b \implies$

$\mathfrak{F}(\text{ArrMap})(f) : \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(b)$   
**and**  $\wedge b c g a f. \llbracket g : b \mapsto_{\mathfrak{A}} c; f : a \mapsto_{\mathfrak{A}} b \rrbracket \implies$   
 $\mathfrak{F}(\text{ArrMap})(g \circ_{A\mathfrak{A}} f) = \mathfrak{F}(\text{ArrMap})(g) \circ_{A\mathfrak{B}} \mathfrak{F}(\text{ArrMap})(f)$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMCA} \mathfrak{B}$   
 $\langle \text{proof} \rangle$

**lemma** *is-semifunctorD'*:

**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMCA} \mathfrak{B}$

**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**  $\mathfrak{F}(\text{HomDom}) = \mathfrak{A}$

**and**  $\mathfrak{F}(\text{HomCod}) = \mathfrak{B}$

**and** *vsv*  $(\mathfrak{F}(\text{ObjMap}))$

**and** *vsv*  $(\mathfrak{F}(\text{ArrMap}))$

**and**  $\mathcal{D}_o(\mathfrak{F}(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$

**and**  $\mathcal{R}_o(\mathfrak{F}(\text{ObjMap})) \subseteq_o \mathfrak{B}(\text{Obj})$

**and**  $\mathcal{D}_o(\mathfrak{F}(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$

**and**  $\wedge a b f. f : a \mapsto_{\mathfrak{A}} b \implies$

$\mathfrak{F}(\text{ArrMap})(f) : \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(b)$

**and**  $\wedge b c g a f. \llbracket g : b \mapsto_{\mathfrak{A}} c; f : a \mapsto_{\mathfrak{A}} b \rrbracket \implies$

$\mathfrak{F}(\text{ArrMap})(g \circ_{A\mathfrak{A}} f) = \mathfrak{F}(\text{ArrMap})(g) \circ_{A\mathfrak{B}} \mathfrak{F}(\text{ArrMap})(f)$

$\langle \text{proof} \rangle$

**lemma** *is-semifunctorE'*:

**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMCA} \mathfrak{B}$

**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**  $\mathfrak{F}(\text{HomDom}) = \mathfrak{A}$

**and**  $\mathfrak{F}(\text{HomCod}) = \mathfrak{B}$

**and** *vsv*  $(\mathfrak{F}(\text{ObjMap}))$

**and** *vsv*  $(\mathfrak{F}(\text{ArrMap}))$

**and**  $\mathcal{D}_o(\mathfrak{F}(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$

**and**  $\mathcal{R}_o(\mathfrak{F}(\text{ObjMap})) \subseteq_o \mathfrak{B}(\text{Obj})$

**and**  $\mathcal{D}_o(\mathfrak{F}(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$

**and**  $\wedge a b f. f : a \mapsto_{\mathfrak{A}} b \implies$

$\mathfrak{F}(\text{ArrMap})(f) : \mathfrak{F}(\text{ObjMap})(a) \mapsto_{\mathfrak{B}} \mathfrak{F}(\text{ObjMap})(b)$

**and**  $\wedge b c g a f. \llbracket g : b \mapsto_{\mathfrak{A}} c; f : a \mapsto_{\mathfrak{A}} b \rrbracket \implies$

$\mathfrak{F}(\text{ArrMap})(g \circ_{A\mathfrak{A}} f) = \mathfrak{F}(\text{ArrMap})(g) \circ_{A\mathfrak{B}} \mathfrak{F}(\text{ArrMap})(f)$

$\langle \text{proof} \rangle$

Slicing.

**context** *is-semifunctor*

**begin**

**interpretation** *dghm*: *is-dghm*  $\alpha \langle \text{smc-dg } \mathfrak{A} \rangle \langle \text{smc-dg } \mathfrak{B} \rangle \langle \text{smcf-dghm } \mathfrak{F} \rangle$

$\langle \text{proof} \rangle$

**sublocale** *ObjMap*: *vsv*  $\langle \mathfrak{F}(\text{ObjMap}) \rangle$

$\langle \text{proof} \rangle$

**sublocale** *ArrMap*: *vsv*  $\langle \mathfrak{F}(\text{ArrMap}) \rangle$

$\langle \text{proof} \rangle$

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

*smcf-ObjMap-vsν* = *dghm.dghm-ObjMap-vsν*  
**and** *smcf-ArrMap-vsν* = *dghm.dghm-ArrMap-vsν*  
**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-νsubset-Vset* = *dghm.dghm-ObjMap-νsubset-Vset*  
**and** *smcf-ArrMap-νsubset-Vset* = *dghm.dghm-ArrMap-νsubset-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-νsubset-Hom* =  
*dghm.dghm-vimage-dghm-ArrMap-νsubset-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**  $\mathfrak{F}(\downarrow ArrMap)(\downarrow f \circ_{A\mathfrak{A}} g) = \mathfrak{F}(\downarrow ArrMap)(\downarrow g) \circ_{A\mathfrak{B}} \mathfrak{F}(\downarrow ArrMap)(\downarrow f)$   
*<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**  $\mathfrak{G}(\downarrow ObjMap) = \mathfrak{F}(\downarrow ObjMap)$   
**and**  $\mathfrak{G}(\downarrow ArrMap) = \mathfrak{F}(\downarrow ArrMap)$   
**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*  $\mathfrak{G} = \text{smcf-dghm } \mathfrak{F}$   
**shows**  $\mathfrak{G} = \mathfrak{F}$   
 ⟨*proof*⟩

**lemma** (in *is-semifunctor*) *smcf-def*:  
 $\mathfrak{F} = [\mathfrak{F}(\text{ObjMap}), \mathfrak{F}(\text{ArrMap}), \mathfrak{F}(\text{HomDom}), \mathfrak{F}(\text{HomCod})]$ .  
 ⟨*proof*⟩

**lemma** (in *is-semifunctor*) *smcf-in-Vset*:  
**assumes**  $Z \beta$  **and**  $\alpha \in_o \beta$   
**shows**  $\mathfrak{F} \in_o \text{Vset } \beta$   
 ⟨*proof*⟩

**lemma** (in *is-semifunctor*) *smcf-is-semifunctor-if-ge-Limit*:  
**assumes**  $Z \beta$  **and**  $\alpha \in_o \beta$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\beta} \mathfrak{B}$   
 ⟨*proof*⟩

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

**lemma** (in *is-semifunctor*) *smcf-in-Vset-7*:  $\mathfrak{F} \in_o \text{Vset } (\alpha + 7_{\mathbb{N}})$   
 ⟨*proof*⟩

**lemma** (in  $Z$ ) *all-smcfs-in-Vset*:  
**assumes**  $Z \beta$  **and**  $\alpha \in_o \beta$   
**shows** *all-smcfs*  $\alpha \in_o \text{Vset } \beta$   
 ⟨*proof*⟩

**lemma** *small-smcfs[simp]*: *small*  $\{\mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}\}$   
 ⟨*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*  $\mathfrak{F} =$   
 $[\mathfrak{F}(\text{ObjMap}), \mathfrak{F}(\text{ArrMap}), \text{op-smc } (\mathfrak{F}(\text{HomDom})), \text{op-smc } (\mathfrak{F}(\text{HomCod}))]$ .

Components.

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

Slicing.

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

#### Further properties

**lemma** (in *is-semifunctor*) *is-semifunctor-op*:

*op-smcf*  $\mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \text{op-smc } \mathfrak{B}$   
 ⟨proof⟩

**lemma** (in *is-semifunctor*) *is-semifunctor-op'*:  
 assumes  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$  and  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$  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*  $\mathfrak{G} = \text{op-smcf } \mathfrak{F} \leftrightarrow \mathfrak{G} = \mathfrak{F}$   
 ⟨proof⟩

#### 4.4.4 Composition of covariant semifunctors

##### Definition and elementary properties

**abbreviation** (*input*) *smcf-comp* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \circ_{SMCF} \rangle$  55)  
 where *smcf-comp*  $\equiv$  *dghm-comp*

Slicing.

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

##### Object map

**lemma** *smcf-comp-ObjMap-vsuv*[*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 *vsuv*  $((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ObjMap}))$   
 ⟨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  $\mathcal{D}_\circ ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ObjMap})) = \mathfrak{A}(\text{Obj})$   
 ⟨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  $\mathcal{R}_\circ ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ObjMap})) \subseteq_\circ \mathfrak{C}(\text{Obj})$   
 ⟨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*]:  $a \in_\circ \mathfrak{A}(\text{Obj})$   
 shows  $(\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ObjMap})(a) = \mathfrak{G}(\text{ObjMap})(\mathfrak{F}(\text{ObjMap})(a))$   
 ⟨proof⟩

**Arrow map**

**lemma** *smcf-comp-ArrMap-vsuv*[*smc-cs-intros*]:  
 assumes  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 shows *vsu*  $((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ArrMap}))$   
*<proof>*

**lemma** *smcf-comp-ArrMap-vdomain*[*smc-cs-simps*]:  
 assumes  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 shows  $\mathcal{D}_\circ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ArrMap})) = \mathfrak{A}(\text{Arr})$   
*<proof>*

**lemma** *smcf-comp-ArrMap-vrange*:  
 assumes  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 shows  $\mathcal{R}_\circ((\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ArrMap})) \subseteq_\circ \mathfrak{C}(\text{Arr})$   
*<proof>*

**lemma** *smcf-comp-ArrMap-app*[*smc-cs-simps*]:  
 assumes  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and [*simp*]:  $f \in_\circ \mathfrak{A}(\text{Arr})$   
 shows  $(\mathfrak{G} \circ_{SMCF} \mathfrak{F})(\text{ArrMap})(f) = \mathfrak{G}(\text{ArrMap})(\mathfrak{F}(\text{ArrMap})(f))$   
*<proof>*

**Further properties**

**lemma** *smcf-comp-is-semifunctor*[*smc-cs-intros*]:  
 assumes  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 shows  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$   
*<proof>*

**lemma** *smcf-comp-assoc*[*smc-cs-simps*]:  
 assumes  $\mathfrak{H} : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$   
 and  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 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*]:  
 $op\text{-smcf}(\mathfrak{G} \circ_{SMCF} \mathfrak{F}) = op\text{-smcf} \mathfrak{G} \circ_{SMCF} op\text{-smcf} \mathfrak{F}$   
*<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**  $\langle_{SMCF^\circ}$  55)

where  $\mathfrak{G}_{SMCF^\circ} \mathfrak{F} =$   
 [  
 $\mathfrak{G}(\text{ObjMap}) \circ_\circ \mathfrak{F}(\text{ObjMap}),$   
 $\mathfrak{G}(\text{ArrMap}) \circ_\circ \mathfrak{F}(\text{ArrMap}),$   
 $op\text{-smc}(\mathfrak{F}(\text{HomDom})),$   
 $\mathfrak{G}(\text{HomCod})$   
 ] $\circ$

Components.

**lemma** *smcf-cn-comp-components*:

**shows**  $(\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{O}bjMap) = \mathfrak{G}(\mathcal{O}bjMap) \circ_\circ \mathfrak{F}(\mathcal{O}bjMap)$   
**and**  $(\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{A}rrMap) = \mathfrak{G}(\mathcal{A}rrMap) \circ_\circ \mathfrak{F}(\mathcal{A}rrMap)$   
**and**  $[smc-cn-cs-simps]: (\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{H}omDom) = op-smc (\mathfrak{F}(\mathcal{H}omDom))$   
**and**  $[smc-cn-cs-simps]: (\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{H}omCod) = \mathfrak{G}(\mathcal{H}omCod)$   
*<proof>*

Slicing.

**lemma** *smcf-dghm-smcf-cn-comp[slicing-commute]:*  
 $smcf-dghm \mathfrak{G}_{DGHM^\circ} smcf-dghm \mathfrak{F} = smcf-dghm (\mathfrak{G}_{SMCF^\circ} \mathfrak{F})$   
*<proof>*

### Object map: two contravariant semifunctors

**lemma** *smcf-cn-comp-ObjMap-usv[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**  $usv ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{O}bjMap))$   
*<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**  $\mathcal{D}_\circ ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{O}bjMap)) = \mathfrak{A}(\mathcal{O}bj)$   
*<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**  $\mathcal{R}_\circ ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{O}bjMap)) \subseteq_\circ \mathfrak{C}(\mathcal{O}bj)$   
*<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**  $a \in_\circ \mathfrak{A}(\mathcal{O}bj)$   
**shows**  $(\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{O}bjMap)(\downarrow a) = \mathfrak{G}(\mathcal{O}bjMap)(\downarrow \mathfrak{F}(\mathcal{O}bjMap)(\downarrow a))$   
*<proof>*

### Arrow map: two contravariant semifunctors

**lemma** *smcf-cn-comp-ArrMap-usv[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**  $usv ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{A}rrMap))$   
*<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**  $\mathcal{D}_\circ ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{A}rrMap)) = \mathfrak{A}(\mathcal{A}rr)$   
*<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**  $\mathcal{R}_\circ ((\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{A}rrMap)) \subseteq_\circ \mathfrak{C}(\mathcal{A}rr)$   
*<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**  $a \in_\circ \mathfrak{A}(\mathcal{A}rr)$   
**shows**  $(\mathfrak{G}_{SMCF^\circ} \mathfrak{F})(\mathcal{A}rrMap)(\downarrow a) = \mathfrak{G}(\mathcal{A}rrMap)(\downarrow \mathfrak{F}(\mathcal{A}rrMap)(\downarrow a))$   
*<proof>*

### Object map: contravariant and covariant semifunctors

**lemma** *smcf-cn-cov-comp-ObjMap-usv[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**  $vsu ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ObjMap))$   
*<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**  $\mathcal{D}_{\circ} ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ObjMap)) = \mathfrak{A}(Obj)$   
*<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**  $\mathcal{R}_{\circ} ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ObjMap)) \subseteq_{\circ} \mathfrak{C}(Obj)$   
*<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**  $a \in_{\circ} \mathfrak{A}(Obj)$   
**shows**  $(\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ObjMap)(a) = \mathfrak{G}(ObjMap)(\mathfrak{F}(ObjMap)(a))$   
*<proof>*

### Arrow map: contravariant and covariant semifunctors

**lemma** *smcf-cn-cov-comp-ArrMap-vsuv[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**  $vsu ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ArrMap))$   
*<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**  $\mathcal{D}_{\circ} ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ArrMap)) = \mathfrak{A}(Arr)$   
*<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**  $\mathcal{R}_{\circ} ((\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ArrMap)) \subseteq_{\circ} \mathfrak{C}(Arr)$   
*<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**  $f \in_{\circ} \mathfrak{A}(Arr)$   
**shows**  $(\mathfrak{G}_{SMCF^{\circ}} \mathfrak{F})(ArrMap)(f) = \mathfrak{G}(ArrMap)(\mathfrak{F}(ArrMap)(f))$   
*<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*  $\alpha$  **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**  $\mathfrak{B} : \mathfrak{B} \mapsto \mapsto_{SMC\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \xrightarrow{SMC} \mapsto \alpha \mathfrak{B}$

**shows**  $\mathfrak{B} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \xrightarrow{SMC} \mapsto \alpha \mathfrak{C}$

*<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*  $\equiv$  *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*  $\alpha$   $\langle$  *smc-dg*  $\mathfrak{C}$   $\rangle$  *<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*  $\equiv$  *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*  $\alpha \mathfrak{C}$   
 and *semicategory*  $\alpha \mathfrak{D}$   
 and  $f : a \mapsto_{\mathfrak{D}} a$   
 and [*simp*]:  $f \circ_{A\mathfrak{D}} f = f$   
 shows *smcf-const*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$   
 ⟨proof⟩

**lemma** *smcf-const-is-semifunctor'*[*smc-cs-intros*]:  
 assumes *semicategory*  $\alpha \mathfrak{C}$   
 and *semicategory*  $\alpha \mathfrak{D}$   
 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*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 ⟨proof⟩

### Further properties

**lemma** (in *is-semifunctor*) *smcf-smcf-comp-smcf-const*[*smc-cs-simps*]:  
 assumes *semicategory*  $\alpha \mathfrak{C}$  and  $f : a \mapsto_{\mathfrak{C}} a$  and  $f \circ_{A\mathfrak{C}} f = f$   
 shows *smcf-const*  $\mathfrak{B} \mathfrak{C} a f \circ_{SMCF} \mathfrak{F} = \text{smcf-const } \mathfrak{A} \mathfrak{C} a f$   
 ⟨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*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$  for  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$   
 assumes *ft-smcf-is-ft-dghm*:  
 $\text{smcf-dghm } \mathfrak{F} : \text{smc-dg } \mathfrak{A} \mapsto_{DG.\text{faithful}\alpha} \text{smc-dg } \mathfrak{B}$

**syntax** *-is-ft-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 (⟨(- :/ -  $\mapsto_{SMC.\text{faithful}}$  -)⟩ [51, 51, 51] 51)

**syntax-consts** *-is-ft-semifunctor*  $\equiv$  *is-ft-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.\text{faithful}\alpha} \mathfrak{B} \equiv \text{CONST } \text{is-ft-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**lemma** (in *is-ft-semifunctor*) *ft-smcf-is-ft-dghm'*[*slicing-intros*]:  
 assumes  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$  and  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$   
 shows *smcf-dghm*  $\mathfrak{F} : \mathfrak{A}' \mapsto_{DG.\text{faithful}\alpha} \mathfrak{B}'$   
 ⟨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  $\mathfrak{F} : \mathfrak{A}' \mapsto_{SMC.\text{faithful}\alpha'} \mathfrak{B}'$   
 ⟨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  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and  $\bigwedge a b. [[ a \in_{\circ} \mathfrak{A}(\text{Obj}); b \in_{\circ} \mathfrak{A}(\text{Obj}) ]] \implies v11 (\mathfrak{F}(\text{ArrMap}) \uparrow^l_{\circ} \text{Hom } \mathfrak{A} a b)$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.\text{faithful}\alpha} \mathfrak{B}$   
 ⟨proof⟩

**lemma** *is-ft-semifunctorD'*:  
 assumes  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.\text{faithful}\alpha} \mathfrak{B}$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and  $\bigwedge a b. [[ a \in_{\circ} \mathfrak{A}(\text{Obj}); b \in_{\circ} \mathfrak{A}(\text{Obj}) ]] \implies v11 (\mathfrak{F}(\text{ArrMap}) \uparrow^l_{\circ} \text{Hom } \mathfrak{A} a b)$   
 ⟨proof⟩

**lemma** *is-ft-semifunctorE'*:

**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{B}$

**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) \]\!] \implies v11 (\mathfrak{F}(ArrMap) \uparrow^l \text{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) \]\!] \implies g = f$

**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{B}$

*<proof>*

Elementary properties.

**context** *is-ft-semifunctor*

**begin**

**interpretation** *dghm*: *is-ft-dghm*  $\alpha$   $\langle smc-dg\ \mathfrak{A} \rangle$   $\langle smc-dg\ \mathfrak{B} \rangle$   $\langle smcf-dghm\ \mathfrak{F} \rangle$

*<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*  $\mathfrak{F} : op-smc\ \mathfrak{A} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} op-smc\ \mathfrak{B}$

*<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*  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{B}'$

*<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**  $\mathfrak{G} : \mathfrak{B} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{B}$

**shows**  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.f\ aithful\ \alpha} \mathfrak{C}$

*<proof>*

### 4.4.9 Full semifunctor

#### Definition and elementary properties

See Chapter I-3 in [39].

**locale** *is-fl-semifunctor* = *is-semifunctor*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  **for**  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +

**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* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 ( $\langle (- \text{ :/ } - \mapsto \mapsto_{SMC.full} -) \rangle$  [51, 51, 51] 51)  
**syntax-consts** *-is-fl-semifunctor*  $\equiv$  *is-fl-semifunctor*  
**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.full} \mathfrak{B} \equiv \text{CONST } is-fl-semifunctor \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F}$

**lemma** (in *is-fl-semifunctor*) *fl-smcf-is-fl-dghm'*[*slicing-intros*]:  
**assumes**  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$  **and**  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$   
**shows** *smcf-dghm*  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.full} \mathfrak{B}'$   
*<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**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC} \mathfrak{B}$   
**and**  $\bigwedge a b. [[ a \in_o \mathfrak{A}(\text{Obj}); b \in_o \mathfrak{A}(\text{Obj}) ] ] \implies$   
 $\mathfrak{F}(\text{ArrMap}) \text{ ' } \circ (\text{Hom } \mathfrak{A} \ a \ b) = \text{Hom } \mathfrak{B} (\mathfrak{F}(\text{ObjMap})(a)) (\mathfrak{F}(\text{ObjMap})(b))$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.full} \mathfrak{B}$   
*<proof>*

**lemma** *is-fl-semifunctorD'*:  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.full} \mathfrak{B}$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC} \mathfrak{B}$   
**and**  $\bigwedge a b. [[ a \in_o \mathfrak{A}(\text{Obj}); b \in_o \mathfrak{A}(\text{Obj}) ] ] \implies$   
 $\mathfrak{F}(\text{ArrMap}) \text{ ' } \circ (\text{Hom } \mathfrak{A} \ a \ b) = \text{Hom } \mathfrak{B} (\mathfrak{F}(\text{ObjMap})(a)) (\mathfrak{F}(\text{ObjMap})(b))$   
*<proof>*

**lemma** *is-fl-semifunctorE'*:  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.full} \mathfrak{B}$   
**obtains**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC} \mathfrak{B}$   
**and**  $\bigwedge a b. [[ a \in_o \mathfrak{A}(\text{Obj}); b \in_o \mathfrak{A}(\text{Obj}) ] ] \implies$   
 $\mathfrak{F}(\text{ArrMap}) \text{ ' } \circ (\text{Hom } \mathfrak{A} \ a \ b) = \text{Hom } \mathfrak{B} (\mathfrak{F}(\text{ObjMap})(a)) (\mathfrak{F}(\text{ObjMap})(b))$   
*<proof>*

Elementary properties.

**context** *is-fl-semifunctor*  
**begin**

**interpretation** *dghm*: *is-fl-dghm*  $\alpha$   $\langle \text{smc-dg } \mathfrak{A} \rangle$   $\langle \text{smc-dg } \mathfrak{B} \rangle$   $\langle \text{smcf-dghm } \mathfrak{F} \rangle$   
*<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*  $\mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto \mapsto_{SMC.full} \text{op-smc } \mathfrak{B}$   
*<proof>*

**lemma** (in *is-fl-semifunctor*) *is-fl-semifunctor-op*'[*smc-op-intros*]:  
 assumes  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$  and  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$   
 shows  $\text{op-smcf } \mathfrak{F} : \mathfrak{A}' \mapsto_{SMC.full\alpha} \mathfrak{B}'$   
 ⟨*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  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC.full\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.full\alpha} \mathfrak{B}$   
 shows  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto_{SMC.full\alpha} \mathfrak{C}$   
 ⟨*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* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$   
 (⟨(- :/ -  $\mapsto_{SMC.ff1}$  -)⟩ [51, 51, 51] 51)

**syntax-consts** *-is-ff-semifunctor*  $\Rightarrow$  *is-ff-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.ff\alpha} \mathfrak{B} \Rightarrow \text{CONST } \textit{is-ff-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

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*  $\mathfrak{F} : \text{smc-dg } \mathfrak{A} \mapsto_{DG.ff\alpha} \text{smc-dg } \mathfrak{B}$   
 ⟨*proof*⟩

**lemma** (in *is-ff-semifunctor*) *ff-smcf-is-ff-dghm'*'[*slicing-intros*]:  
 assumes  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$  and  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$   
 shows *smcf-dghm*  $\mathfrak{F} : \mathfrak{A}' \mapsto_{DG.ff\alpha} \mathfrak{B}'$   
 ⟨*proof*⟩

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

##### Opposite fully faithful semifunctor

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

**lemma** (in *is-ff-semifunctor*) *is-ff-semifunctor-op'*'[*smc-op-intros*]:  
 assumes  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$  and  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$

**shows**  $op\text{-}smcf \mathfrak{F} : \mathfrak{A}' \mapsto_{SMC.ff\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $is\text{-}ff\text{-}semifunctor\text{-}op[smc\text{-}op\text{-}intros] =$   
 $is\text{-}ff\text{-}semifunctor.is\text{-}ff\text{-}semifunctor\text{-}op'$

**The composition of fully faithful semifunctors is a fully faithful semifunctor**

**lemma**  $smcf\text{-}comp\text{-}is\text{-}ff\text{-}semifunctor[smcf\text{-}cs\text{-}intros]:$   
**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC.ff\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.ff\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto_{SMC.ff\alpha} \mathfrak{C}$   
 ⟨proof⟩

#### 4.4.11 Isomorphism of semicategories

##### Definition and elementary properties

See Chapter I-3 in [39].

**locale**  $is\text{-}iso\text{-}semifunctor = is\text{-}semifunctor \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$  **for**  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$   
**assumes**  $iso\text{-}smcf\text{-}is\text{-}iso\text{-}dghm:$   
 $smcf\text{-}dghm \mathfrak{F} : smc\text{-}dg \mathfrak{A} \mapsto_{DG.iso\alpha} smc\text{-}dg \mathfrak{B}$

**syntax**  $is\text{-}iso\text{-}semifunctor :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$   
 ⟨⟨(- :/ -  $\mapsto_{SMC.iso1}$  -)⟩ [51, 51, 51] 51⟩

**syntax-consts**  $is\text{-}iso\text{-}semifunctor \equiv is\text{-}iso\text{-}semifunctor$

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.iso\alpha} \mathfrak{B} \equiv CONST is\text{-}iso\text{-}semifunctor \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**lemma** **(in**  $is\text{-}iso\text{-}semifunctor$ )  $iso\text{-}smcf\text{-}is\text{-}iso\text{-}dghm'[slicing\text{-}intros]:$   
**assumes**  $\mathfrak{A}' = smc\text{-}dg \mathfrak{A}$   $\mathfrak{B}' = smc\text{-}dg \mathfrak{B}$   
**shows**  $smcf\text{-}dghm \mathfrak{F} : \mathfrak{A}' \mapsto_{DG.iso\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $[slicing\text{-}intros] = is\text{-}iso\text{-}semifunctor.iso\text{-}smcf\text{-}is\text{-}iso\text{-}dghm'$

Rules.

**lemma** **(in**  $is\text{-}iso\text{-}semifunctor$ )  $is\text{-}iso\text{-}semifunctor\text{-}axioms'[smcf\text{-}cs\text{-}intros]:$   
**assumes**  $\alpha' = \alpha$  **and**  $\mathfrak{A}' = \mathfrak{A}$  **and**  $\mathfrak{B}' = \mathfrak{B}$   
**shows**  $\mathfrak{F} : \mathfrak{A}' \mapsto_{SMC.iso\alpha'} \mathfrak{B}'$   
 ⟨proof⟩

**mk-ide rf**  $is\text{-}iso\text{-}semifunctor\text{-}def[unfolding is\text{-}iso\text{-}semifunctor\text{-}axioms\text{-}def]$   
 |intro  $is\text{-}iso\text{-}semifunctorI$  |  
 |dest  $is\text{-}iso\text{-}semifunctorD[dest]$  |  
 |elim  $is\text{-}iso\text{-}semifunctorE[elim]$  |

**lemma**  $is\text{-}iso\text{-}semifunctorI'$ :  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $v11 (\mathfrak{F}(ObjMap))$   
**and**  $v11 (\mathfrak{F}(ArrMap))$   
**and**  $\mathcal{R}_o (\mathfrak{F}(ObjMap)) = \mathfrak{B}(Obj)$   
**and**  $\mathcal{R}_o (\mathfrak{F}(ArrMap)) = \mathfrak{B}(Arr)$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.iso\alpha} \mathfrak{B}$   
 ⟨proof⟩

**lemma**  $is\text{-}iso\text{-}semifunctorD'$ :  
**assumes**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.iso\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $v11 (\mathfrak{F}(ObjMap))$

**and**  $v11$  ( $\mathfrak{F}(\downarrow ArrMap)$ )  
**and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ObjMap)$ ) =  $\mathfrak{B}(\downarrow Obj)$   
**and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ArrMap)$ ) =  $\mathfrak{B}(\downarrow Arr)$   
 ⟨*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**  $v11$  ( $\mathfrak{F}(\downarrow ObjMap)$ )  
**and**  $v11$  ( $\mathfrak{F}(\downarrow ArrMap)$ )  
**and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ObjMap)$ ) =  $\mathfrak{B}(\downarrow Obj)$   
**and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ArrMap)$ ) =  $\mathfrak{B}(\downarrow Arr)$   
 ⟨*proof*⟩

Elementary properties.

**context** *is-iso-semifunctor*  
**begin**

**interpretation** *dghm: is-iso-dghm*  $\alpha$  ⟨*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*:  $v11$  ⟨ $\mathfrak{F}(\downarrow ObjMap)$ ⟩

**rewrites**  $\mathcal{D}_\circ$  ( $\mathfrak{F}(\downarrow ObjMap)$ ) =  $\mathfrak{A}(\downarrow Obj)$  **and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ObjMap)$ ) =  $\mathfrak{B}(\downarrow Obj)$   
 ⟨*proof*⟩

**sublocale** *ArrMap*:  $v11$  ⟨ $\mathfrak{F}(\downarrow ArrMap)$ ⟩

**rewrites**  $\mathcal{D}_\circ$  ( $\mathfrak{F}(\downarrow ArrMap)$ ) =  $\mathfrak{A}(\downarrow Arr)$  **and**  $\mathcal{R}_\circ$  ( $\mathfrak{F}(\downarrow ArrMap)$ ) =  $\mathfrak{B}(\downarrow Arr)$   
 ⟨*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*  $\subseteq$  *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**  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC.iso\alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.iso\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto_{SMC.iso\alpha} \mathfrak{C}$   
*<proof>*

#### 4.4.12 Inverse semifunctor

**abbreviation** (*input*) *inv-smcf* ::  $V \Rightarrow V$   
**where** *inv-smcf*  $\equiv$  *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*  $\alpha$  *<smc-dg*  $\mathfrak{A}$  *<smc-dg*  $\mathfrak{B}$  *<smcf-dghm*  $\mathfrak{F}$   
*<proof>*

**lemmas-with** [*unfolded slicing-simps slicing-commute*]:  
*inv-smcf-ObjMap-v11* = *dghm.inv-dghm-ObjMap-v11*  
**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-v11* = *dghm.inv-dghm-ArrMap-v11*  
**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-v11*  
*is-iso-semifunctor.inv-smcf-ArrMap-v11*

**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**  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} = smcf-id \mathfrak{A}$

**and**  $\mathfrak{F} \circ_{SMCF} \mathfrak{G} = smcf-id \mathfrak{B}$

**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*]:  $inv-smcf \mathfrak{F} \circ_{SMCF} \mathfrak{F} = smcf-id \mathfrak{A}$

**and** [*smcf-cs-simps*]:  $\mathfrak{F} \circ_{SMCF} inv-smcf \mathfrak{F} = smcf-id \mathfrak{B}$

*<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*  $\alpha \mathfrak{A} + R$ : *semicategory*  $\alpha \mathfrak{B}$

**for**  $\alpha \mathfrak{A} \mathfrak{B} +$

**assumes** *iso-smc-is-iso-semifunctor*:  $\exists \mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.iso\alpha} \mathfrak{B}$

**notation** *iso-semicategory* (**infixl**  $\langle \approx_{SMC} \rangle$  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*  $\alpha \mathfrak{A}$   
**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*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$  for  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} +$

**assumes** *tm-smcf-is-tm-dghm*[*slicing-intros*]:

*smcf-dghm*  $\mathfrak{F} : \text{smc-dg } \mathfrak{A} \mapsto \mapsto_{DG.tm\alpha} \text{smc-dg } \mathfrak{B}$

**syntax** *-is-tm-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - : / - \mapsto \mapsto_{SMC.tm^1} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-tm-semifunctor*  $\equiv$  *is-tm-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tm\alpha} \mathfrak{B} \equiv \text{CONST } \textit{is-tm-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-tm-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

**where** *is-cn-tm-semifunctor*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \equiv \mathfrak{F} : \text{op-dg } \mathfrak{A} \mapsto \mapsto_{SMC.tm\alpha} \mathfrak{B}$

**syntax** *-is-cn-tm-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle \langle - : / - \text{ }_{SMC.tm} \mapsto \mapsto_1 - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-cn-tm-semifunctor*  $\equiv$  *is-cn-tm-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{SMC.tm} \mapsto \mapsto_{\alpha} \mathfrak{B} \rightarrow \text{CONST } \textit{is-cn-tm-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** *all-tm-smcfs* ::  $V \Rightarrow V$

**where** *all-tm-smcfs*  $\alpha \equiv \text{set } \{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

**abbreviation** *small-tm-smcfs* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *small-tm-smcfs*  $\alpha \mathfrak{A} \mathfrak{B} \equiv \text{set } \{\mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

**lemma** (**in** *is-tm-semifunctor*) *tm-smcf-is-tm-dghm'*:

**assumes**  $\alpha' = \alpha$

**and**  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$

**and**  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$

**shows** *smcf-dghm*  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{DG.tm\alpha'} \mathfrak{B}'$

$\langle \textit{proof} \rangle$

**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**  $\mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{SMC.tm\alpha'} \mathfrak{B}'$

$\langle \textit{proof} \rangle$

**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*  $\alpha \langle \text{smc-dg } \mathfrak{A} \rangle \langle \text{smc-dg } \mathfrak{B} \rangle \langle \text{smcf-dghm } \mathfrak{F} \rangle$

$\langle \textit{proof} \rangle$

**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**  $is-tm-semifunctor \subseteq HomDom: tiny-semicategory \alpha \mathfrak{A}$   
 ⟨proof⟩

Further rules.

**lemma**  $is-tm-semifunctorI'$ :  
**assumes**  $[simp]: \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $[simp]: \mathfrak{F}(ObjMap) \in_0 Vset \alpha$   
**and**  $[simp]: \mathfrak{F}(ArrMap) \in_0 Vset \alpha$   
**and**  $[simp]: semicategory \alpha \mathfrak{B}$   
**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**  $semicategory \alpha \mathfrak{B}$   
**and**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{F}(ObjMap) \in_0 Vset \alpha$   
**and**  $\mathfrak{F}(ArrMap) \in_0 Vset \alpha$   
 ⟨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**  $semicategory \alpha \mathfrak{B}$   
**and**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{F}(ObjMap) \in_0 Vset \alpha$   
**and**  $\mathfrak{F}(ArrMap) \in_0 Vset \alpha$   
 ⟨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*  $\alpha \mathfrak{A}$

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*  $\alpha \mathfrak{C}$

and *semicategory*  $\alpha \mathfrak{D}$

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*  $\alpha \mathfrak{C}$

and *semicategory*  $\alpha \mathfrak{D}$

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*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$  for  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$  +

assumes *tiny-smcf-is-tiny-dghm*[*slicing-intros*]:

*smcf-dghm*  $\mathfrak{F} : \text{smc-dg } \mathfrak{A} \mapsto \mapsto_{DG.tiny\alpha} \text{smc-dg } \mathfrak{B}$

**syntax** *-is-tiny-semifunctor* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

( $\langle \langle - \text{ :/ } - \mapsto \mapsto_{SMC.tiny1} - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-tiny-semifunctor*  $\equiv$  *is-tiny-semifunctor*

**translations**  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B} \equiv \text{CONST } \textit{is-tiny-semifunctor } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** (*input*) *is-cn-tiny-smcf* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

where *is-cn-tiny-smcf*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \equiv \mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**syntax** *-is-cn-tiny-smcf* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

( $\langle \langle - \text{ :/ } - \text{ }_{SMC.tiny} \mapsto \mapsto_1 - \rangle \rangle$  [51, 51, 51] 51)

**syntax-consts** *-is-cn-tiny-smcf*  $\equiv$  *is-cn-tiny-smcf*

**translations**  $\mathfrak{F} : \mathfrak{A} \text{ }_{SMC.tiny} \mapsto \mapsto_{\alpha} \mathfrak{B} \rightarrow \text{CONST } \textit{is-cn-tiny-smcf } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

**abbreviation** *all-tiny-smcfs* ::  $V \Rightarrow V$

where *all-tiny-smcfs*  $\alpha \equiv \text{set } \{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

**abbreviation** *tiny-smcfs* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$

where *tiny-smcfs*  $\alpha \mathfrak{A} \mathfrak{B} \equiv \text{set } \{\mathfrak{F}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

**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*  $\subseteq$  *HomDom: tiny-semicategory*  $\alpha$   $\mathfrak{A}$

*<proof>*

**sublocale** *is-tiny-semifunctor*  $\subseteq$  *HomCod: tiny-semicategory*  $\alpha$   $\mathfrak{B}$

*<proof>*

**sublocale** *is-tiny-semifunctor*  $\subseteq$  *is-tm-semifunctor*

*<proof>*

Further rules.

**lemma** *is-tiny-semifunctorI'*:

assumes  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B}$

and *tiny-semicategory*  $\alpha$   $\mathfrak{A}$

and *tiny-semicategory*  $\alpha$   $\mathfrak{B}$

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*  $\alpha$   $\mathfrak{A}$

and *tiny-semicategory*  $\alpha$   $\mathfrak{B}$

*<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*  $\alpha$   $\mathfrak{A}$

and *tiny-semicategory*  $\alpha$   $\mathfrak{B}$

*<proof>*

**lemma** *is-tiny-semifunctor-iff*:

$\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B} \iff$

$(\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC\alpha} \mathfrak{B} \wedge \text{tiny-semicategory } \alpha \mathfrak{A} \wedge \text{tiny-semicategory } \alpha \mathfrak{B})$

*<proof>*

Size.

**lemma** (in *is-tiny-semifunctor*) *tiny-smcf-in-Vset*:  $\mathfrak{F} \in_{\circ} Vset \alpha$

*<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  $\{\mathfrak{F}. \exists \mathfrak{A} \mathfrak{B}. \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}\} \subseteq_o Vset \alpha$   
*<proof>*

**lemma** (in *is-semifunctor*) *smcf-is-tiny-semifunctor-if-ge-Limit*:  
 assumes  $Z \beta$  and  $\alpha \in_o \beta$   
 shows  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\beta} \mathfrak{B}$   
*<proof>*

### Opposite tiny semifunctor

**lemma** (in *is-tiny-semifunctor*) *is-tiny-semifunctor-op*:  
 $op-smcf \mathfrak{F} : op-smc \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} op-smc \mathfrak{B}$   
*<proof>*

**lemma** (in *is-tiny-semifunctor*) *is-tiny-semifunctor-op'[smc-op-intros]*:  
 assumes  $\mathfrak{A}' = op-smc \mathfrak{A}$  and  $\mathfrak{B}' = op-smc \mathfrak{B}$  and  $\alpha' = \alpha$   
 shows  $op-smcf \mathfrak{F} : \mathfrak{A}' \mapsto \mapsto_{SMC.tiny\alpha'} \mathfrak{B}'$   
*<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  $\mathfrak{G} : \mathfrak{B} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{C}$  and  $\mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}$   
 shows  $\mathfrak{G} \circ_{SMCF} \mathfrak{F} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{C}$   
*<proof>*

### Tiny constant semifunctor

**lemma** *smcf-const-is-tiny-semifunctor*:  
 assumes *tiny-semicategory*  $\alpha \mathfrak{C}$   
 and *tiny-semicategory*  $\alpha \mathfrak{D}$   
 and  $f : a \mapsto_{\mathfrak{D}} a$   
 and  $f \circ_{A\mathfrak{D}} f = f$   
 shows *smcf-const*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{C} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{D}$   
*<proof>*

**lemma** *smcf-const-is-tiny-semifunctor'[smc-small-cs-intros]*:  
 assumes *tiny-semicategory*  $\alpha \mathfrak{C}$   
 and *tiny-semicategory*  $\alpha \mathfrak{D}$   
 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*  $\mathfrak{C} \mathfrak{D} a f : \mathfrak{C}' \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{D}'$   
*<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*  $\mathfrak{N}$  =

[  
 $\mathfrak{N}(\text{NTMap})$ ,  
 $\text{smcf-dghm } (\mathfrak{N}(\text{NTDom}))$ ,  
 $\text{smcf-dghm } (\mathfrak{N}(\text{NTCod}))$ ,  
 $\text{smc-dg } (\mathfrak{N}(\text{NTDGDom}))$ ,  
 $\text{smc-dg } (\mathfrak{N}(\text{NTDGCod}))$   
 ]<sub>o</sub>.

Components.

**lemma** *ntsmcf-tdghm-components*:

**shows** [*slicing-simps*]: *ntsmcf-tdghm*  $\mathfrak{N}(\text{NTMap}) = \mathfrak{N}(\text{NTMap})$

**and** [*slicing-commute*]: *ntsmcf-tdghm*  $\mathfrak{N}(\text{NTDom}) = \text{smcf-dghm } (\mathfrak{N}(\text{NTDom}))$

**and** [*slicing-commute*]: *ntsmcf-tdghm*  $\mathfrak{N}(\text{NTCod}) = \text{smcf-dghm } (\mathfrak{N}(\text{NTCod}))$

**and** [*slicing-commute*]: *ntsmcf-tdghm*  $\mathfrak{N}(\text{NTDGDom}) = \text{smc-dg } (\mathfrak{N}(\text{NTDGDom}))$

**and** [*slicing-commute*]: *ntsmcf-tdghm*  $\mathfrak{N}(\text{NTDGCod}) = \text{smc-dg } (\mathfrak{N}(\text{NTDGCod}))$

*<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* =

$\mathcal{Z}$   $\alpha$  +

*vfsequence*  $\mathfrak{N}$  +

*NTDom*: *is-semifunctor*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$  +

*NTCod*: *is-semifunctor*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{G}$

**for**  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$   $\mathfrak{G}$   $\mathfrak{N}$  +

**assumes** *ntsmcf-length*[*smc-cs-simps*]: *vcard*  $\mathfrak{N} = 5_{\mathbb{N}}$

**and** *ntsmcf-is-tdghm*[*slicing-intros*]: *ntsmcf-tdghm*  $\mathfrak{N}$  :

$\text{smcf-dghm } \mathfrak{F} \mapsto_{DGHM} \text{smcf-dghm } \mathfrak{G} : \text{smc-dg } \mathfrak{A} \mapsto_{DG\alpha} \text{smc-dg } \mathfrak{B}$

**and** *ntsmcf-NTDom*[*smc-cs-simps*]:  $\mathfrak{N}(\text{NTDom}) = \mathfrak{F}$

**and** *ntsmcf-NTCod*[*smc-cs-simps*]:  $\mathfrak{N}(\text{NTCod}) = \mathfrak{G}$

**and** *ntsmcf-NTDGDom*[*smc-cs-simps*]:  $\mathfrak{N}(\text{NTDGDom}) = \mathfrak{A}$

**and** *ntsmcf-NTDGCod*[*smc-cs-simps*]:  $\mathfrak{N}(\text{NTDGCod}) = \mathfrak{B}$

**and** *ntsmcf-Comp-commute*[*smc-cs-intros*]:  $f : a \mapsto_{\mathfrak{A}} b \implies$

$\mathfrak{N}(\text{NTMap})(\downarrow b) \circ_{A\mathfrak{B}} \mathfrak{F}(\downarrow \text{ArrMap})(\downarrow f) = \mathfrak{G}(\downarrow \text{ArrMap})(\downarrow f) \circ_{A\mathfrak{B}} \mathfrak{N}(\text{NTMap})(\downarrow a)$

**syntax** *-is-ntsmcf* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle \langle - \cdot / - \mapsto_{SMCF} - \cdot / - \mapsto_{SMC^1} - \rangle [51, 51, 51, 51, 51] 51 \rangle$

**syntax-consts** *-is-ntsmcf*  $\Leftarrow$  *is-ntsmcf*

**translations**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B} \Leftarrow$

*CONST is-ntsmcf*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$   $\mathfrak{G}$   $\mathfrak{N}$

**abbreviation**  $all\text{-}ntsmcfs :: V \Rightarrow V$

**where**  $all\text{-}ntsmcfs \alpha \equiv set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B} \}$

**abbreviation**  $ntsmcfs :: V \Rightarrow V \Rightarrow V \Rightarrow V$

**where**  $ntsmcfs \alpha \mathfrak{A} \mathfrak{B} \equiv set \{ \mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B} \}$

**abbreviation**  $these\text{-}ntsmcfs :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

**where**  $these\text{-}ntsmcfs \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \equiv set \{ \mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B} \}$

**lemmas**  $[smc\text{-}cs\text{-}simps] =$

$is\text{-}ntsmcf.ntsmcf\text{-}length$

$is\text{-}ntsmcf.ntsmcf\text{-}NTDom$

$is\text{-}ntsmcf.ntsmcf\text{-}NTCod$

$is\text{-}ntsmcf.ntsmcf\text{-}NTDGDom$

$is\text{-}ntsmcf.ntsmcf\text{-}NTDGCod$

$is\text{-}ntsmcf.ntsmcf\text{-}Comp\text{-}commute$

**lemmas**  $[smc\text{-}cs\text{-}intros] = is\text{-}ntsmcf.ntsmcf\text{-}Comp\text{-}commute$

**lemma** (in  $is\text{-}ntsmcf$ )  $ntsmcf\text{-}is\text{-}tdghm'$ :

**assumes**  $\mathfrak{F}' = smcf\text{-}dghm \mathfrak{F}$

**and**  $\mathfrak{G}' = smcf\text{-}dghm \mathfrak{G}$

**and**  $\mathfrak{A}' = smc\text{-}dg \mathfrak{A}$

**and**  $\mathfrak{B}' = smc\text{-}dg \mathfrak{B}$

**shows**  $ntsmcf\text{-}tdghm \mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG\alpha} \mathfrak{B}'$

$\langle proof \rangle$

**lemmas**  $[slicing\text{-}intros] = is\text{-}ntsmcf.ntsmcf\text{-}is\text{-}tdghm'$

Rules.

**lemma** (in  $is\text{-}ntsmcf$ )  $is\text{-}ntsmcf\text{-}axioms'[smc\text{-}cs\text{-}intros]$ :

**assumes**  $\alpha' = \alpha$  **and**  $\mathfrak{A}' = \mathfrak{A}$  **and**  $\mathfrak{B}' = \mathfrak{B}$  **and**  $\mathfrak{F}' = \mathfrak{F}$  **and**  $\mathfrak{G}' = \mathfrak{G}$

**shows**  $\mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto_{SMC\alpha'} \mathfrak{B}'$

$\langle proof \rangle$

**mk-ide rf**  $is\text{-}ntsmcf\text{-}def[unfolding\ is\text{-}ntsmcf\text{-}axioms\text{-}def]$

$|intro\ is\text{-}ntsmcfI|$

$|dest\ is\text{-}ntsmcfD[dest]|$

$|elim\ is\text{-}ntsmcfE[elim]|$

**lemmas**  $[smc\text{-}cs\text{-}intros] =$

$is\text{-}ntsmcfD(3,4)$

**lemma**  $is\text{-}ntsmcfI'$ :

**assumes**  $Z \alpha$

**and**  $vfsequence \mathfrak{N}$

**and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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

**and**  $vcard \mathfrak{N} = 5_{\mathbb{N}}$

**and**  $\mathfrak{N}(NTDom) = \mathfrak{F}$

**and**  $\mathfrak{N}(NTCod) = \mathfrak{G}$

**and**  $\mathfrak{N}(NTDGDom) = \mathfrak{A}$

**and**  $\mathfrak{N}(NTDGCod) = \mathfrak{B}$

**and**  $vsv (\mathfrak{N}(NTMap))$

**and**  $\mathcal{D}_o (\mathfrak{N}(NTMap)) = \mathfrak{A}(Obj)$

**and**  $\wedge a. a \in_o \mathfrak{A}(Obj) \implies \mathfrak{N}(NTMap)(a) : \mathfrak{F}(ObjMap)(a) \mapsto_{\mathfrak{B}} \mathfrak{G}(ObjMap)(a)$

**and**  $\wedge a\ b\ f. f : a \mapsto_{\mathfrak{A}} b \implies$

$\mathfrak{N}(NTMap)(b) \circ_{A\mathfrak{B}} \mathfrak{F}(ArrMap)(f) = \mathfrak{G}(ArrMap)(f) \circ_{A\mathfrak{B}} \mathfrak{N}(NTMap)(a)$

shows  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 ⟨proof⟩

**lemma** *is-ntsmcfD'*:

assumes  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows  $Z \alpha$

and *vfsequence*  $\mathfrak{N}$

and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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

and *vcard*  $\mathfrak{N} = 5_{\mathbb{N}}$

and  $\mathfrak{N}(\downarrow NTDom) = \mathfrak{F}$

and  $\mathfrak{N}(\downarrow NTCod) = \mathfrak{G}$

and  $\mathfrak{N}(\downarrow NTDGDom) = \mathfrak{A}$

and  $\mathfrak{N}(\downarrow NTDGCod) = \mathfrak{B}$

and *vsv* ( $\mathfrak{N}(\downarrow NTMap)$ )

and  $\mathcal{D}_o$  ( $\mathfrak{N}(\downarrow NTMap)$ ) =  $\mathfrak{A}(\downarrow Obj)$

and  $\wedge a. a \in_o \mathfrak{A}(\downarrow Obj) \implies \mathfrak{N}(\downarrow NTMap)(\downarrow a) : \mathfrak{F}(\downarrow ObjMap)(\downarrow a) \mapsto_{\mathfrak{B}} \mathfrak{G}(\downarrow ObjMap)(\downarrow a)$

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

$\mathfrak{N}(\downarrow NTMap)(\downarrow b) \circ_{A\mathfrak{B}} \mathfrak{F}(\downarrow ArrMap)(\downarrow f) = \mathfrak{G}(\downarrow ArrMap)(\downarrow f) \circ_{A\mathfrak{B}} \mathfrak{N}(\downarrow NTMap)(\downarrow a)$

⟨proof⟩

**lemma** *is-ntsmcfE'*:

assumes  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

obtains  $Z \alpha$

and *vfsequence*  $\mathfrak{N}$

and  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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

and *vcard*  $\mathfrak{N} = 5_{\mathbb{N}}$

and  $\mathfrak{N}(\downarrow NTDom) = \mathfrak{F}$

and  $\mathfrak{N}(\downarrow NTCod) = \mathfrak{G}$

and  $\mathfrak{N}(\downarrow NTDGDom) = \mathfrak{A}$

and  $\mathfrak{N}(\downarrow NTDGCod) = \mathfrak{B}$

and *vsv* ( $\mathfrak{N}(\downarrow NTMap)$ )

and  $\mathcal{D}_o$  ( $\mathfrak{N}(\downarrow NTMap)$ ) =  $\mathfrak{A}(\downarrow Obj)$

and  $\wedge a. a \in_o \mathfrak{A}(\downarrow Obj) \implies \mathfrak{N}(\downarrow NTMap)(\downarrow a) : \mathfrak{F}(\downarrow ObjMap)(\downarrow a) \mapsto_{\mathfrak{B}} \mathfrak{G}(\downarrow ObjMap)(\downarrow a)$

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

$\mathfrak{N}(\downarrow NTMap)(\downarrow b) \circ_{A\mathfrak{B}} \mathfrak{F}(\downarrow ArrMap)(\downarrow f) = \mathfrak{G}(\downarrow ArrMap)(\downarrow f) \circ_{A\mathfrak{B}} \mathfrak{N}(\downarrow NTMap)(\downarrow a)$

⟨proof⟩

Slicing.

**context** *is-ntsmcf*

**begin**

**interpretation** *tdghm: is-tdghm*

$\alpha \langle smc-dg \mathfrak{A} \rangle \langle smc-dg \mathfrak{B} \rangle \langle smcf-dghm \mathfrak{F} \rangle \langle smcf-dghm \mathfrak{G} \rangle \langle ntsmcf-tdghm \mathfrak{N} \rangle$

⟨proof⟩

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

*ntsmcf-NTMap-vsuv* = *tdghm.tdghm-NTMap-vsuv*

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: vsuv*  $\langle \mathfrak{N}(\downarrow NTMap) \rangle$

rewrites  $\mathcal{D}_o$  ( $\mathfrak{N}(\downarrow NTMap)$ ) =  $\mathfrak{A}(\downarrow Obj)$

⟨proof⟩

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

$ntsmcf\text{-}NTMap\text{-}app\text{-}in\text{-}Arr[smc\text{-}cs\text{-}intros] = tdghm.tdghm\text{-}NTMap\text{-}app\text{-}in\text{-}Arr$   
**and**  $ntsmcf\text{-}NTMap\text{-}vrangle\text{-}vifunion = tdghm.tdghm\text{-}NTMap\text{-}vrangle\text{-}vifunion$   
**and**  $ntsmcf\text{-}NTMap\text{-}vrangle = tdghm.tdghm\text{-}NTMap\text{-}vrangle$   
**and**  $ntsmcf\text{-}NTMap\text{-}vsubset\text{-}Vset = tdghm.tdghm\text{-}NTMap\text{-}vsubset\text{-}Vset$   
**and**  $ntsmcf\text{-}NTMap\text{-}in\text{-}Vset = tdghm.tdghm\text{-}NTMap\text{-}in\text{-}Vset$   
**and**  $ntsmcf\text{-}is\text{-}tdghm\text{-}if\text{-}ge\text{-}Limit = tdghm.tdghm\text{-}is\text{-}tdghm\text{-}if\text{-}ge\text{-}Limit$

end

lemmas  $[smc\text{-}cs\text{-}intros] = is\text{-}ntsmcf.ntsmcf\text{-}NTMap\text{-}is\text{-}arr'$

**lemma (in  $is\text{-}ntsmcf$ )  $ntsmcf\text{-}Comp\text{-}commute'$ :**  
**assumes**  $f : a \mapsto_{\mathfrak{A}} b$  **and**  $g : c \mapsto_{\mathfrak{B}} \mathfrak{F}(ObjMap)(a)$   
**shows**  
 $\mathfrak{N}(NTMap)(b) \circ_{A\mathfrak{B}} (\mathfrak{F}(ArrMap)(f) \circ_{A\mathfrak{B}} g) =$   
 $(\mathfrak{G}(ArrMap)(f) \circ_{A\mathfrak{B}} \mathfrak{N}(NTMap)(a)) \circ_{A\mathfrak{B}} g$   
 $\langle proof \rangle$

**lemma (in  $is\text{-}ntsmcf$ )  $ntsmcf\text{-}Comp\text{-}commute''$ :**  
**assumes**  $f : a \mapsto_{\mathfrak{A}} b$  **and**  $g : c \mapsto_{\mathfrak{B}} \mathfrak{F}(ObjMap)(a)$   
**shows**  
 $\mathfrak{G}(ArrMap)(f) \circ_{A\mathfrak{B}} (\mathfrak{N}(NTMap)(a) \circ_{A\mathfrak{B}} g) =$   
 $(\mathfrak{N}(NTMap)(b) \circ_{A\mathfrak{B}} \mathfrak{F}(ArrMap)(f)) \circ_{A\mathfrak{B}} g$   
 $\langle proof \rangle$

Elementary properties.

**lemma  $ntsmcf\text{-}eqI$ :**  
**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$   
**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**  $\mathfrak{N} = \mathfrak{N}'$   
 $\langle proof \rangle$

**lemma  $ntsmcf\text{-}tdghm\text{-}eqI$ :**  
**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$   
**and**  $\mathfrak{F} = \mathfrak{F}'$   
**and**  $\mathfrak{G} = \mathfrak{G}'$   
**and**  $\mathfrak{A} = \mathfrak{A}'$   
**and**  $\mathfrak{B} = \mathfrak{B}'$   
**and**  $ntsmcf\text{-}tdghm \mathfrak{N} = ntsmcf\text{-}tdghm \mathfrak{N}'$   
**shows**  $\mathfrak{N} = \mathfrak{N}'$   
 $\langle proof \rangle$

**lemma (in  $is\text{-}ntsmcf$ )  $ntsmcf\text{-}def$ :**  
 $\mathfrak{N} = [\mathfrak{N}(NTMap), \mathfrak{N}(NTDom), \mathfrak{N}(NTCod), \mathfrak{N}(NTDGDom), \mathfrak{N}(NTDGCod)]$ .  
 $\langle proof \rangle$

Size.

**lemma (in  $is\text{-}ntsmcf$ )  $ntsmcf\text{-}in\text{-}Vset$ :**  
**assumes**  $Z \beta$  **and**  $\alpha \in_{\circ} \beta$   
**shows**  $\mathfrak{N} \in_{\circ} Vset \beta$   
 $\langle proof \rangle$

**lemma** (in *is-ntsmcf*) *ntsmcf-is-ntsmcf-if-ge-Limit*:

assumes  $Z \beta$  and  $\alpha \in_{\circ} \beta$

shows  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMCF} \mathfrak{B}$

*<proof>*

**lemma** *small-all-ntsmcfs[simp]*:

*small*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMCF} \mathfrak{B}\}$

*<proof>*

**lemma** *small-ntsmcfs[simp]*: *small*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMCF} \mathfrak{B}\}$

*<proof>*

**lemma** *small-these-ntcfs[simp]*: *small*  $\{\mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMCF} \mathfrak{B}\}$

*<proof>*

Further elementary results.

**lemma** *these-ntsmcfs-iff*:

$\mathfrak{N} \in_{\circ} \text{these-ntsmcfs } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \iff \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMCF} \mathfrak{B}$

*<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*  $\mathfrak{N} =$

[  
 $\mathfrak{N}(\text{NTMap})$ ,  
 $\text{op-smcf } (\mathfrak{N}(\text{NTCod}))$ ,  
 $\text{op-smcf } (\mathfrak{N}(\text{NTDom}))$ ,  
 $\text{op-smc } (\mathfrak{N}(\text{NTDGDom}))$ ,  
 $\text{op-smc } (\mathfrak{N}(\text{NTDGCod}))$   
 ]<sub>o</sub>

Components.

**lemma** *op-ntsmcf-components[smc-op-simps]*:

shows *op-ntsmcf*  $\mathfrak{N}(\text{NTMap}) = \mathfrak{N}(\text{NTMap})$

and *op-ntsmcf*  $\mathfrak{N}(\text{NTDom}) = \text{op-smcf } (\mathfrak{N}(\text{NTCod}))$

and *op-ntsmcf*  $\mathfrak{N}(\text{NTCod}) = \text{op-smcf } (\mathfrak{N}(\text{NTDom}))$

and *op-ntsmcf*  $\mathfrak{N}(\text{NTDGDom}) = \text{op-smc } (\mathfrak{N}(\text{NTDGDom}))$

and *op-ntsmcf*  $\mathfrak{N}(\text{NTDGCod}) = \text{op-smc } (\mathfrak{N}(\text{NTDGCod}))$

*<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*  $\mathfrak{N} : \text{op-smcf } \mathfrak{G} \mapsto_{SMCF} \text{op-smcf } \mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto_{SMCF} \text{op-smc } \mathfrak{B}$

*<proof>*

**lemma** (in *is-ntsmcf*) *is-ntsmcf-op'[smc-op-intros]*:

assumes  $\mathfrak{G}' = \text{op-smcf } \mathfrak{G}$

**and**  $\mathfrak{F}' = \text{op-smcf } \mathfrak{F}$   
**and**  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$   
**and**  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$   
**shows**  $\text{op-ntsmcf } \mathfrak{N} : \mathfrak{G}' \mapsto_{SMCF} \mathfrak{F}' : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $[\text{smc-op-intros}] = \text{is-ntsmcf.is-ntsmcf-op}'$

**lemma** (in  $\text{is-ntsmcf}$ )  $\text{ntsmcf-op-ntsmcf-op-ntsmcf}[\text{smc-op-simps}]$ :  
 $\text{op-ntsmcf } (\text{op-ntsmcf } \mathfrak{N}) = \mathfrak{N}$   
 ⟨proof⟩

**lemmas**  $\text{ntsmcf-op-ntsmcf-op-ntsmcf}[\text{smc-op-simps}] =$   
 $\text{is-ntsmcf.ntsmcf-op-ntsmcf-op-ntsmcf}$

**lemma**  $\text{eq-op-ntsmcf-iff}$ :  
**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$   
**shows**  $\text{op-ntsmcf } \mathfrak{N} = \text{op-ntsmcf } \mathfrak{N}' \leftrightarrow \mathfrak{N} = \mathfrak{N}'$   
 ⟨proof⟩

#### 4.6.4 Vertical composition of natural transformations

##### Definition and elementary properties

See Chapter II-4 in [39].

**definition**  $\text{ntsmcf-vcomp} :: V \Rightarrow V \Rightarrow V$  (infixl  $\langle \cdot_{NTSMCF} \rangle$  55)

**where**  $\text{ntsmcf-vcomp } \mathfrak{M} \mathfrak{N} =$

$[$   
 $(\lambda a \in \circ \mathfrak{N}(\text{NTDGDom})(\text{Obj}). (\mathfrak{M}(\text{NTMap})(a)) \circ_{A\mathfrak{N}(\text{NTDGCod})} (\mathfrak{N}(\text{NTMap})(a))),$   
 $\mathfrak{N}(\text{NTDom}),$   
 $\mathfrak{N}(\text{NTCod}),$   
 $\mathfrak{N}(\text{NTDGDom}),$   
 $\mathfrak{N}(\text{NTDGCod})$   
 $]$ .

Components.

**lemma**  $\text{ntsmcf-vcomp-components}$ :

**shows**

$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTMap}) =$   
 $(\lambda a \in \circ \mathfrak{N}(\text{NTDGDom})(\text{Obj}). (\mathfrak{M}(\text{NTMap})(a)) \circ_{A\mathfrak{N}(\text{NTDGCod})} (\mathfrak{N}(\text{NTMap})(a)))$

**and**  $[\text{dg-shared-cs-simps}, \text{smc-cs-simps}] : (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTDom}) = \mathfrak{N}(\text{NTDom})$

**and**  $[\text{dg-shared-cs-simps}, \text{smc-cs-simps}] : (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTCod}) = \mathfrak{N}(\text{NTCod})$

**and**  $[\text{dg-shared-cs-simps}, \text{smc-cs-simps}] :$

$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTDGDom}) = \mathfrak{N}(\text{NTDGDom})$

**and**  $[\text{dg-shared-cs-simps}, \text{smc-cs-simps}] :$

$(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTDGCod}) = \mathfrak{N}(\text{NTDGCod})$

⟨proof⟩

##### Natural transformation map

**lemma**  $\text{ntsmcf-vcomp-NTMap-vs}[\text{dg-shared-cs-intros}, \text{smc-cs-intros}]$ :

$\text{vs} ((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(\text{NTMap}))$

⟨proof⟩

**lemma**  $\text{ntsmcf-vcomp-NTMap-vdomain}[\text{smc-cs-simps}]$ :

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows  $\mathcal{D}_\circ ((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(NTMap)) = \mathfrak{A}(\mathcal{O}bj)$   
 ⟨proof⟩

**lemma** *ntsmcf-vcomp-NTMap-app*[*smc-cs-simps*]:

assumes  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $a \in_\circ \mathfrak{A}(\mathcal{O}bj)$

shows  $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(NTMap)(a) = \mathfrak{M}(NTMap)(a) \circ_{A\mathfrak{B}} \mathfrak{N}(NTMap)(a)$

⟨proof⟩

**lemma** *ntsmcf-vcomp-NTMap-vrange*:

assumes  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows  $\mathcal{R}_\circ ((\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N})(NTMap)) \subseteq_\circ \mathfrak{B}(\mathcal{A}rr)$

⟨proof⟩

### Further properties

**lemma** *ntsmcf-vcomp-composable-commute*[*smc-cs-simps*]:

— See Chapter II-4 in [39].

assumes  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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

shows

$(\mathfrak{M}(NTMap)(b) \circ_{A\mathfrak{B}} \mathfrak{N}(NTMap)(b)) \circ_{A\mathfrak{B}} \mathfrak{F}(\mathcal{A}rrMap)(f) =$

$\mathfrak{H}(\mathcal{A}rrMap)(f) \circ_{A\mathfrak{B}} (\mathfrak{M}(NTMap)(a) \circ_{A\mathfrak{B}} \mathfrak{N}(NTMap)(a))$

(is  $\langle (?MC \circ_{A\mathfrak{B}} ?NC) \circ_{A\mathfrak{B}} ?R = ?T \circ_{A\mathfrak{B}} (?MD \circ_{A\mathfrak{B}} ?ND) \rangle$ )

⟨proof⟩

**lemma** *ntsmcf-vcomp-is-ntsmcf*[*smc-cs-intros*]:

— See Chapter II-4 in [39].

assumes  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows  $\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

⟨proof⟩

**lemma** *ntsmcf-vcomp-assoc*[*smc-cs-simps*]:

— See Chapter II-4 in [39].

assumes  $\mathfrak{L} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{K} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows *op-ntsmcf*  $(\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N}) = \textit{op-ntsmcf} \mathfrak{N} \cdot_{NTSMCF} \textit{op-ntsmcf} \mathfrak{M}$

⟨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  $\langle \circ_{NTSMCF} \rangle$  55)

where  $ntsmcf\text{-}hcomp \mathfrak{M} \mathfrak{N} =$

$$\begin{aligned} & [ \\ & \quad ( \\ & \quad \quad \lambda a \in_{\circ} \mathfrak{N}(\downarrow NTDGDom)(\downarrow Obj). \\ & \quad \quad ( \\ & \quad \quad \quad \mathfrak{M}(\downarrow NTCod)(\downarrow ArrMap)(\downarrow \mathfrak{N}(\downarrow NTMap)(\downarrow a)) \circ_A \mathfrak{M}(\downarrow NTDGCod) \\ & \quad \quad \quad \mathfrak{M}(\downarrow NTMap)(\downarrow \mathfrak{N}(\downarrow NTDom)(\downarrow ObjMap)(\downarrow a)) \\ & \quad \quad ) \\ & \quad ), \\ & \quad (\mathfrak{M}(\downarrow NTDom) \circ_{SMCF} \mathfrak{N}(\downarrow NTDom)), \\ & \quad (\mathfrak{M}(\downarrow NTCod) \circ_{SMCF} \mathfrak{N}(\downarrow NTCod)), \\ & \quad (\mathfrak{N}(\downarrow NTDGDom)), \\ & \quad (\mathfrak{M}(\downarrow NTDGCod)) \\ & ]_{\circ} \end{aligned}$$

Components.

**lemma**  $ntsmcf\text{-}hcomp\text{-}components$ :

**shows**

$$\begin{aligned} & (\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap) = \\ & \quad ( \\ & \quad \quad \lambda a \in_{\circ} \mathfrak{N}(\downarrow NTDGDom)(\downarrow Obj). \\ & \quad \quad ( \\ & \quad \quad \quad \mathfrak{M}(\downarrow NTCod)(\downarrow ArrMap)(\downarrow \mathfrak{N}(\downarrow NTMap)(\downarrow a)) \circ_A \mathfrak{M}(\downarrow NTDGCod) \\ & \quad \quad \quad \mathfrak{M}(\downarrow NTMap)(\downarrow \mathfrak{N}(\downarrow NTDom)(\downarrow ObjMap)(\downarrow a)) \\ & \quad \quad ) \\ & \quad ) \end{aligned}$$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, smc\text{-}cs\text{-}simps]$ :

$$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTDom) = \mathfrak{M}(\downarrow NTDom) \circ_{SMCF} \mathfrak{N}(\downarrow NTDom)$$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, smc\text{-}cs\text{-}simps]$ :

$$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTCod) = \mathfrak{M}(\downarrow NTCod) \circ_{SMCF} \mathfrak{N}(\downarrow NTCod)$$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, smc\text{-}cs\text{-}simps]$ :

$$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTDGDom) = \mathfrak{N}(\downarrow NTDGDom)$$

**and**  $[dg\text{-}shared\text{-}cs\text{-}simps, smc\text{-}cs\text{-}simps]$ :

$$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTDGCod) = \mathfrak{M}(\downarrow NTDGCod)$$

$\langle proof \rangle$

## Natural transformation map

**lemma**  $ntsmcf\text{-}hcomp\text{-}NTMap\text{-}vsu[smc\text{-}cs\text{-}intros]$ :  $vsu ((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap))$

$\langle proof \rangle$

**lemma**  $ntsmcf\text{-}hcomp\text{-}NTMap\text{-}vdomain[smc\text{-}cs\text{-}simps]$ :

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**shows**  $\mathcal{D}_{\circ} ((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap)) = \mathfrak{A}(\downarrow Obj)$

$\langle proof \rangle$

**lemma**  $ntsmcf\text{-}hcomp\text{-}NTMap\text{-}app[smc\text{-}cs\text{-}simps]$ :

**assumes**  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**and**  $a \in_{\circ} \mathfrak{A}(\downarrow Obj)$

**shows**  $(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap)(\downarrow a) =$

$$\mathfrak{G}'(\downarrow ArrMap)(\downarrow \mathfrak{N}(\downarrow NTMap)(\downarrow a)) \circ_{A\mathfrak{C}} \mathfrak{M}(\downarrow NTMap)(\downarrow \mathfrak{F}(\downarrow ObjMap)(\downarrow a))$$

$\langle proof \rangle$

**lemma**  $ntsmcf\text{-}hcomp\text{-}NTMap\text{-}vrangle$ :

**assumes**  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**shows**  $\mathcal{R}_{\circ} ((\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap)) \subseteq_{\circ} \mathfrak{C}(\downarrow Arr)$

*<proof>*

### Further properties

**lemma** *ntsmcf-hcomp-composable-commute*:

— See Chapter II-5 in [39].

assumes  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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

shows

$$(\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap)(\downarrow b) \circ_{A\mathfrak{C}} (\mathfrak{F}' \circ_{SMCF} \mathfrak{F})(\downarrow ArrMap)(\downarrow f) =$$

$$(\mathfrak{G}' \circ_{SMCF} \mathfrak{G})(\downarrow ArrMap)(\downarrow f) \circ_{A\mathfrak{C}} (\mathfrak{M} \circ_{NTSMCF} \mathfrak{N})(\downarrow NTMap)(\downarrow a)$$

*<proof>*

**lemma** *ntsmcf-hcomp-is-ntsmcf*:

— See Chapter II-5 in [39].

assumes  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

shows  $\mathfrak{M} \circ_{NTSMCF} \mathfrak{N} : \mathfrak{F}' \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF} \mathfrak{G}' \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$

*<proof>*

**lemma** *ntsmcf-hcomp-is-ntsmcf'[smc-cs-intros]*:

assumes  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{S} = \mathfrak{F}' \circ_{SMCF} \mathfrak{F}$

and  $\mathfrak{S}' = \mathfrak{G}' \circ_{SMCF} \mathfrak{G}$

shows  $\mathfrak{M} \circ_{NTSMCF} \mathfrak{N} : \mathfrak{S} \mapsto_{SMCF} \mathfrak{S}' : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$

*<proof>*

**lemma** *ntsmcf-hcomp-assoc[smc-cs-simps]*:

— See Chapter II-5 in [39].

assumes  $\mathfrak{L} : \mathfrak{F}'' \mapsto_{SMCF} \mathfrak{G}'' : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$

and  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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  $\mathfrak{M} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

and  $\mathfrak{M}' : \mathfrak{G}' \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

and  $\mathfrak{N}' : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

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**  $\langle \circ_{NTSMCF-SMCF} \rangle$  55)  
**where** *ntsmcf-smcf-comp*  $\equiv$  *tdghm-dghm-comp*

Slicing.

**lemma** *ntsmcf-tdghm-ntsmcf-smcf-comp*[*slicing-commute*]:  
 $ntsmcf-tdghm \mathfrak{N} \circ_{TDGHM-DGHM} smcf-dghm \mathfrak{H} = ntsmcf-tdghm (\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H})$   
*<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**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  **and**  $\mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**shows**  $\mathcal{R}_\circ ((\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H})(NTMap)) \subseteq_\circ \mathcal{C}(Arr)$   
*<proof>*

#### Opposite of the composition of a natural transformation of semifunctors and a semifunctor

**lemma** *op-ntsmcf-ntsmcf-smcf-comp*[*smc-op-simps*]:  
 $op-ntsmcf (\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H}) = op-ntsmcf \mathfrak{N} \circ_{NTSMCF-SMCF} op-smcf \mathfrak{H}$   
*<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**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  **and**  $\mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H} : \mathfrak{F} \circ_{SMCF} \mathfrak{H} \mapsto_{SMCF} \mathfrak{G} \circ_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$   
*<proof>*

**lemma** *ntsmcf-smcf-comp-is-semifunctor*'[*smc-cs-intros*]:  
**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
**and**  $\mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**and**  $\mathfrak{F}' = \mathfrak{F} \circ_{SMCF} \mathfrak{H}$   
**and**  $\mathfrak{G}' = \mathfrak{G} \circ_{SMCF} \mathfrak{H}$   
**shows**  $\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$   
*<proof>*

#### Further properties

**lemma** *ntsmcf-smcf-comp-ntsmcf-smcf-comp-assoc*:  
**assumes**  $\mathfrak{N} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$   
**and**  $\mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
**and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**shows**  $(\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{G}) \circ_{NTSMCF-SMCF} \mathfrak{F} = \mathfrak{N} \circ_{NTSMCF-SMCF} (\mathfrak{G} \circ_{SMCF} \mathfrak{F})$

*<proof>*

**lemma** (in *is-ntsmcf*) *ntsmcf-ntsmcf-smcf-comp-smcf-id*[*smc-cs-simps*]:

$$\mathfrak{N} \circ_{NTSMCF-SMCF} \text{smcf-id } \mathfrak{A} = \mathfrak{N}$$

*<proof>*

**lemmas** [*smc-cs-simps*] = *is-ntsmcf.ntsmcf-ntsmcf-smcf-comp-smcf-id*

**lemma** *ntsmcf-vcomp-ntsmcf-smcf-comp*[*smc-cs-simps*]:

**assumes**  $\mathfrak{R} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**and**  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$

**shows**

$$\begin{aligned} & (\mathfrak{M} \circ_{NTSMCF-SMCF} \mathfrak{R}) \cdot_{NTSMCF} (\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{R}) = \\ & (\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N}) \circ_{NTSMCF-SMCF} \mathfrak{R} \end{aligned}$$

*<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*  $\equiv$  *dghm-tdghm-comp*

Slicing.

**lemma** *ntsmcf-tdghm-smcf-ntsmcf-comp*[*slicing-commute*]:

$$\text{smcf-dghm } \mathfrak{H} \circ_{DGHM-TDGHM} \text{ntsmcf-tdghm } \mathfrak{N} = \text{ntsmcf-tdghm } (\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N})$$

*<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**  $\mathfrak{H} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  **and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**shows**  $\mathcal{R}_\circ ((\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N})(\text{NTMap})) \subseteq_\circ \mathfrak{C}(\text{Arr})$

*<proof>*

#### Opposite of the composition of a semifunctor and a natural transformation of semifunctors

**lemma** *op-ntsmcf-smcf-ntsmcf-comp*[*smc-op-simps*]:

$$\text{op-ntsmcf } (\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N}) = \text{op-smcf } \mathfrak{H} \circ_{SMCF-NTSMCF} \text{op-ntsmcf } \mathfrak{N}$$

*<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**  $\mathfrak{H} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$  **and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**shows**  $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{H} \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF} \mathfrak{H} \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$   
 ⟨proof⟩

**lemma** *smcf-ntsmcf-comp-is-semifunctor'*[*smc-cs-intros*]:

**assumes**  $\mathfrak{H} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and  $\mathfrak{F}' = \mathfrak{H} \circ_{SMCF} \mathfrak{F}$   
 and  $\mathfrak{G}' = \mathfrak{H} \circ_{SMCF} \mathfrak{G}$   
**shows**  $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{C}$   
 ⟨proof⟩

### Further properties

**lemma** *smcf-comp-smcf-ntsmcf-comp-assoc*:

**assumes**  $\mathfrak{N} : \mathfrak{H} \mapsto_{SMCF} \mathfrak{H}' : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and  $\mathfrak{F} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{G} : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$   
**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**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{H} : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{D}$   
 and  $\mathfrak{K} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**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**  $\mathfrak{K} : \mathfrak{B} \mapsto_{SMC\alpha} \mathfrak{C}$   
 and  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
 and  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$   
**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**  $is\text{-}tm\text{-}ntsmcf = is\text{-}ntsmcf \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F} \ \mathfrak{G} \ \mathfrak{N}$  for  $\alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F} \ \mathfrak{G} \ \mathfrak{N} +$

**assumes**  $tm\text{-}ntsmcf\text{-}is\text{-}tm\text{-}tdghm[slicing\text{-}intros]: ntsmcf\text{-}tdghm \ \mathfrak{N} :$

$smcf\text{-}dghm \ \mathfrak{F} \mapsto_{DGHM.tm} smcf\text{-}dghm \ \mathfrak{G} : smc\text{-}dg \ \mathfrak{A} \mapsto\mapsto_{DG.tm\alpha} smc\text{-}dg \ \mathfrak{B}$

**syntax**  $is\text{-}tm\text{-}ntsmcf :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow bool$

$\langle\langle(- : / - \mapsto_{SMCF.tm} - : / - \mapsto\mapsto_{SMC.tm1} -)\rangle [51, 51, 51, 51, 51] \ 51\rangle$

**syntax-consts**  $is\text{-}tm\text{-}ntsmcf \equiv is\text{-}tm\text{-}ntsmcf$

**translations**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B} \equiv$

$CONST \ is\text{-}tm\text{-}ntsmcf \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F} \ \mathfrak{G} \ \mathfrak{N}$

**abbreviation**  $all\text{-}tm\text{-}ntsmcfs :: V \Rightarrow V$

**where**  $all\text{-}tm\text{-}ntsmcfs \ \alpha \equiv$

$set \ \{\mathfrak{N}. \exists \mathfrak{F} \ \mathfrak{G} \ \mathfrak{A} \ \mathfrak{B}. \ \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

**abbreviation**  $tm\text{-}ntsmcfs :: V \Rightarrow V \Rightarrow V \Rightarrow V$

**where**  $tm\text{-}ntsmcfs \ \alpha \ \mathfrak{A} \ \mathfrak{B} \equiv$

$set \ \{\mathfrak{N}. \exists \mathfrak{F} \ \mathfrak{G}. \ \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

**abbreviation**  $these\text{-}tm\text{-}ntsmcfs :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

**where**  $these\text{-}tm\text{-}ntsmcfs \ \alpha \ \mathfrak{A} \ \mathfrak{B} \ \mathfrak{F} \ \mathfrak{G} \equiv$

$set \ \{\mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

**lemma (in  $is\text{-}tm\text{-}ntsmcf$ )**  $tm\text{-}ntsmcf\text{-}is\text{-}tm\text{-}tdghm'$ :

**assumes**  $\alpha' = \alpha$

**and**  $\mathfrak{F}' = smcf\text{-}dghm \ \mathfrak{F}$

**and**  $\mathfrak{G}' = smcf\text{-}dghm \ \mathfrak{G}$

**and**  $\mathfrak{A}' = smc\text{-}dg \ \mathfrak{A}$

**and**  $\mathfrak{B}' = smc\text{-}dg \ \mathfrak{B}$

**shows**  $ntsmcf\text{-}tdghm \ \mathfrak{N} :$

$\mathfrak{F}' \mapsto_{DGHM.tm} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{DG.tm\alpha'} \mathfrak{B}'$

$\langle proof \rangle$

**lemmas**  $[slicing\text{-}intros] = is\text{-}tm\text{-}ntsmcf.tm\text{-}ntsmcf\text{-}is\text{-}tm\text{-}tdghm'$

Rules.

**lemma (in  $is\text{-}tm\text{-}ntsmcf$ )**  $is\text{-}tm\text{-}ntsmcf\text{-}axioms'[smc\text{-}small\text{-}cs\text{-}intros]:$

**assumes**  $\alpha' = \alpha$  **and**  $\mathfrak{A}' = \mathfrak{A}$  **and**  $\mathfrak{B}' = \mathfrak{B}$  **and**  $\mathfrak{F}' = \mathfrak{F}$  **and**  $\mathfrak{G}' = \mathfrak{G}$

**shows**  $\mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF.tm} \mathfrak{G}' : \mathfrak{A}' \mapsto\mapsto_{SMC.tm\alpha} \mathfrak{B}'$

$\langle proof \rangle$

**mk-ide rf**  $is\text{-}tm\text{-}ntsmcf\text{-}def[unfolding \ is\text{-}tm\text{-}ntsmcf\text{-}axioms\text{-}def]$

$|intro \ is\text{-}tm\text{-}ntsmcfI|$

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

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

**lemmas**  $[smc\text{-}small\text{-}cs\text{-}intros] = is\text{-}tm\text{-}ntsmcfD(1)$

Slicing.

**context**  $is\text{-}tm\text{-}ntsmcf$

**begin**

**interpretation**  $tdghm: is\text{-}tm\text{-}tdghm$

$\alpha \ \langle smc\text{-}dg \ \mathfrak{A} \rangle \ \langle smc\text{-}dg \ \mathfrak{B} \rangle \ \langle smcf\text{-}dghm \ \mathfrak{F} \rangle \ \langle smcf\text{-}dghm \ \mathfrak{G} \rangle \ \langle ntsmcf\text{-}tdghm \ \mathfrak{N} \rangle$

*<proof>*

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

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

**end**

Elementary properties.

**sublocale** *is-tm-ntsmcf*  $\subseteq$  *NTDom*: *is-tm-semifunctor*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{F}$

*<proof>*

**sublocale** *is-tm-ntsmcf*  $\subseteq$  *NTCod*: *is-tm-semifunctor*  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{G}$

*<proof>*

Further rules.

**lemma** *is-tm-ntsmcfI'*:

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

**and**  $\mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

**shows**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

*<proof>*

**lemma** *is-tm-ntsmcfD'*:

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

**shows**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

**and**  $\mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

*<proof>*

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

Size.

**lemma** *small-all-tm-ntsmcfs[simp]*:

*small*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

*<proof>*

**lemma** *small-tm-ntsmcfs[simp]*:

*small*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

*<proof>*

**lemma** *small-these-tm-ntsmcfs[simp]*:

*small*  $\{\mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}\}$

*<proof>*

Further elementary results.

**lemma** *these-tm-ntsmcfs-iff*:

$\mathfrak{N} \in_{\circ} \text{these-tm-ntsmcfs } \alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \longleftrightarrow$

$\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$

*<proof>*

**Opposite natural transformation of semifunctors with tiny maps**

**lemma** (**in** *is-tm-ntsmcf*) *is-tm-ntsmcf-op*: *op-ntsmcf*  $\mathfrak{N} :$

*op-smcf*  $\mathfrak{G} \mapsto_{SMCF.tm} \text{op-smcf } \mathfrak{F} : \text{op-smc } \mathfrak{A} \mapsto_{SMC.tm\alpha} \text{op-smc } \mathfrak{B}$

*<proof>*

**lemma** (**in** *is-tm-ntsmcf*) *is-tm-ntsmcf-op'*[*smc-op-intros*]:

**assumes**  $\mathfrak{G}' = \text{op-smcf } \mathfrak{G}$   
**and**  $\mathfrak{F}' = \text{op-smcf } \mathfrak{F}$   
**and**  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$   
**and**  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$   
**shows**  $\text{op-ntsmcf } \mathfrak{N} : \mathfrak{G}' \mapsto_{SMCF.tm} \mathfrak{F}' : \mathfrak{A}' \mapsto_{SMC.tm\alpha} \mathfrak{B}'$   
 ⟨proof⟩

**lemmas**  $\text{is-tm-ntsmcf-op}[smc-op-intros] = \text{is-tm-ntsmcf.is-tm-ntsmcf-op}'$

### Vertical composition of natural transformations of semifunctors with tiny maps

**lemma**  $\text{ntsmcf-vcomp-is-tm-ntsmcf}[smc-small-cs-intros]$ :

**assumes**  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF.tm} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
 ⟨proof⟩

### Composition of a natural transformation of semifunctors and a semifunctor

**lemma**  $\text{ntsmcf-smcf-comp-is-tm-ntsmcf}$ :

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC.tm\alpha} \mathfrak{C}$  **and**  $\mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H} : \mathfrak{F} \circ_{SMCF} \mathfrak{H} \mapsto_{SMCF.tm} \mathfrak{G} \circ_{SMCF} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
 ⟨proof⟩

**lemma**  $\text{ntsmcf-smcf-comp-is-tm-ntsmcf}'[smc-small-cs-intros]$ :

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{B} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
**and**  $\mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**and**  $\mathfrak{F}' = \mathfrak{F} \circ_{SMCF} \mathfrak{H}$   
**and**  $\mathfrak{G}' = \mathfrak{G} \circ_{SMCF} \mathfrak{H}$   
**shows**  $\mathfrak{N} \circ_{NTSMCF-SMCF} \mathfrak{H} : \mathfrak{F}' \mapsto_{SMCF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
 ⟨proof⟩

### Composition of a semifunctor and a natural transformation of semifunctors

**lemma**  $\text{smcf-ntsmcf-comp-is-tm-ntsmcf}$ :

**assumes**  $\mathfrak{H} : \mathfrak{B} \mapsto_{SMC.tm\alpha} \mathfrak{C}$  **and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**shows**  $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{H} \circ_{SMCF} \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{H} \circ_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
 ⟨proof⟩

**lemma**  $\text{smcf-ntsmcf-comp-is-tm-ntsmcf}'[smc-small-cs-intros]$ :

**assumes**  $\mathfrak{H} : \mathfrak{B} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tm} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{B}$   
**and**  $\mathfrak{F}' = \mathfrak{H} \circ_{SMCF} \mathfrak{F}$   
**and**  $\mathfrak{G}' = \mathfrak{H} \circ_{SMCF} \mathfrak{G}$   
**shows**  $\mathfrak{H} \circ_{SMCF-NTSMCF} \mathfrak{N} : \mathfrak{F}' \mapsto_{SMCF.tm} \mathfrak{G}' : \mathfrak{A} \mapsto_{SMC.tm\alpha} \mathfrak{C}$   
 ⟨proof⟩

## 4.7.2 Tiny natural transformation of semifunctors

### Definition and elementary properties

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

**assumes**  $\text{tiny-ntsmcf-is-tdghm}[slicing-intros]$ :  $\text{ntsmcf-tdghm } \mathfrak{N} :$

$\text{smcf-dghm } \mathfrak{F} \mapsto_{DGHM.tiny} \text{smcf-dghm } \mathfrak{G} : \text{smc-dg } \mathfrak{A} \mapsto_{DG.tiny\alpha} \text{smc-dg } \mathfrak{B}$

**syntax**  $\text{-is-tiny-ntsmcf} :: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow \text{bool}$

$\langle\langle - \cdot / - \mapsto_{SMCF.tiny} - \cdot / - \mapsto_{SMC.tiny1} - \rangle\rangle [51, 51, 51, 51, 51] 51)$

**syntax-consts**  $\text{-is-tiny-ntsmcf} \equiv \text{is-tiny-ntsmcf}$

**translations**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B} \equiv$

*CONST is-tiny-ntsmcf*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \mathfrak{N}$

**abbreviation** *all-tiny-ntsmcfs*  $:: V \Rightarrow V$

**where** *all-tiny-ntsmcfs*  $\alpha \equiv$

*set*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G} \mathfrak{A} \mathfrak{B}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

**abbreviation** *tiny-ntsmcfs*  $:: V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *tiny-ntsmcfs*  $\alpha \mathfrak{A} \mathfrak{B} \equiv$

*set*  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

**abbreviation** *these-tiny-ntsmcfs*  $:: V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V \Rightarrow V$

**where** *these-tiny-ntsmcfs*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F} \mathfrak{G} \equiv$

*set*  $\{\mathfrak{N}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}\}$

**lemma** (in *is-tiny-ntsmcf*) *tiny-ntsmcf-is-tdghm'*:

**assumes**  $\alpha' = \alpha$

**and**  $\mathfrak{F}' = \text{smcf-dghm } \mathfrak{F}$

**and**  $\mathfrak{G}' = \text{smcf-dghm } \mathfrak{G}$

**and**  $\mathfrak{A}' = \text{smc-dg } \mathfrak{A}$

**and**  $\mathfrak{B}' = \text{smc-dg } \mathfrak{B}$

**shows** *ntsmcf-tdghm*  $\mathfrak{N} : \mathfrak{F}' \mapsto_{DGHM.tiny} \mathfrak{G}' : \mathfrak{A}' \mapsto_{DG.tiny\alpha'} \mathfrak{B}'$

*<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**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

*<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*  $\subseteq$  *NTDom: is-tiny-semifunctor*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{F}$

*<proof>*

**sublocale** *is-tiny-ntsmcf*  $\subseteq$  *NTCod: is-tiny-semifunctor*  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{G}$

*<proof>*

**sublocale** *is-tiny-ntsmcf*  $\subseteq$  *is-tm-ntsmcf*

*<proof>*

Further rules.

**lemma** *is-tiny-ntsmcfI'*:

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

**and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**and**  $\mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**shows**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

*<proof>*

**lemma** *is-tiny-ntsmcfD'*:

**assumes**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**shows**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC\alpha} \mathfrak{B}$

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:  $\mathfrak{N} \in_0 Vset\ \alpha$   
 ⟨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-uset-Vset[simp]:

set  $\{\mathfrak{N}. \exists \mathfrak{F} \mathfrak{G}. \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}\} \subseteq_0 Vset\ \alpha$   
 (is <set ?ntsmcfs  $\subseteq_0$  ->)  
 ⟨proof⟩

lemma (in is-ntsmcf) ntsmcf-is-tiny-ntsmcf-if-ge-Limit:

assumes  $Z\ \beta$  and  $\alpha \in_0 \beta$   
 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:

$\mathfrak{N} \in_0 these-tiny-ntsmcfs\ \alpha\ \mathfrak{A}\ \mathfrak{B}\ \mathfrak{F}\ \mathfrak{G} \longleftrightarrow$   
 $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto \mapsto_{SMC.tiny\alpha} \mathfrak{B}$   
 ⟨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**  $\mathfrak{G}' = \text{op-smcf } \mathfrak{G}$

**and**  $\mathfrak{F}' = \text{op-smcf } \mathfrak{F}$

**and**  $\mathfrak{A}' = \text{op-smc } \mathfrak{A}$

**and**  $\mathfrak{B}' = \text{op-smc } \mathfrak{B}$

**shows**  $\text{op-ntsmcf } \mathfrak{N} : \mathfrak{G}' \mapsto_{SMCF.tiny} \mathfrak{F}' : \mathfrak{A}' \mapsto_{SMC.tiny\alpha} \mathfrak{B}'$

*<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**  $\mathfrak{M} : \mathfrak{G} \mapsto_{SMCF.tiny} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**and**  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{G} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

**shows**  $\mathfrak{M} \cdot_{NTSMCF} \mathfrak{N} : \mathfrak{F} \mapsto_{SMCF.tiny} \mathfrak{H} : \mathfrak{A} \mapsto_{SMC.tiny\alpha} \mathfrak{B}$

*<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*  $I \mathfrak{A} =$

[  
 $(\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i (Obj))$ ,  
 $(\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i (Arr))$ ,  
 $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i (Arr)). (\lambda i \in_{\circ} I. \mathfrak{A} i (Dom) (f(i))))$ ,  
 $(\lambda f \in_{\circ} (\prod_{\circ} i \in_{\circ} I. \mathfrak{A} i (Arr)). (\lambda i \in_{\circ} I. \mathfrak{A} i (Cod) (f(i))))$ ,  
 $(\lambda gf \in_{\circ} \text{composable-arrs} (dg\text{-prod } I \mathfrak{A}). (\lambda i \in_{\circ} I. gf(0)(i) \circ_{A\mathfrak{A} i} gf(1_{\mathbb{N}})(i)))$   
 ]<sub>o</sub>

**syntax** *-PSEMICATEGORY* ::  $pttrn \Rightarrow V \Rightarrow (V \Rightarrow V) \Rightarrow V$

$\langle \langle 3 \prod_{SMC-\epsilon_{\circ}} \cdot \rangle \rangle [0, 0, 10] 10$

**syntax-consts** *-PSEMICATEGORY*  $\Leftarrow$  *smc-prod*

**translations**  $\prod_{SMC} i \in_{\circ} I. \mathfrak{A} \Leftarrow$  *CONST* *smc-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**  $(\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i (Comp)) =$

$(\lambda gf \in_{\circ} \text{composable-arrs} (dg\text{-prod } I \mathfrak{A}). (\lambda i \in_{\circ} I. gf(0)(i) \circ_{A\mathfrak{A} i} gf(1_{\mathbb{N}})(i)))$

$\langle \text{proof} \rangle$

Slicing.

**lemma** *smc-dg-smc-prod[slicing-commute]*:

$dg\text{-prod } I (\lambda i. \text{smc-dg } (\mathfrak{A} i)) = \text{smc-dg } (\text{smc-prod } I \mathfrak{A})$

$\langle \text{proof} \rangle$

**context**

**fixes**  $\mathfrak{A} \varphi :: V \Rightarrow V$

**and**  $\mathfrak{C} :: V$

**begin**

**lemmas-with**

$[\text{where } \mathfrak{A} = \langle \lambda i. \text{smc-dg } (\mathfrak{A} i) \rangle, \text{ unfolded slicing-simps slicing-commute}]$ :

$\text{smc-prod-Obj} I = dg\text{-prod-Obj} I$

**and**  $\text{smc-prod-Obj} D = dg\text{-prod-Obj} D$

**and**  $\text{smc-prod-Obj} E = dg\text{-prod-Obj} E$

**and**  $\text{smc-prod-Obj-cong} = dg\text{-prod-Obj-cong}$

**and**  $\text{smc-prod-Arr} I = dg\text{-prod-Arr} I$

**and**  $\text{smc-prod-Arr} D = dg\text{-prod-Arr} D$

**and**  $\text{smc-prod-Arr} E = dg\text{-prod-Arr} E$

```

and smc-prod-Arr-cong = dg-prod-Arr-cong
and smc-prod-Dom-vsν[smc-cs-intros] = dg-prod-Dom-vsν
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-vsν[smc-cs-intros] = dg-prod-Cod-vsν
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 ↦ ∏DG i ∈◦ I.  $\mathfrak{A}$  i c ↔ g : b ↦ ∏SMC i ∈◦ I.  $\mathfrak{A}$  i c
⟨proof⟩

```

**lemma** *smc-prod-composable-arrs-dg-prod*:

```

composable-arrs (∏DG i ∈◦ I.  $\mathfrak{A}$  i) = composable-arrs (∏SMC i ∈◦ 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** *psemicategories*[*smc-prod-cs-intros*]:

*i* ∈<sub>◦</sub> *I* ⇒ *semicategory*  $\alpha$  ( $\mathfrak{A}$  *i*)

**and** *psemic-index-in-Vset*[*smc-cs-intros*]: *I* ∈<sub>◦</sub> *Vset*  $\alpha$

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* ( $\lambda i$ . *smc-dg* ( $\mathfrak{A}$  *i*))

**and**  $\bigwedge i$ . *i* ∈<sub>◦</sub> *I* ⇒ *semicategory*  $\alpha$  ( $\mathfrak{A}$  *i*)

**shows** *psemicategory-base*  $\alpha$  *I*  $\mathfrak{A}$

⟨*proof*⟩

Product semicategory is a product digraph.

**context** *psemicategory-base*

**begin**

**lemma** *psemic-pdigraph-base*: *pdigraph-base*  $\alpha$  *I* ( $\lambda i$ . *smc-dg* ( $\mathfrak{A}$  *i*))

⟨*proof*⟩

**interpretation** *pdg*: *pdigraph-base*  $\alpha$  *I*  $\langle \lambda i$ . *smc-dg* ( $\mathfrak{A}$  *i*)  $\rangle$

*<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**  $J \subseteq_o I$

**shows** *psemicategory-base*  $\alpha J \mathfrak{A}$

*<proof>*

## Composition

**lemma** *smc-prod-Comp*:

$(\prod_{SMC i \in_o I. \mathfrak{A} i})(Comp) =$

(  
 $\lambda gf \in_o \text{composable-arrs} (\prod_{SMC i \in_o I. \mathfrak{A} i).$   
 $(\lambda i \in_o I. gf(0)(i) \circ_{A \mathfrak{A} i} gf(1_{\mathbf{N}})(i))$   
 )

*<proof>*

**lemma** *smc-prod-Comp-vdomain*[*smc-cs-simps*]:

$\mathcal{D}_o ((\prod_{SMC i \in_o I. \mathfrak{A} i})(Comp)) = \text{composable-arrs} (\prod_{SMC i \in_o I. \mathfrak{A} i)$

*<proof>*

**lemma** *smc-prod-Comp-app*:

**assumes**  $g : b \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i c$  **and**  $f : a \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i b$

**shows**  $g \circ_A (\prod_{SMC i \in_o I. \mathfrak{A} i) f = (\lambda i \in_o I. g(i) \circ_{A \mathfrak{A} i} f(i))$

*<proof>*

**lemma** *smc-prod-Comp-app-component*[*smc-cs-simps*]:

**assumes**  $g : b \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i c$

**and**  $f : a \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i b$

**and**  $i \in_o I$

**shows**  $(g \circ_A (\prod_{SMC i \in_o I. \mathfrak{A} i) f)(i) = g(i) \circ_{A \mathfrak{A} i} f(i)$

*<proof>*

**lemma** (in *psemicategory-base*) *smc-prod-Comp-vrange*:

$\mathcal{R}_o ((\prod_{SMC i \in_o I. \mathfrak{A} i})(Comp)) \subseteq_o (\prod_{SMC i \in_o I. \mathfrak{A} i)(Arr)$

*<proof>*

**lemma** *smc-prod-Comp-app-vdomain*[*smc-cs-simps*]:

**assumes**  $g : b \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i c$  **and**  $f : a \mapsto \prod_{SMC i \in_o I. \mathfrak{A} i b$

**shows**  $\mathcal{D}_o (g \circ_A (\prod_{SMC i \in_o I. \mathfrak{A} i) f) = I$

*<proof>*

**A product  $\alpha$ -semicategory is a tiny  $\beta$ -semicategory****lemma** (in *psemicategory-base*) *psmc-tiny-semicategory-smc-prod*:assumes  $\mathcal{Z} \beta$  and  $\alpha \in_{\circ} \beta$ shows *tiny-semicategory*  $\beta$  ( $\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i$ )*<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*: $J \subseteq_{\circ} I \implies (\prod_{SMC} i \in_{\circ} J. \mathfrak{A} i) \langle Obj \rangle \subseteq_{\circ} Vset \alpha$ and *psmc-Hom-vifunion-in-Vset*:

[[

 $J \subseteq_{\circ} I;$  $A \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} J. \mathfrak{A} i) \langle Obj \rangle;$  $B \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} J. \mathfrak{A} i) \langle Obj \rangle;$  $A \in_{\circ} Vset \alpha;$  $B \in_{\circ} Vset \alpha$ ]]  $\implies (\cup_{\circ} a \in_{\circ} A. \cup_{\circ} b \in_{\circ} B. Hom (\prod_{SMC} i \in_{\circ} J. \mathfrak{A} i) a b) \in_{\circ} Vset \alpha$ 

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*  $\alpha I$  ( $\lambda i. smc-dg (\mathfrak{A} i)$ )and  $\wedge i. i \in_{\circ} I \implies semicategory \alpha (\mathfrak{A} i)$ shows *psemicategory*  $\alpha I \mathfrak{A}$ *<proof>*

Product semicategory is a product digraph.

**context** *psemicategory***begin****lemma** *psmc-pdigraph*: *pdigraph*  $\alpha I$  ( $\lambda i. smc-dg (\mathfrak{A} i)$ )*<proof>***interpretation** *pdg*: *pdigraph*  $\alpha I$   $\langle \lambda i. smc-dg (\mathfrak{A} i) \rangle$  *<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*  $\alpha$   $J$   $\mathfrak{A}$

*<proof>*

**A product  $\alpha$ -semicategory is an  $\alpha$ -semicategory**

**lemma** (in *psemicategory*) *psmc-semicategory-smc-prod*:

*semicategory*  $\alpha$   $(\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i)$

*<proof>*

#### 4.8.5 Local assumptions for a finite product semicategory

**Definition and elementary properties**

**locale** *finite-psemicategory* = *psemicategory-base*  $\alpha$   $I$   $\mathfrak{A}$  **for**  $\alpha$   $I$   $\mathfrak{A}$  +

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*  $\alpha'$   $I'$   $\mathfrak{A}$

*<proof>*

**mk-ide** **rf** *finite-psemicategory-def*[*unfolded finite-psemicategory-axioms-def*]

|*intro finite-psemicategory*  $I$ |

|*dest finite-psemicategory*  $D$ [*dest*]|

|*elim finite-psemicategory*  $E$ [*elim*]|

**lemmas** [*smc-prod-cs-intros*] = *finite-psemicategory*  $D(1)$

**lemma** *finite-psemicategory-finite-pdigraph*  $I$ :

assumes *finite-pdigraph*  $\alpha$   $I$   $(\lambda i. \text{smc-dg } (\mathfrak{A} i))$

**and**  $\bigwedge i. i \in_{\circ} I \implies \text{semicategory } \alpha$   $(\mathfrak{A} i)$

shows *finite-psemicategory*  $\alpha$   $I$   $\mathfrak{A}$

*<proof>*

**Local assumptions for a finite product semicategory and local assumptions for an arbitrary product semicategory**

**sublocale** *finite-psemicategory*  $\subseteq$  *psemicategory*  $\alpha$   $I$   $\mathfrak{A}$

*<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  $(g \circ_A (\prod_{SMC} j \in_{\circ} J. \mathfrak{A} j) f) \cup_{\circ} (g' \circ_A (\prod_{SMC} k \in_{\circ} K. \mathfrak{A} k) f') =$

$g \cup_{\circ} g' \circ_A (\prod_{SMC} j \in_{\circ} J \cup_{\circ} K. \mathfrak{A} j) f \cup_{\circ} f'$

*<proof>*

**lemma** (in *psemicategory*) *psmc-smc-prod-vdiff-vunion-Comp*:  
**assumes**  $J \subseteq_o I$   
**and**  $g : b \mapsto (\prod_{SMC} j \in_o I \text{ } \_ \circ J. \mathfrak{A} j) \text{ } c$   
**and**  $g' : b' \mapsto (\prod_{SMC} k \in_o J. \mathfrak{A} k) \text{ } c'$   
**and**  $f : a \mapsto (\prod_{SMC} j \in_o I \text{ } \_ \circ J. \mathfrak{A} j) \text{ } b$   
**and**  $f' : a' \mapsto (\prod_{SMC} k \in_o J. \mathfrak{A} k) \text{ } b'$   
**shows**  $(g \circ_A (\prod_{SMC} j \in_o I \text{ } \_ \circ J. \mathfrak{A} j) f) \cup_o (g' \circ_A (\prod_{SMC} j \in_o J. \mathfrak{A} j) f') =$   
 $g \cup_o g' \circ_A (\prod_{SMC} j \in_o I. \mathfrak{A} j) f \cup_o f'$   
*<proof>*

### 4.8.7 Projection

#### Definition and elementary properties

See Chapter II-3 in [39].

**definition** *smcf-proj* ::  $V \Rightarrow (V \Rightarrow V) \Rightarrow V \Rightarrow V$  (*π<sub>SMC</sub>*)  
**where**  $\pi_{SMC} I \mathfrak{A} i =$   
 $[$   
 $(\lambda a \in_o (\prod_{\circ} i \in_o I. \mathfrak{A} i \langle Obj \rangle)). a \langle i \rangle,$   
 $(\lambda f \in_o (\prod_{\circ} i \in_o I. \mathfrak{A} i \langle Arr \rangle)). f \langle i \rangle,$   
 $(\prod_{SMC} i \in_o I. \mathfrak{A} i),$   
 $\mathfrak{A} i$   
 $]$ .

Components.

**lemma** *smcf-proj-components*:  
**shows**  $(\pi_{SMC} I \mathfrak{A} i) \langle ObjMap \rangle = (\lambda a \in_o (\prod_{\circ} i \in_o I. \mathfrak{A} i \langle Obj \rangle)). a \langle i \rangle$   
**and**  $(\pi_{SMC} I \mathfrak{A} i) \langle ArrMap \rangle = (\lambda f \in_o (\prod_{\circ} i \in_o I. \mathfrak{A} i \langle Arr \rangle)). f \langle i \rangle$   
**and**  $(\pi_{SMC} I \mathfrak{A} i) \langle HomDom \rangle = (\prod_{SMC} i \in_o I. \mathfrak{A} i)$   
**and**  $(\pi_{SMC} I \mathfrak{A} i) \langle HomCod \rangle = \mathfrak{A} i$   
*<proof>*

Slicing

**lemma** *smcf-dghm-smcf-proj[slicing-commute]*:  
 $\pi_{DG} I (\lambda i. \text{smc-dg } (\mathfrak{A} i)) i = \text{smcf-dghm } (\pi_{SMC} I \mathfrak{A} i)$   
*<proof>*

**context** *psemicategory*  
**begin**

**interpretation** *pdg*: *pdigraph*  $\alpha I \langle \lambda i. \text{smc-dg } (\mathfrak{A} i) \rangle$  *<proof>*

**lemmas-with** [*unfolded slicing-simps slicing-commute*]:  
 $\text{smcf-proj-is-dghm} = \text{pdg.pdg-dghm-proj-is-dghm}$

**end**

#### Projection semifunctor is a semifunctor

**lemma** (in *psemicategory*) *psmc-smcf-proj-is-semifunctor*:  
**assumes**  $i \in_o I$   
**shows**  $\pi_{SMC} I \mathfrak{A} i : (\prod_{SMC} i \in_o I. \mathfrak{A} i) \mapsto_{SMC} \alpha \mathfrak{A} i$   
*<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**  $\pi_{SMC} I \mathfrak{A} i : \mathfrak{C} \mapsto \mapsto_{SMC\alpha} \mathfrak{D}$   
 $\langle proof \rangle$

**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*  $I \mathfrak{A} \mathfrak{C} \varphi =$   
 $[$   
 $(\lambda a \in_{\circ} \mathfrak{C}(\mathfrak{A} Obj). (\lambda i \in_{\circ} I. \varphi i(\mathfrak{A} ObjMap)(\mathfrak{A} a))),$   
 $(\lambda f \in_{\circ} \mathfrak{C}(\mathfrak{A} Arr). (\lambda i \in_{\circ} I. \varphi i(\mathfrak{A} ArrMap)(\mathfrak{A} f))),$   
 $\mathfrak{C},$   
 $(\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i)$   
 $]$

Components.

**lemma** *smcf-up-components*:

**shows** *smcf-up*  $I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{A} ObjMap) = (\lambda a \in_{\circ} \mathfrak{C}(\mathfrak{A} Obj). (\lambda i \in_{\circ} I. \varphi i(\mathfrak{A} ObjMap)(\mathfrak{A} a)))$   
**and** *smcf-up*  $I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{A} ArrMap) = (\lambda f \in_{\circ} \mathfrak{C}(\mathfrak{A} Arr). (\lambda i \in_{\circ} I. \varphi i(\mathfrak{A} ArrMap)(\mathfrak{A} f)))$   
**and** *smcf-up*  $I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{A} HomDom) = \mathfrak{C}$   
**and** *smcf-up*  $I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{A} HomCod) = (\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i)$   
 $\langle proof \rangle$

Slicing.

**lemma** *smcf-dghm-smcf-up[slicing-commute]*:

*dghm-up*  $I (\lambda i. \text{smc-dg } (\mathfrak{A} i)) (\text{smc-dg } \mathfrak{C}) (\lambda i. \text{smcf-dghm } (\varphi i)) =$   
 $\text{smcf-dghm } (\text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi)$   
 $\langle proof \rangle$

**context**

**fixes**  $\mathfrak{A} \varphi :: V \Rightarrow V$   
**and**  $\mathfrak{C} :: V$

**begin**

**lemmas-with**

$[$   
**where**  $\mathfrak{A} = \langle \lambda i. \text{smc-dg } (\mathfrak{A} i) \rangle$  **and**  $\varphi = \langle \lambda i. \text{smcf-dghm } (\varphi i) \rangle$  **and**  $\mathfrak{C} = \langle \text{smc-dg } \mathfrak{C} \rangle,$   
 $\text{unfolded slicing-simps slicing-commute}$   
 $]$   
 $\text{smcf-up-ObjMap-vdomain}[\text{smc-cs-simps}] = \text{dghm-up-ObjMap-vdomain}$   
**and**  $\text{smcf-up-ObjMap-app} = \text{dghm-up-ObjMap-app}$   
**and**  $\text{smcf-up-ObjMap-app-vdomain}[\text{smc-cs-simps}] = \text{dghm-up-ObjMap-app-vdomain}$   
**and**  $\text{smcf-up-ObjMap-app-component}[\text{smc-cs-simps}] = \text{dghm-up-ObjMap-app-component}$   
**and**  $\text{smcf-up-ArrMap-vdomain}[\text{smc-cs-simps}] = \text{dghm-up-ArrMap-vdomain}$   
**and**  $\text{smcf-up-ArrMap-app} = \text{dghm-up-ArrMap-app}$   
**and**  $\text{smcf-up-ArrMap-app-vdomain}[\text{smc-cs-simps}] = \text{dghm-up-ArrMap-app-vdomain}$   
**and**  $\text{smcf-up-ArrMap-app-component}[\text{smc-cs-simps}] = \text{dghm-up-ArrMap-app-component}$

**lemma** *smcf-up-ObjMap-vrange*:

**assumes**  $\bigwedge i. i \in_{\circ} I \implies \varphi i : \mathfrak{C} \mapsto \mapsto_{SMC\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_{\circ} (\text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi(\mathfrak{A} ObjMap)) \subseteq_{\circ} (\prod_{SMC} i \in_{\circ} I. \mathfrak{A} i)(\mathfrak{A} Obj)$

*<proof>*

**lemma** *smcf-up-ObjMap-app-vrange:*

**assumes**  $a \in_0 \mathfrak{C}(\text{Obj})$  **and**  $\bigwedge i. i \in_0 I \implies \varphi i : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o(\text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi(\text{ObjMap})(a)) \subseteq_o (\bigcup_o i \in_0 I. \mathfrak{A} i(\text{Obj}))$

*<proof>*

**lemma** *smcf-up-ArrMap-vrange:*

**assumes**  $\bigwedge i. i \in_0 I \implies \varphi i : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o(\text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi(\text{ArrMap})) \subseteq_o (\prod_{SMC} i \in_0 I. \mathfrak{A} i)(\text{Arr})$

*<proof>*

**lemma** *smcf-up-ArrMap-app-vrange:*

**assumes**  $a \in_0 \mathfrak{C}(\text{Arr})$  **and**  $\bigwedge i. i \in_0 I \implies \varphi i : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{A} i$   
**shows**  $\mathcal{R}_o(\text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi(\text{ArrMap})(a)) \subseteq_o (\bigcup_o i \in_0 I. \mathfrak{A} i(\text{Arr}))$

*<proof>*

**end**

**context** *psemicategory*

**begin**

**interpretation** *pdg: pdigraph*  $\alpha I \langle \lambda i. \text{smc-dg } (\mathfrak{A} i) \rangle$  *<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**  $\bigwedge i. i \in_0 I \implies \varphi i : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{A} i$   
**shows** *smcf-up*  $I \mathfrak{A} \mathfrak{C} \varphi : \mathfrak{C} \mapsto_{SMC\alpha} (\prod_{SMC} i \in_0 I. \mathfrak{A} i)$

*<proof>*

### Further properties

**lemma** (in *psemicategory*) *psmc-Comp-smcf-proj-smcf-up:*

**assumes** *semicategory*  $\alpha \mathfrak{C}$   
**and**  $\bigwedge i. i \in_0 I \implies \varphi i : \mathfrak{C} \mapsto_{SMC\alpha} \mathfrak{A} i$   
**and**  $i \in_0 I$

**shows**  $\varphi i = \pi_{SMC} I \mathfrak{A} i \circ_{SMCF} \text{smcf-up } I \mathfrak{A} \mathfrak{C} \varphi$

*<proof>*

**lemma** (in *psemicategory*) *psmc-smcf-up-eq-smcf-proj:*

**assumes**  $\mathfrak{F} : \mathfrak{C} \mapsto_{SMC\alpha} (\prod_{SMC} i \in_0 I. \mathfrak{A} i)$   
**and**  $\bigwedge i. i \in_0 I \implies \varphi i = \pi_{SMC} I \mathfrak{A} i \circ_{SMCF} \mathfrak{F}$

**shows** *smcf-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  $\mathfrak{C} = \langle \text{smc-dg } \mathfrak{C} \rangle$ , *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*  $\alpha$   $\langle \text{smc-dg } \mathfrak{C} \rangle$  *<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**  $j \in_{\circ} V \text{set } \alpha$   
**shows** *finite-psemicategory*  $\alpha$  (*set*  $\{j\}$ ) (*li.*  $\mathfrak{C}$ )  
*<proof>*

**lemma** (in *semicategory*) *smc-semicategory-smc-singleton*:  
**assumes**  $j \in_{\circ} V \text{set } \alpha$   
**shows** *semicategory*  $\alpha$  ( $\prod_{SMC} i \in_{\circ} \text{set } \{j\}$ ).  $\mathfrak{C}$ )  
*<proof>*

### 4.8.10 Singleton semifunctor

#### Definition and elementary properties

**definition** *smcf-singleton* ::  $V \Rightarrow V \Rightarrow V$   
**where** *smcf-singleton*  $j$   $\mathfrak{C} =$   
 [  $(\lambda a \in_{\circ} \mathfrak{C}(\text{Obj}). \text{set } \{(j, a)\})$ ,  
 $(\lambda f \in_{\circ} \mathfrak{C}(\text{Arr}). \text{set } \{(j, f)\})$ ,  
 $\mathfrak{C}$ ,  
 $(\prod_{SMC} i \in_{\circ} \text{set } \{j\}$ ).  $\mathfrak{C}$ )  
 ]<sub>o</sub>

Components.

**lemma** *smcf-singleton-components*:  
**shows** *smcf-singleton*  $j$   $\mathfrak{C}(\text{ObjMap}) = (\lambda a \in_{\circ} \mathfrak{C}(\text{Obj}). \text{set } \{(j, a)\})$   
**and** *smcf-singleton*  $j$   $\mathfrak{C}(\text{ArrMap}) = (\lambda f \in_{\circ} \mathfrak{C}(\text{Arr}). \text{set } \{(j, f)\})$   
**and** *smcf-singleton*  $j$   $\mathfrak{C}(\text{HomDom}) = \mathfrak{C}$   
**and** *smcf-singleton*  $j$   $\mathfrak{C}(\text{HomCod}) = (\prod_{SMC} i \in_{\circ} \text{set } \{j\}$ ).  $\mathfrak{C}$ )  
*<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  $\mathfrak{C} = \langle \text{smc-dg } \mathfrak{C} \rangle$ , *unfolded slicing-simps slicing-commute*]:

*smcf-singleton-ObjMap-vsuv*[*smc-cs-intros*] = *dghm-singleton-ObjMap-vsuv*

**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-vsuv*[*smc-cs-intros*] = *dghm-singleton-ArrMap-vsuv*

**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*  $\alpha$   $\langle \text{smc-dg } \mathfrak{C} \rangle$  *<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  $j \in_{\circ} Vset \alpha$

shows *smcf-singleton*  $j \mathfrak{C} : \mathfrak{C} \mapsto \mapsto_{SMC} iso \alpha \left( \prod_{SMC} i \in_{\circ} set \{j\}. \mathfrak{C} \right)$

*<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**  $\langle \times_{SMC} \rangle$  80)

where  $\mathfrak{A} \times_{SMC} \mathfrak{B} \equiv \text{smc-prod } (2_{\mathbb{N}}) (\lambda i. (i = 0 \ ? \ \mathfrak{A} : \mathfrak{B}))$

Slicing.

**lemma** *smc-dg-smc-prod-2*[*slicing-commute*]:

*smc-dg*  $\mathfrak{A} \times_{DG} \text{smc-dg } \mathfrak{B} = \text{smc-dg } (\mathfrak{A} \times_{SMC} \mathfrak{B})$

*<proof>*

**context**

fixes  $\alpha \mathfrak{A} \mathfrak{B}$

assumes  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$

**begin**

**interpretation**  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  *<proof>*

**interpretation**  $\mathfrak{B}$ : *semicategory*  $\alpha$   $\mathfrak{B}$   $\langle$ *proof* $\rangle$

**lemmas-with**

[  
 where  $\mathfrak{A}=\langle$ *smc-dg*  $\mathfrak{A}\rangle$  and  $\mathfrak{B}=\langle$ *smc-dg*  $\mathfrak{B}\rangle$ ,  
*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-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-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-vsuv* = *dg-prod-2-Dom-vsuv*  
 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-vsuv* = *dg-prod-2-Cod-vsuv*  
 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  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$

assumes  $\mathfrak{A}$ : *semicategory*  $\alpha$   $\mathfrak{A}$  and  $\mathfrak{B}$ : *semicategory*  $\alpha$   $\mathfrak{B}$

**begin**

**interpretation**  $\mathfrak{Z}$   $\alpha$   $\langle$ *proof* $\rangle$

**interpretation**  $\mathfrak{A}$ : *semicategory*  $\alpha$   $\mathfrak{A}$   $\langle$ *proof* $\rangle$

**interpretation**  $\mathfrak{B}$ : *semicategory*  $\alpha$   $\mathfrak{B}$   $\langle$ *proof* $\rangle$

**lemma** *finite-psemicategory-smc-prod-2*:

*finite-psemicategory*  $\alpha$   $\langle$ 2<sub>N</sub> $\rangle$  (*if2*  $\mathfrak{A}$   $\mathfrak{B}$ )

$\langle$ *proof* $\rangle$

**interpretation** *finite-psemicategory*  $\alpha$   $\langle$ 2<sub>N</sub> $\rangle$   $\langle$ *if2*  $\mathfrak{A}$   $\mathfrak{B}$  $\rangle$

$\langle$ *proof* $\rangle$

**lemma** *semicategory-smc-prod-2*[*smc-cs-intros*]: *semicategory*  $\alpha$   $(\mathfrak{A} \times_{SMC} \mathfrak{B})$

$\langle$ *proof* $\rangle$

**end**

**Composition**

**context**

fixes  $\alpha \mathfrak{A} \mathfrak{B}$   
 assumes  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$   
 begin

interpretation  $\mathcal{Z} \alpha$  *<proof>*

interpretation *finite-psemicategory*  $\alpha$   $\langle 2_{\mathbb{N}} \rangle$  *<if2  $\mathfrak{A} \mathfrak{B}$ >*  
*<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  $\alpha \mathfrak{A} \mathfrak{B}$   
 assumes  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$   
 begin

interpretation  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  *<proof>*

interpretation  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$  *<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* ::  $V \Rightarrow V \Rightarrow V$  ( $\langle \pi_{SMC.1} \rangle$ )  
 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* ::  $V \Rightarrow V \Rightarrow V$  ( $\langle \pi_{SMC.2} \rangle$ )  
 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  $\alpha \mathfrak{A} \mathfrak{B}$   
 assumes  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  and  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$   
 begin

interpretation  $\mathcal{Z} \alpha$  *<proof>*

**interpretation**  $\mathfrak{A}$ : *semicategory*  $\alpha$   $\mathfrak{A}$   $\langle$ *proof* $\rangle$

**interpretation**  $\mathfrak{B}$ : *semicategory*  $\alpha$   $\mathfrak{B}$   $\langle$ *proof* $\rangle$

**lemmas-with**

[

**where**  $\mathfrak{A}=\langle$ *smc-dg*  $\mathfrak{A}\rangle$  **and**  $\mathfrak{B}=\langle$ *smc-dg*  $\mathfrak{B}\rangle$ ,

*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*:  $\pi_{SMC.1} \mathfrak{A} \mathfrak{B}(\text{HomDom}) = \mathfrak{A} \times_{SMC} \mathfrak{B}$   
 $\langle$ *proof* $\rangle$

**lemma** *smcf-proj-fst-HomCod*:  $\pi_{SMC.1} \mathfrak{A} \mathfrak{B}(\text{HomCod}) = \mathfrak{A}$   
 $\langle$ *proof* $\rangle$

**lemma** *smcf-proj-snd-HomDom*:  $\pi_{SMC.2} \mathfrak{A} \mathfrak{B}(\text{HomDom}) = \mathfrak{A} \times_{SMC} \mathfrak{B}$   
 $\langle$ *proof* $\rangle$

**lemma** *smcf-proj-snd-HomCod*:  $\pi_{SMC.2} \mathfrak{A} \mathfrak{B}(\text{HomCod}) = \mathfrak{B}$   
 $\langle$ *proof* $\rangle$

**Projection of a product of two semicategories is a semifunctor**

**context**

**fixes**  $\alpha$   $\mathfrak{A} \mathfrak{B}$

**assumes**  $\mathfrak{A}$ : *semicategory*  $\alpha$   $\mathfrak{A}$  **and**  $\mathfrak{B}$ : *semicategory*  $\alpha$   $\mathfrak{B}$

**begin**

**interpretation**  $\mathcal{Z}$   $\alpha$   $\langle$ *proof* $\rangle$

**interpretation** *finite-psemicategory*  $\alpha$   $\langle$ 2 $\mathbb{N}$  $\rangle$   $\langle$ *if2*  $\mathfrak{A} \mathfrak{B}$  $\rangle$   
 $\langle$ *proof* $\rangle$

**lemma** *smcf-proj-fst-is-semifunctor*:

**assumes**  $i \in_{\circ} I$

**shows**  $\pi_{SMC.1} \mathfrak{A} \mathfrak{B} : \mathfrak{A} \times_{SMC} \mathfrak{B} \mapsto \mapsto_{SMC} \alpha \mathfrak{A}$

$\langle$ *proof* $\rangle$

**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**  $\pi_{SMC.1} \mathfrak{A} \mathfrak{B} : \mathfrak{C} \mapsto \mapsto_{SMC} \alpha \mathfrak{D}$

$\langle$ *proof* $\rangle$

**lemma** *smcf-proj-snd-is-semifunctor*:

**assumes**  $i \in_{\circ} I$

**shows**  $\pi_{SMC.2} \mathfrak{A} \mathfrak{B} : \mathfrak{A} \times_{SMC} \mathfrak{B} \mapsto \mapsto_{SMC} \alpha \mathfrak{B}$

$\langle$ *proof* $\rangle$

**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  $\pi_{SMC.2} \mathfrak{A} \mathfrak{B} : \mathfrak{C} \mapsto \pi_{SMC\alpha} \mathfrak{D}$   
 ⟨proof⟩

end

### 4.8.13 Product of three semicategories

**Definition and elementary properties.**

**definition** *smc-prod-3* ::  $V \Rightarrow V \Rightarrow V \Rightarrow V$   
 (⟨(- ×<sub>SMC3</sub> - ×<sub>SMC3</sub> -)⟩ [81, 81, 81] 80)  
 where  $\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C} = (\prod_{SMC} i \in \mathbb{N}. if3 \mathfrak{A} \mathfrak{B} \mathfrak{C} i)$

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  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{C}$   
 assumes  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$   
 and  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$   
 and  $\mathfrak{C}$ : *semicategory*  $\alpha \mathfrak{C}$

**begin**

**interpretation**  $\mathfrak{A}$ : *semicategory*  $\alpha \mathfrak{A}$  ⟨proof⟩  
**interpretation**  $\mathfrak{B}$ : *semicategory*  $\alpha \mathfrak{B}$  ⟨proof⟩  
**interpretation**  $\mathfrak{C}$ : *semicategory*  $\alpha \mathfrak{C}$  ⟨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  $smc-prod-3-ObjE' = dg-prod-3-ObjE'$   
 and  $smc-prod-3-ObjI = dg-prod-3-ObjI$   
 and  $smc-prod-3-ObjI'[smc-prod-cs-intros] = dg-prod-3-ObjI'$   
 and  $smc-prod-3-ObjE = dg-prod-3-ObjE$   
 and  $smc-prod-3-ObjE' = dg-prod-3-ObjE'$   
 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  $smc-prod-3-is-arrE' = dg-prod-3-is-arrE'$   
 and  $smc-prod-3-Dom-usv = dg-prod-3-Dom-usv$   
 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  $smc-prod-3-Cod-usv = dg-prod-3-Cod-usv$   
 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  $\alpha \mathfrak{A} \mathfrak{B} \mathfrak{C}$

```

assumes  $\mathfrak{A}$ : semicategory  $\alpha$   $\mathfrak{A}$ 
and  $\mathfrak{B}$ : semicategory  $\alpha$   $\mathfrak{B}$ 
and  $\mathfrak{C}$ : semicategory  $\alpha$   $\mathfrak{C}$ 
begin

interpretation  $\mathcal{Z}$   $\alpha$   $\langle$ proof $\rangle$ 
interpretation  $\mathfrak{A}$ : semicategory  $\alpha$   $\mathfrak{A}$   $\langle$ proof $\rangle$ 
interpretation  $\mathfrak{B}$ : semicategory  $\alpha$   $\mathfrak{B}$   $\langle$ proof $\rangle$ 
interpretation  $\mathfrak{C}$ : semicategory  $\alpha$   $\mathfrak{C}$   $\langle$ proof $\rangle$ 

lemma finite-psemicategory-smc-prod-3:
  finite-psemicategory  $\alpha$   $\langle$  $\exists_{\mathbb{N}}$  $\rangle$   $\langle$ if3  $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{C}$  $\rangle$ 
 $\langle$ proof $\rangle$ 

interpretation finite-psemicategory  $\alpha$   $\langle$  $\exists_{\mathbb{N}}$  $\rangle$   $\langle$ if3  $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{C}$  $\rangle$ 
 $\langle$ proof $\rangle$ 

lemma semicategory-smc-prod-3[smc-cs-intros]:
  semicategory  $\alpha$   $(\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C})$ 
 $\langle$ proof $\rangle$ 

```

**end**

## Composition

```

context
fixes  $\alpha$   $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{C}$ 
assumes  $\mathfrak{A}$ : semicategory  $\alpha$   $\mathfrak{A}$ 
and  $\mathfrak{B}$ : semicategory  $\alpha$   $\mathfrak{B}$ 
and  $\mathfrak{C}$ : semicategory  $\alpha$   $\mathfrak{C}$ 
begin

interpretation  $\mathcal{Z}$   $\alpha$   $\langle$ proof $\rangle$ 

interpretation finite-psemicategory  $\alpha$   $\langle$  $\exists_{\mathbb{N}}$  $\rangle$   $\langle$ if3  $\mathfrak{A}$   $\mathfrak{B}$   $\mathfrak{C}$  $\rangle$ 
 $\langle$ proof $\rangle$ 

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
 $[g, g', g'']_{\circ} \circ_{A\mathfrak{A} \times_{SMC3} \mathfrak{B} \times_{SMC3} \mathfrak{C}} [f, f', f'']_{\circ} =$ 
 $[g \circ_{A\mathfrak{A}} f, g' \circ_{A\mathfrak{B}} f', g'' \circ_{A\mathfrak{C}} f'']_{\circ}$ 
 $\langle$ proof $\rangle$ 

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*  $\alpha$   $\mathfrak{B}$  + dg: *semicategory*  $\alpha$   $\mathfrak{C}$  for  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$  +  
 assumes *subsmc-subdigraph*[*slicing-intros*]: *smc-dg*  $\mathfrak{B} \subseteq_{DG\alpha}$  *smc-dg*  $\mathfrak{C}$   
 and *subsmc-Comp*[*smc-sub-fw-cs-intros*]:  
 [[  $g : b \mapsto_{\mathfrak{B}} c$ ;  $f : a \mapsto_{\mathfrak{B}} b$  ]]  $\implies g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$

abbreviation *is-subsemicategory* ( $\langle (-/ \subseteq_{SMC\alpha} -) \rangle$  [51, 51] 50)

where  $\mathfrak{B} \subseteq_{SMC\alpha}$   $\mathfrak{C} \equiv$  *subsemicategory*  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$

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*  $\alpha$   $\mathfrak{B}$   
 and *semicategory*  $\alpha$   $\mathfrak{C}$   
 and  $\bigwedge a. a \in_{\circ} \mathfrak{B}(\text{Obj}) \implies a \in_{\circ} \mathfrak{C}(\text{Obj})$   
 and  $\bigwedge a b f. f : a \mapsto_{\mathfrak{B}} b \implies f : a \mapsto_{\mathfrak{C}} b$   
 and  $\bigwedge b c g a f. [[ g : b \mapsto_{\mathfrak{B}} c; f : a \mapsto_{\mathfrak{B}} b ]] \implies$   
    $g \circ_{A\mathfrak{B}} f = g \circ_{A\mathfrak{C}} f$   
 shows  $\mathfrak{B} \subseteq_{SMC\alpha}$   $\mathfrak{C}$   
 <proof>

Subsemicategory is a subdigraph.

context *subsemicategory*  
 begin

interpretation *subdg*: *subdigraph*  $\alpha$   $\langle$  *smc-dg*  $\mathfrak{B}$   $\rangle$   $\langle$  *smc-dg*  $\mathfrak{C}$   $\rangle$

*<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**  $f : \mapsto_{id\mathfrak{B}} b$   
**shows**  $f : \mapsto_{id\mathfrak{C}} b$   
*<proof>*

**lemmas** [*smc-sub-fw-cs-intros*] = *subsemicategory.subsmc-is-idem-arrD*

**Subsemicategory relation is a partial order**

**lemma** *subsmc-refl*:  
**assumes** *semicategory*  $\alpha$   $\mathfrak{A}$

**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*  $\equiv$  *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*  $\subseteq inc: is-ft-semifunctor \alpha \mathfrak{B} \mathfrak{C} \langle smcf-inc \mathfrak{B} \mathfrak{C} \rangle$   
 ⟨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*]:  
 $smcf-inc (op-smc \mathfrak{B}) (op-smc \mathfrak{C}) : op-smc \mathfrak{B} \mapsto_{SMC} \mapsto_{SMC} \alpha op-smc \mathfrak{C}$   
 ⟨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* ( $\langle -/ \subseteq_{SMC.full} - \rangle$ ) [51, 51] 50  
**where**  $\mathfrak{B} \subseteq_{SMC.full\alpha}$   $\mathfrak{C} \equiv$  *fl-subsemicategory*  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$

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*  $\subseteq$  *inc: is-fl-semifunctor*  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$  *<smcf-inc*  $\mathfrak{B}$   $\mathfrak{C}$   
*<proof>*

#### 4.9.5 Wide subsemicategory

##### Definition and elementary properties

See [3]<sup>4</sup>).

**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* ( $\langle -/ \subseteq_{SMC.wide} - \rangle$ ) [51, 51] 50  
**where**  $\mathfrak{B} \subseteq_{SMC.wide\alpha}$   $\mathfrak{C} \equiv$  *wide-subsemicategory*  $\alpha$   $\mathfrak{B}$   $\mathfrak{C}$

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!]|*

<sup>4</sup><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*  $\alpha$   $\langle$ *smc-dg*  $\mathfrak{B}$  $\rangle$   $\langle$ *smc-dg*  $\mathfrak{C}$  $\rangle$   
 $\langle$ *proof* $\rangle$

**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*  $\alpha$   $\mathfrak{A}$

**shows**  $\mathfrak{A} \subseteq_{SMC.wide\alpha} \mathfrak{A}$

$\langle$ *proof* $\rangle$

**lemma** *wide-subsmc-trans*[*trans*]:

**assumes**  $\mathfrak{A} \subseteq_{SMC.wide\alpha} \mathfrak{B}$  **and**  $\mathfrak{B} \subseteq_{SMC.wide\alpha} \mathfrak{C}$

**shows**  $\mathfrak{A} \subseteq_{SMC.wide\alpha} \mathfrak{C}$

$\langle$ *proof* $\rangle$

**lemma** *wide-subsmc-antisym*:

**assumes**  $\mathfrak{A} \subseteq_{SMC.wide\alpha} \mathfrak{B}$  **and**  $\mathfrak{B} \subseteq_{SMC.wide\alpha} \mathfrak{A}$

**shows**  $\mathfrak{A} = \mathfrak{B}$

$\langle$ *proof* $\rangle$

## 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**  $smc-0 = [0, 0, 0, 0, 0]$ .

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**  $smc-0(Comp) = 0$   
 $\langle proof \rangle$

Slicing.

**lemma**  $smc-dg-smc-0$ :  $smc-dg\ smc-0 = dg-0$   
 $\langle proof \rangle$

**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  $\mathcal{Z}$ )  $semicategory-smc-0[smc-cs-intros]$ :  $semicategory\ \alpha\ smc-0$   
 $\langle proof \rangle$

**lemmas** [ $smc-cs-intros$ ] =  $\mathcal{Z}.semicategory-smc-0$

#### Opposite of the semicategory $0$

**lemma**  $op-smc-smc-0[smc-op-simps]$ :  $op-smc\ (smc-0) = smc-0$   
 $\langle proof \rangle$

#### A semicategory without objects is empty

**lemma** (in  $semicategory$ )  $smc-smc-0-if-Obj-0$ :  
**assumes**  $\mathcal{C}(Obj) = 0$   
**shows**  $\mathcal{C} = smc-0$   
 $\langle proof \rangle$

### 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  $smcf-0 \mathfrak{A} = [0, 0, smc-0, \mathfrak{A}]_o$ .

Components.

**lemma**  $smcf-0-components$ :

shows  $smcf-0 \mathfrak{A}(\text{ObjMap}) = 0$

and  $smcf-0 \mathfrak{A}(\text{ArrMap}) = 0$

and  $smcf-0 \mathfrak{A}(\text{HomDom}) = smc-0$

and  $smcf-0 \mathfrak{A}(\text{HomCod}) = \mathfrak{A}$

$\langle proof \rangle$

Slicing.

**lemma**  $smcf-dghm-smcf-0$ :  $smcf-dghm (smcf-0 \mathfrak{A}) = dghm-0 (smc-dg \mathfrak{A})$

$\langle proof \rangle$

Opposite empty semicategory homomorphism.

**lemma**  $op-smcf-smcf-0$ :  $op-smcf (smcf-0 \mathfrak{C}) = smcf-0 (op-smc \mathfrak{C})$

$\langle proof \rangle$

**Object map**

**lemma**  $smcf-0-ObjMap-vsuv[smc-cs-intros]$ :  $vsuv (smcf-0 \mathfrak{C}(\text{ObjMap}))$

$\langle proof \rangle$

**Arrow map**

**lemma**  $smcf-0-ArrMap-vsuv[smc-cs-intros]$ :  $vsuv (smcf-0 \mathfrak{C}(\text{ArrMap}))$

$\langle proof \rangle$

**Empty semifunctor is a faithful semifunctor**

**lemma** (in  $\mathcal{Z}$ )  $smcf-0-is-ft-semifunctor$ :

assumes  $semicategory \alpha \mathfrak{A}$

shows  $smcf-0 \mathfrak{A} : smc-0 \mapsto_{SMC.f\text{aithful}\alpha} \mathfrak{A}$

$\langle proof \rangle$

**lemma** (in  $\mathcal{Z}$ )  $smcf-0-is-ft-semifunctor'[smcf-cs-intros]$ :

assumes  $semicategory \alpha \mathfrak{A}$

and  $\mathfrak{B}' = \mathfrak{A}$

and  $\mathfrak{A}' = smc-0$

shows  $smcf-0 \mathfrak{A} : \mathfrak{A}' \mapsto_{SMC.f\text{aithful}\alpha} \mathfrak{B}'$

$\langle proof \rangle$

**lemmas**  $[smcf-cs-intros] = \mathcal{Z}.smcf-0-is-ft-semifunctor'$

**lemma** (in  $\mathcal{Z}$ )  $smcf-0-is-semifunctor$ :

assumes  $semicategory \alpha \mathfrak{A}$

shows  $smcf-0 \mathfrak{A} : smc-0 \mapsto_{SMC\alpha} \mathfrak{A}$

$\langle proof \rangle$

**lemma** (in  $\mathcal{Z}$ )  $smcf-0-is-semifunctor'[smc-cs-intros]$ :

assumes  $semicategory \alpha \mathfrak{A}$

and  $\mathfrak{B}' = \mathfrak{A}$

and  $\mathfrak{A}' = smc-0$

shows  $smcf-0 \mathfrak{A} : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$

$\langle proof \rangle$

lemmas [smc-cs-intros] =  $\mathcal{Z}.smcf\text{-}0\text{-is-semifunctor}'$

### Further properties

lemma *is-semifunctor-is-smcf-0-if-smc-0*:

assumes  $\mathfrak{F} : smc\text{-}0 \mapsto \mapsto_{SMC\alpha} \mathfrak{C}$

shows  $\mathfrak{F} = smcf\text{-}0 \mathfrak{C}$

*<proof>*

## 4.10.4 Empty natural transformation of semifunctors

### Definition and elementary properties

definition *ntsmcf-0* ::  $V \Rightarrow V$

where *ntsmcf-0*  $\mathfrak{C} = [0, smcf\text{-}0 \mathfrak{C}, smcf\text{-}0 \mathfrak{C}, smc\text{-}0, \mathfrak{C}]_0$ .

Components.

lemma *ntsmcf-0-components*:

shows *ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TMap) = 0$

and [smc-cs-simps]: *ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TDom) = smcf\text{-}0 \mathfrak{C}$

and [smc-cs-simps]: *ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TCod) = smcf\text{-}0 \mathfrak{C}$

and [smc-cs-simps]: *ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TDGDom) = smc\text{-}0$

and [smc-cs-simps]: *ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TDGCod) = \mathfrak{C}$

*<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-vsuv*[smc-cs-intros]: *vsuv* (*ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TMap)$ )

*<proof>*

lemma *ntsmcf-0-NTMap-vdomain*[smc-cs-simps]:  $\mathcal{D}_\circ$  (*ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TMap)$ ) = 0

*<proof>*

lemma *ntsmcf-0-NTMap-vrange*[smc-cs-simps]:  $\mathcal{R}_\circ$  (*ntsmcf-0*  $\mathfrak{C}(\mathcal{N}TMap)$ ) = 0

*<proof>*

## Empty natural transformation of semifunctors is a natural transformation of semifunctors

lemma (in *semicategory*) *smc-ntsmcf-0-is-ntsmcfI*:

*ntsmcf-0*  $\mathfrak{C} : smcf\text{-}0 \mathfrak{C} \mapsto \mapsto_{SMCF} smcf\text{-}0 \mathfrak{C} : smc\text{-}0 \mapsto \mapsto_{SMC\alpha} \mathfrak{C}$

*<proof>*

lemma (in *semicategory*) *smc-ntsmcf-0-is-ntsmcfI'*[smc-cs-intros]:

assumes  $\mathfrak{F}' = smcf\text{-}0 \mathfrak{C}$

and  $\mathfrak{G}' = smcf\text{-}0 \mathfrak{C}$

and  $\mathfrak{A}' = smc\text{-}0$

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

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

and  $\mathfrak{G}' = \mathfrak{G}$   
 shows  $ntsmcf\text{-}0 \ \mathfrak{C} : \mathfrak{F}' \mapsto_{SMCF} \mathfrak{G}' : \mathfrak{A}' \mapsto_{SMC\alpha} \mathfrak{B}'$   
 ⟨proof⟩

lemmas [smc-cs-intros] = semicategory.smc-ntsmcf-0-is-ntsmcfI'

lemma *is-ntsmcf-is-ntsmcf-0-if-smc-0*:  
 assumes  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto_{SMC\alpha} \mathfrak{C}$   
 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  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto_{SMC\alpha} \mathfrak{C}$   
 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  $\mathfrak{N} : \mathfrak{F} \mapsto_{SMCF} \mathfrak{G} : smc\text{-}0 \mapsto_{SMC\alpha} \mathfrak{C}$   
 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-10*  $\mathfrak{a} = [\text{set } \{\mathfrak{a}\}, 0, 0, 0, 0]$ .

Components.

lemma *smc-10-components*:  
 shows *smc-10*  $\mathfrak{a}(\text{Obj}) = \text{set } \{\mathfrak{a}\}$   
 and *smc-10*  $\mathfrak{a}(\text{Arr}) = 0$   
 and *smc-10*  $\mathfrak{a}(\text{Dom}) = 0$   
 and *smc-10*  $\mathfrak{a}(\text{Cod}) = 0$   
 and *smc-10*  $\mathfrak{a}(\text{Comp}) = 0$   
 ⟨proof⟩

Slicing.

lemma *smc-dg-smc-10*: *smc-dg* (*smc-10*  $\mathfrak{a}$ ) = (*dg-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  $\mathfrak{a} \in_{\circ} V\text{set } \alpha$   
 shows *semicategory*  $\alpha$  (*smc-10*  $\mathfrak{a}$ )  
 ⟨proof⟩

### Arrow with a domain and a codomain

lemma *smc-10-is-arr-iff*:  $\mathfrak{F} : \mathfrak{A} \mapsto_{smc\text{-}10} \mathfrak{a} \ \mathfrak{B} \longleftrightarrow \text{False}$   
 ⟨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**  $smc-1 \text{ a } f =$

$[set \{a\}, set \{f\}, set \{f, a\}, set \{f, a\}, set \{([f, f]_o, f)\}]_o$

Components.

**lemma**  $smc-1-components:$

**shows**  $smc-1 \text{ a } f(\text{Obj}) = set \{a\}$

**and**  $smc-1 \text{ a } f(\text{Arr}) = set \{f\}$

**and**  $smc-1 \text{ a } f(\text{Dom}) = set \{f, a\}$

**and**  $smc-1 \text{ a } f(\text{Cod}) = set \{f, a\}$

**and**  $smc-1 \text{ a } f(\text{Comp}) = set \{([f, f]_o, f)\}$

$\langle proof \rangle$

Slicing.

**lemma**  $dg-smc-1: smc-dg (smc-1 \text{ a } f) = dg-1 \text{ a } f$

$\langle proof \rangle$

**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**  $smc-1-Comp-app[simp]: f \circ_{A_{smc-1} \text{ a } f} f = f$

$\langle proof \rangle$

## 1 is a semicategory

**lemma**  $(in \mathcal{Z}) \text{ semicategory-smc-1:}$

**assumes**  $a \in_o Vset \alpha$  **and**  $f \in_o Vset \alpha$

**shows**  $semicategory \alpha (smc-1 \text{ a } f)$

$\langle proof \rangle$

## 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]<sup>5</sup>.

**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-v11*  
*arr-Rel-comp-Rel-converse-Rel-right-if-v11*

**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*  $\alpha$  =

[  
  *Vset*  $\alpha$ ,  
  *set* {*T*. *arr-Rel*  $\alpha$  *T*},  
  ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Rel } \alpha T\}. T(\text{ArrDom})$ ),  
  ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Rel } \alpha T\}. T(\text{ArrCod})$ ),  
  ( $\lambda ST \in_{\circ} \text{composable-arrs } (dg-Rel \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}})$ )  
]

Components.

**lemma** *smc-Rel-components*:

**shows** *smc-Rel*  $\alpha(\text{Obj})$  = *Vset*  $\alpha$

**and** *smc-Rel*  $\alpha(\text{Arr})$  = *set* {*T*. *arr-Rel*  $\alpha$  *T*}

**and** *smc-Rel*  $\alpha(\text{Dom})$  = ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Rel } \alpha T\}. T(\text{ArrDom})$ )

**and** *smc-Rel*  $\alpha(\text{Cod})$  = ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Rel } \alpha T\}. T(\text{ArrCod})$ )

**and** *smc-Rel*  $\alpha(\text{Comp})$  = ( $\lambda ST \in_{\circ} \text{composable-arrs } (dg-Rel \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}})$ )

*<proof>*

Slicing.

<sup>5</sup><https://ncatlab.org/nlab/show/Rel>

**lemma** *smc-dg-smc-Rel*:  $smc-dg (smc-Rel \alpha) = dg-Rel \alpha$   
 ⟨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**  $smc-Rel-Dom-vsuv[smc-Rel-cs-intros] = dg-Rel-Dom-vsuv$   
**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**  $smc-Rel-Cod-vsuv[smc-Rel-cs-intros] = dg-Rel-Cod-vsuv$   
**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*]:  
 $smc-Rel-Hom-vifunior-in-Vset = dg-Rel-Hom-vifunior-in-Vset$   
**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*:  
 $composable-arrs (dg-Rel \alpha) = composable-arrs (smc-Rel \alpha)$   
 ⟨proof⟩

**lemma** *smc-Rel-Comp*:  
 $smc-Rel \alpha \langle Comp \rangle = (\lambda ST \epsilon_0. composable-arrs (smc-Rel \alpha). ST \langle 0 \rangle \circ_{Rel} ST \langle 1_N \rangle)$   
 ⟨proof⟩

### Composition

**lemma** *smc-Rel-Comp-app*[*smc-Rel-cs-simps*]:  
**assumes**  $S : b \mapsto_{smc-Rel \alpha} c$  **and**  $T : a \mapsto_{smc-Rel \alpha} b$   
**shows**  $S \circ_{A_{smc-Rel \alpha}} T = S \circ_{Rel} T$   
 ⟨proof⟩

**lemma** *smc-Rel-Comp-vdomain*:  $\mathcal{D}_\circ (smc-Rel \alpha \langle Comp \rangle) = composable-arrs (smc-Rel \alpha)$   
 ⟨proof⟩

**lemma** (**in**  $\mathcal{Z}$ ) *smc-Rel-Comp-vrange*:  $\mathcal{R}_\circ (smc-Rel \alpha \langle Comp \rangle) \subseteq_\circ set \{T. arr-Rel \alpha T\}$   
 ⟨proof⟩

### Rel is a semicategory

**lemma** (**in**  $\mathcal{Z}$ ) *semicategory-smc-Rel*:  $semicategory \alpha (smc-Rel \alpha)$   
 ⟨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  $\dagger_{SMC.Rel} \alpha =$   
 $[$   
 $\quad vid-on (smc-Rel \alpha \langle Obj \rangle),$   
 $\quad VLambda (smc-Rel \alpha \langle Arr \rangle) \textit{ converse-Rel},$   
 $\quad op-smc (smc-Rel \alpha),$   
 $\quad smc-Rel \alpha$   
 $]$ .

Components.

**lemma**  $smcf-dag-Rel-components:$

**shows**  $\dagger_{SMC.Rel} \alpha \langle ObjMap \rangle = vid-on (smc-Rel \alpha \langle Obj \rangle)$   
**and**  $\dagger_{SMC.Rel} \alpha \langle ArrMap \rangle = VLambda (smc-Rel \alpha \langle Arr \rangle) \textit{ converse-Rel}$   
**and**  $\dagger_{SMC.Rel} \alpha \langle HomDom \rangle = op-smc (smc-Rel \alpha)$   
**and**  $\dagger_{SMC.Rel} \alpha \langle HomCod \rangle = smc-Rel \alpha$   
 $\langle proof \rangle$

Slicing.

**lemma**  $smcf-dghm-smcf-dag-Rel: smcf-dghm (\dagger_{SMC.Rel} \alpha) = \dagger_{DG.Rel} \alpha$   
 $\langle proof \rangle$

**lemmas-with**  $[$

$\quad folded \textit{ smc-dg-smc-Rel smcf-dghm-smcf-dag-Rel, unfolded slicing-simps}$   
 $\quad ]:$   
 $smcf-dag-Rel-ObjMap-vsuv[smc-Rel-cs-intros] = dghm-dag-Rel-ObjMap-vsuv$   
**and**  $smcf-dag-Rel-ObjMap-vdomain[smc-Rel-cs-simps] =$   
 $\quad dghm-dag-Rel-ObjMap-vdomain$   
**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**  $smcf-dag-Rel-ArrMap-vsuv[smc-Rel-cs-intros] = dghm-dag-Rel-ArrMap-vsuv$   
**and**  $smcf-dag-Rel-ArrMap-vdomain[smc-Rel-cs-simps] =$   
 $\quad dghm-dag-Rel-ArrMap-vdomain$   
**and**  $smcf-dag-Rel-ArrMap-app[smc-Rel-cs-simps] = dghm-dag-Rel-ArrMap-app$   
**and**  $smcf-dag-Rel-ArrMap-app-vdomain[smc-cs-simps] =$   
 $\quad dghm-dag-Rel-ArrMap-app-vdomain$   
**and**  $smcf-dag-Rel-ArrMap-app-vrange[smc-cs-simps] =$   
 $\quad dghm-dag-Rel-ArrMap-app-vrange$   
**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  $\mathcal{Z}$ )  $smcf-dag-Rel-is-iso-semifunctor:$

$\dagger_{SMC.Rel} \alpha : op-smc (smc-Rel \alpha) \mapsto \mapsto_{SMC.iso} \alpha \textit{ smc-Rel} \alpha$   
 $\langle proof \rangle$

#### Further properties of the canonical dagger

**lemma** (in  $\mathcal{Z}$ )  $smcf-cn-comp-smcf-dag-Rel-smcf-dag-Rel:$

$\dagger_{SMC.Rel} \alpha \textit{ SMC}F^\circ \dagger_{SMC.Rel} \alpha = smcf-id (smc-Rel \alpha)$   
 $\langle proof \rangle$

**lemma**  $smcf-dag-Rel-ArrMap-smc-Rel-Comp:$

**assumes**  $S : b \mapsto_{smc-Rel \alpha} c$  **and**  $T : a \mapsto_{smc-Rel \alpha} b$

**shows**  $\dagger_{SMC.Rel} \alpha(\text{ArrMap})(S \circ_{A smc-Rel} \alpha T) =$   
 $\dagger_{SMC.Rel} \alpha(\text{ArrMap})(T) \circ_{A smc-Rel} \alpha \dagger_{SMC.Rel} \alpha(\text{ArrMap})(S)$   
 ⟨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]<sup>6</sup>.

**context**

**begin**

**private lemma** *smc-Rel-is-monic-arr-vsubset*:

**assumes**  $T : A \mapsto_{smc-Rel} \alpha B$   
**and**  $R : A' \mapsto_{smc-Rel} \alpha A$   
**and**  $S : A' \mapsto_{smc-Rel} \alpha A$   
**and**  $T \circ_{A smc-Rel} \alpha R = T \circ_{A smc-Rel} \alpha S$   
**and**  $\bigwedge y z X.$   
 $\llbracket y \subseteq_o A; z \subseteq_o A; T(\text{ArrVal}) \text{ ' } \circ y = X; T(\text{ArrVal}) \text{ ' } \circ z = X \rrbracket \implies y = z$   
**shows**  $R(\text{ArrVal}) \subseteq_o S(\text{ArrVal})$   
 ⟨proof⟩

**lemma** *smc-Rel-is-monic-arrI*:

**assumes**  $T : A \mapsto_{smc-Rel} \alpha B$   
**and**  $\bigwedge y z X. \llbracket y \subseteq_o A; z \subseteq_o A; T(\text{ArrVal}) \text{ ' } \circ y = X; T(\text{ArrVal}) \text{ ' } \circ z = X \rrbracket \implies$   
 $y = z$   
**shows**  $T : A \mapsto_{mon smc-Rel} \alpha B$   
 ⟨proof⟩

**end**

**lemma** *smc-Rel-is-monic-arrD[dest]*:

**assumes**  $T : A \mapsto_{mon smc-Rel} \alpha B$   
**and**  $y \subseteq_o A$   
**and**  $z \subseteq_o A$   
**and**  $T(\text{ArrVal}) \text{ ' } \circ y = X$   
**and**  $T(\text{ArrVal}) \text{ ' } \circ z = X$   
**shows**  $y = z$   
 ⟨proof⟩

**lemma** *smc-Rel-is-monic-arr*:

$T : A \mapsto_{mon smc-Rel} \alpha B \iff$   
 $T : A \mapsto_{smc-Rel} \alpha B \wedge$   
 (  
 $\forall y z X.$   
 $y \subseteq_o A \longrightarrow$   
 $z \subseteq_o A \longrightarrow$   
 $(T(\text{ArrVal}) \text{ ' } \circ y = X \longrightarrow$   
 $(T(\text{ArrVal}) \text{ ' } \circ z = X \longrightarrow$   
 $y = z$   
 )  
 ⟨proof⟩

**lemma** *smc-Rel-is-monic-arr-is-epic-arr*:

**assumes**  $T : A \mapsto_{smc-Rel} \alpha B$   
**and**  $(\dagger_{SMC.Rel} \alpha)(\text{ArrMap})(T) : B \mapsto_{mon smc-Rel} \alpha A$   
**shows**  $T : A \mapsto_{epi smc-Rel} \alpha B$   
 ⟨proof⟩

<sup>6</sup><https://ncatlab.org/nlab/show/Rel>

**lemma** *smc-Rel-is-epic-arr-is-monic-arr*:

**assumes**  $T : A \mapsto_{\text{epi smc-Rel } \alpha} B$   
**shows**  $\dagger_{\text{SMC.Rel } \alpha} (\text{ArrMap}) (T) : B \mapsto_{\text{mon smc-Rel } \alpha} A$   
 ⟨proof⟩

**lemma** *smc-Rel-is-epic-arrI*:

**assumes**  $T : A \mapsto_{\text{smc-Rel } \alpha} B$   
**and**  $\bigwedge y z X. \llbracket y \subseteq_{\circ} B; z \subseteq_{\circ} B; T(\text{ArrVal}) -'_{\circ} y = X; T(\text{ArrVal}) -'_{\circ} z = X \rrbracket \implies$   
 $y = z$   
**shows**  $T : A \mapsto_{\text{epi smc-Rel } \alpha} B$   
 ⟨proof⟩

**lemma** *smc-Rel-is-epic-arrD[dest]*:

**assumes**  $T : A \mapsto_{\text{epi smc-Rel } \alpha} B$   
**and**  $y \subseteq_{\circ} B$   
**and**  $z \subseteq_{\circ} B$   
**and**  $T(\text{ArrVal}) -'_{\circ} y = X$   
**and**  $T(\text{ArrVal}) -'_{\circ} z = X$   
**shows**  $y = z$   
 ⟨proof⟩

**lemma** *smc-Rel-is-epic-arr*:

$T : A \mapsto_{\text{epi smc-Rel } \alpha} B \iff$   
 $T : A \mapsto_{\text{smc-Rel } \alpha} B \wedge$   
 (  
 $\forall y z X.$   
 $y \subseteq_{\circ} B \implies$   
 $z \subseteq_{\circ} B \implies$   
 $T(\text{ArrVal}) -'_{\circ} y = X \implies$   
 $T(\text{ArrVal}) -'_{\circ} z = X \implies$   
 $y = z$   
 )  
 ⟨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])<sup>7</sup>.

**lemma** (in  $\mathcal{Z}$ ) *smc-Rel-obj-terminal*:  $\text{obj-terminal } (\text{smc-Rel } \alpha) A \iff A = 0$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *smc-Rel-obj-initial*:  $\text{obj-initial } (\text{smc-Rel } \alpha) A \iff A = 0$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *smc-Rel-obj-terminal-obj-initial*:  
 $\text{obj-initial } (\text{smc-Rel } \alpha) A \iff \text{obj-terminal } (\text{smc-Rel } \alpha) A$   
 ⟨proof⟩

**lemma** (in  $\mathcal{Z}$ ) *smc-Rel-obj-null*:  $\text{obj-null } (\text{smc-Rel } \alpha) A \iff A = 0$   
 ⟨proof⟩

<sup>7</sup><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**  $A \in_{\circ} \text{smc-Rel } \alpha(\text{Obj})$  **and**  $B \in_{\circ} \text{smc-Rel } \alpha(\text{Obj})$

**shows**  $T : A \mapsto_{\circ \text{smc-Rel } \alpha} B \longleftrightarrow T = [0, A, B]_{\circ}$ .

*<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*  $\alpha$  =

[  
  *Vset*  $\alpha$ ,  
  *set* {*T*. *arr-Par*  $\alpha$  *T*},  
  ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Par } \alpha T\}. T(\text{ArrDom})$ ),  
  ( $\lambda T \in_{\circ} \text{set } \{T. \text{arr-Par } \alpha T\}. T(\text{ArrCod})$ ),  
  ( $\lambda ST \in_{\circ} \text{composable-arrs } (dg\text{-Par } \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}})$ )  
]

Components.

**lemma** *smc-Par-components*:

**shows** *smc-Par*  $\alpha(\text{Obj}) = \text{Vset } \alpha$

**and** *smc-Par*  $\alpha(\text{Arr}) = \text{set } \{T. \text{arr-Par } \alpha T\}$

**and** *smc-Par*  $\alpha(\text{Dom}) = (\lambda T \in_{\circ} \text{set } \{T. \text{arr-Par } \alpha T\}. T(\text{ArrDom}))$

**and** *smc-Par*  $\alpha(\text{Cod}) = (\lambda T \in_{\circ} \text{set } \{T. \text{arr-Par } \alpha T\}. T(\text{ArrCod}))$

**and** *smc-Par*  $\alpha(\text{Comp}) = (\lambda ST \in_{\circ} \text{composable-arrs } (dg\text{-Par } \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}}))$

*<proof>*

Slicing.

**lemma** *smc-dg-smc-Par*: *smc-dg* (*smc-Par*  $\alpha$ ) = *dg-Par*  $\alpha$

*<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-vsuv* [*smc-Par-cs-intros*] = *dg-Par-Dom-vsuv*

**and** *smc-Par-Dom-vdomain* [*smc-Par-cs-simps*] = *dg-Par-Dom-vdomain*

**and** *smc-Par-Dom-vrange* = *dg-Par-Dom-vrange*

**and**  $smc\text{-}Par\text{-}Dom\text{-}app[smc\text{-}Par\text{-}cs\text{-}simps] = dg\text{-}Par\text{-}Dom\text{-}app$   
**and**  $smc\text{-}Par\text{-}Cod\text{-}vsu[smc\text{-}Par\text{-}cs\text{-}intros] = dg\text{-}Par\text{-}Cod\text{-}vsu$   
**and**  $smc\text{-}Par\text{-}Cod\text{-}vdomain[smc\text{-}Par\text{-}cs\text{-}simps] = dg\text{-}Par\text{-}Cod\text{-}vdomain$   
**and**  $smc\text{-}Par\text{-}Cod\text{-}vrangle = dg\text{-}Par\text{-}Cod\text{-}vrangle$   
**and**  $smc\text{-}Par\text{-}Cod\text{-}app[smc\text{-}Par\text{-}cs\text{-}simps] = dg\text{-}Par\text{-}Cod\text{-}app$   
**and**  $smc\text{-}Par\text{-}is\text{-}arrI = dg\text{-}Par\text{-}is\text{-}arrI$   
**and**  $smc\text{-}Par\text{-}is\text{-}arrD = dg\text{-}Par\text{-}is\text{-}arrD$   
**and**  $smc\text{-}Par\text{-}is\text{-}arrE = dg\text{-}Par\text{-}is\text{-}arrE$

**lemmas**  $[smc\text{-}cs\text{-}simps] = smc\text{-}Par\text{-}is\text{-}arrD(2,3)$

**lemmas**  $[smc\text{-}Par\text{-}cs\text{-}intros] = smc\text{-}Par\text{-}is\text{-}arrI$

**lemmas-with** (**in**  $\mathcal{Z}$ )  $[folded\ smc\text{-}dg\text{-}smc\text{-}Par, unfolded\ slicing\text{-}simps]:$

$smc\text{-}Par\text{-}Hom\text{-}vifunion\text{-}in\text{-}Vset = dg\text{-}Par\text{-}Hom\text{-}vifunion\text{-}in\text{-}Vset$   
**and**  $smc\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr = dg\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr$   
**and**  $smc\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr'[smc\text{-}Par\text{-}cs\text{-}intros] = dg\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr'$

**lemmas**  $[smc\text{-}Par\text{-}cs\text{-}intros] = \mathcal{Z}.smc\text{-}Par\text{-}incl\text{-}Par\text{-}is\text{-}arr'$

### Composable arrows

**lemma**  $smc\text{-}Par\text{-}composable\text{-}arrrs\text{-}dg\text{-}Par:$

$composable\text{-}arrrs\ (dg\text{-}Par\ \alpha) = composable\text{-}arrrs\ (smc\text{-}Par\ \alpha)$   
 $\langle proof \rangle$

**lemma**  $smc\text{-}Par\text{-}Comp:$

$smc\text{-}Par\ \alpha(\text{Comp}) = (\lambda ST \epsilon_o. composable\text{-}arrrs\ (smc\text{-}Par\ \alpha). ST(0) \circ_{Rel} ST(1_N))$   
 $\langle proof \rangle$

### Composition

**lemma**  $smc\text{-}Par\text{-}Comp\text{-}app[smc\text{-}Par\text{-}cs\text{-}simps]:$

**assumes**  $S : B \mapsto_{smc\text{-}Par\ \alpha} C$  **and**  $T : A \mapsto_{smc\text{-}Par\ \alpha} B$   
**shows**  $S \circ_{A\ smc\text{-}Par\ \alpha} T = S \circ_{Rel} T$

$\langle proof \rangle$

**lemma**  $smc\text{-}Par\text{-}Comp\text{-}vdomain: \mathcal{D}_o\ (smc\text{-}Par\ \alpha(\text{Comp})) = composable\text{-}arrrs\ (smc\text{-}Par\ \alpha)$

$\langle proof \rangle$

**lemma** (**in**  $\mathcal{Z}$ )  $smc\text{-}Par\text{-}Comp\text{-}vrangle: \mathcal{R}_o\ (smc\text{-}Par\ \alpha(\text{Comp})) \subseteq_o\ set\ \{T. arr\text{-}Par\ \alpha\ T\}$

$\langle proof \rangle$

### $Par$ is a semicategory

**lemma** (**in**  $\mathcal{Z}$ )  $semicategory\text{-}smc\text{-}Par: semicategory\ \alpha\ (smc\text{-}Par\ \alpha)$

$\langle proof \rangle$

### $Par$ is a wide subsemicategory of $Rel$

**lemma** (**in**  $\mathcal{Z}$ )  $wide\text{-}subsemicategory\text{-}smc\text{-}Par\text{-}smc\text{-}Rel:$

$smc\text{-}Par\ \alpha \subseteq_{SMC.wide\alpha} smc\text{-}Rel\ \alpha$

$\langle proof \rangle$

### 4.12.3 Monic arrow and epic arrow

**lemma**  $smc\text{-}Par\text{-}is\text{-}monic\text{-}arrI[intro]:$

**assumes**  $T : A \mapsto_{smc\text{-}Par\ \alpha} B$  **and**  $v11\ (T(\text{ArrVal}))$  **and**  $\mathcal{D}_o\ (T(\text{ArrVal})) = A$   
**shows**  $T : A \mapsto_{mon\ smc\text{-}Par\ \alpha} B$

*<proof>*

**lemma** *smc-Par-is-monic-arrD:*

**assumes**  $T : A \mapsto_{\text{mon smc-Par } \alpha} B$

**shows**  $T : A \mapsto_{\text{smc-Par } \alpha} B$  **and**  $v11 (T(\downarrow \text{ArrVal}))$  **and**  $\mathcal{D}_\circ (T(\downarrow \text{ArrVal})) = A$

*<proof>*

**lemma** *smc-Par-is-monic-arr:*

$T : A \mapsto_{\text{mon smc-Par } \alpha} B \longleftrightarrow$

$T : A \mapsto_{\text{smc-Par } \alpha} B \wedge v11 (T(\downarrow \text{ArrVal})) \wedge \mathcal{D}_\circ (T(\downarrow \text{ArrVal})) = A$

*<proof>*

**context**

**begin**

**private lemma** *smc-Par-is-epic-arr-vsubset:*

**assumes**  $T : A \mapsto_{\text{smc-Par } \alpha} B$

**and**  $\mathcal{R}_\circ (T(\downarrow \text{ArrVal})) = B$

**and**  $R : B \mapsto_{\text{smc-Par } \alpha} C$

**and**  $S : B \mapsto_{\text{smc-Par } \alpha} C$

**and**  $R \circ_{A \text{ smc-Par } \alpha} T = S \circ_{A \text{ smc-Par } \alpha} T$

**shows**  $R(\downarrow \text{ArrVal}) \subseteq_\circ S(\downarrow \text{ArrVal})$

*<proof>*

**lemma** *smc-Par-is-epic-arrI:*

**assumes**  $T : A \mapsto_{\text{smc-Par } \alpha} B$  **and**  $\mathcal{R}_\circ (T(\downarrow \text{ArrVal})) = B$

**shows**  $T : A \mapsto_{\text{epi smc-Par } \alpha} B$

*<proof>*

**lemma** *smc-Par-is-epic-arrD:*

**assumes**  $T : A \mapsto_{\text{epi smc-Par } \alpha} B$

**shows**  $T : A \mapsto_{\text{smc-Par } \alpha} B$  **and**  $\mathcal{R}_\circ (T(\downarrow \text{ArrVal})) = B$

*<proof>*

**end**

**lemma** *smc-Par-is-epic-arr:*

$T : A \mapsto_{\text{epi smc-Par } \alpha} B \longleftrightarrow T : A \mapsto_{\text{smc-Par } \alpha} B \wedge \mathcal{R}_\circ (T(\downarrow \text{ArrVal})) = B$

*<proof>*

#### 4.12.4 Terminal object, initial object and null object

**lemma** (in  $\mathcal{Z}$ ) *smc-Par-obj-terminal: obj-terminal (smc-Par  $\alpha$ )  $A \longleftrightarrow A = 0$*

*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *smc-Par-obj-initial: obj-initial (smc-Par  $\alpha$ )  $A \longleftrightarrow A = 0$*

*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *smc-Par-obj-terminal-obj-initial:*

*obj-initial (smc-Par  $\alpha$ )  $A \longleftrightarrow obj-terminal (smc-Par  $\alpha$ )  $A$$*

*<proof>*

**lemma** (in  $\mathcal{Z}$ ) *smc-Par-obj-null: obj-null (smc-Par  $\alpha$ )  $A \longleftrightarrow A = 0$*

*<proof>*

#### 4.12.5 Zero arrow

**lemma** (in  $\mathcal{Z}$ ) *smc-Par-is-zero-arr:*

**assumes**  $A \in_0 \text{smc-Par } \alpha(\text{Obj})$  **and**  $B \in_0 \text{smc-Par } \alpha(\text{Obj})$

**shows**  $T : A \mapsto_{0\text{smc-Par } \alpha} B \iff T = [0, A, B]$ .

*<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*  $\alpha$  =

[  
  *Vset*  $\alpha$ ,  
  *set* {*T*. *arr-Set*  $\alpha$  *T*},  
  ( $\lambda T \in_o \text{set } \{T. \text{arr-Set } \alpha T\}. T(\text{ArrDom})$ ),  
  ( $\lambda T \in_o \text{set } \{T. \text{arr-Set } \alpha T\}. T(\text{ArrCod})$ ),  
  ( $\lambda ST \in_o \text{composable-arrs } (dg\text{-Set } \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}})$ )  
  ].

Components.

**lemma** *smc-Set-components*:

**shows** *smc-Set*  $\alpha(\text{Obj}) = \text{Vset } \alpha$

**and** *smc-Set*  $\alpha(\text{Arr}) = \text{set } \{T. \text{arr-Set } \alpha T\}$

**and** *smc-Set*  $\alpha(\text{Dom}) = (\lambda T \in_o \text{set } \{T. \text{arr-Set } \alpha T\}. T(\text{ArrDom}))$

**and** *smc-Set*  $\alpha(\text{Cod}) = (\lambda T \in_o \text{set } \{T. \text{arr-Set } \alpha T\}. T(\text{ArrCod}))$

**and** *smc-Set*  $\alpha(\text{Comp}) = (\lambda ST \in_o \text{composable-arrs } (dg\text{-Set } \alpha). ST(0) \circ_{Rel} ST(1_{\mathbb{N}}))$

*<proof>*

Slicing.

**lemma** *smc-dg-smc-Set*: *smc-dg* (*smc-Set*  $\alpha$ ) = *dg-Set*  $\alpha$

*<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-vsuv*[*smc-Set-cs-intros*] = *dg-Set-Dom-vsuv*

**and** *smc-Set-Dom-vdomain*[*smc-Set-cs-simps*] = *dg-Set-Dom-vdomain*

**and** *smc-Set-Dom-vrange* = *dg-Set-Dom-vrange*

**and**  $smc\text{-Set-Dom-app}[smc\text{-Set-cs-simps}] = dg\text{-Set-Dom-app}$   
**and**  $smc\text{-Set-Cod-vsuv}[smc\text{-Set-cs-intros}] = dg\text{-Set-Cod-vsuv}$   
**and**  $smc\text{-Set-Cod-vdomain}[smc\text{-Set-cs-simps}] = dg\text{-Set-Cod-vdomain}$   
**and**  $smc\text{-Set-Cod-vrange} = dg\text{-Set-Cod-vrange}$   
**and**  $smc\text{-Set-Cod-app}[smc\text{-Set-cs-simps}] = dg\text{-Set-Cod-app}$   
**and**  $smc\text{-Set-is-arrI} = dg\text{-Set-is-arrI}$   
**and**  $smc\text{-Set-is-arrD} = dg\text{-Set-is-arrD}$   
**and**  $smc\text{-Set-is-arrE} = dg\text{-Set-is-arrE}$   
**and**  $smc\text{-Set-ArrVal-vdomain}[smc\text{-Set-cs-simps}] = dg\text{-Set-ArrVal-vdomain}$   
**and**  $smc\text{-Set-ArrVal-app-vrange}[smc\text{-Set-cs-intros}] = dg\text{-Set-ArrVal-app-vrange}$

**lemmas**  $[smc\text{-cs-simps}] = smc\text{-Set-is-arrD}(2,3)$

**lemmas-with** (**in**  $\mathcal{Z}$ ) [*folded smc-dg-smc-Set, unfolded slicing-simps*]:  
 $smc\text{-Set-Hom-vifunion-in-Vset} = dg\text{-Set-Hom-vifunion-in-Vset}$   
**and**  $smc\text{-Set-incl-Set-is-arr} = dg\text{-Set-incl-Set-is-arr}$

**lemmas**  $[smc\text{-Set-cs-intros}] =$   
 $smc\text{-Set-is-arrI}$

**lemma** (**in**  $\mathcal{Z}$ )  $smc\text{-Set-incl-Set-is-arr}'[smc\text{-cs-intros}, smc\text{-Set-cs-intros}]$ :  
**assumes**  $A \in_{\circ} smc\text{-Set } \alpha(|Obj|)$   
**and**  $B \in_{\circ} smc\text{-Set } \alpha(|Obj|)$   
**and**  $A \subseteq_{\circ} B$   
**and**  $A' = A$   
**and**  $B' = B$   
**and**  $\mathfrak{C}' = smc\text{-Set } \alpha$   
**shows**  $incl\text{-Set } A B : A' \mapsto_{\mathfrak{C}'} B'$   
*<proof>*

**lemmas**  $[smc\text{-Set-cs-intros}] = \mathcal{Z}.smc\text{-Set-incl-Set-is-arr}'$

### Composable arrows

**lemma**  $smc\text{-Set-composable-arrs-dg-Set}$ :  
 $composable\text{-arrs } (dg\text{-Set } \alpha) = composable\text{-arrs } (smc\text{-Set } \alpha)$   
*<proof>*

**lemma**  $smc\text{-Set-Comp}$ :  
 $smc\text{-Set } \alpha(|Comp|) =$   
 $VLambda (composable\text{-arrs } (smc\text{-Set } \alpha)) (\lambda ST. ST(|0|) \circ_{Rel} ST(|1_{\mathbb{N}}|))$   
*<proof>*

### Composition

**lemma**  $smc\text{-Set-Comp-app}[smc\text{-Set-cs-simps}]$ :  
**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**  $smc\text{-Set-Comp-vdomain}$ :  $\mathcal{D}_{\circ} (smc\text{-Set } \alpha(|Comp|)) = composable\text{-arrs } (smc\text{-Set } \alpha)$   
*<proof>*

**lemma** (**in**  $\mathcal{Z}$ )  $smc\text{-Set-Comp-vrange}$ :  
 $\mathcal{R}_{\circ} (smc\text{-Set } \alpha(|Comp|)) \subseteq_{\circ} set \{T. arr\text{-Set } \alpha T\}$   
*<proof>*

**lemma**  $smc\text{-Set-composable-vrange-vdomain}[smc\text{-Set-cs-intros}]$ :

**assumes**  $g : b \mapsto_{smc-Set \alpha} c$  **and**  $f : a \mapsto_{smc-Set \alpha} b$   
**shows**  $\mathcal{R}_\circ (f \downarrow ArrVal) \subseteq_\circ \mathcal{D}_\circ (g \downarrow ArrVal)$   
 ⟨proof⟩

**lemma** *smc-Set-Comp-ArrVal*[*smc-cs-simps*]:

**assumes**  $S : y \mapsto_{smc-Set \alpha} z$  **and**  $T : x \mapsto_{smc-Set \alpha} y$  **and**  $a \in_\circ x$   
**shows**  $(S \circ_{A smc-Set \alpha} T) \downarrow ArrVal \downarrow a = S \downarrow ArrVal \downarrow T \downarrow ArrVal \downarrow a$   
 ⟨proof⟩

*Set* is a semicategory

**lemma** (in  $\mathcal{Z}$ ) *semicategory-smc-Set*: *semicategory*  $\alpha$  (*smc-Set*  $\alpha$ )  
 ⟨proof⟩

*Set* is a wide subsemicategory of *Par*

**lemma** (in  $\mathcal{Z}$ ) *wide-subsemicategory-smc-Set-smc-Par*:  
 $smc-Set \alpha \subseteq_{SMC.wide\alpha} smc-Par \alpha$   
 ⟨proof⟩

### 4.13.3 Monic arrow and epic arrow

**lemma** *smc-Set-is-monic-arrI*:

— See Chapter I-5 in [39].

**assumes**  $T : A \mapsto_{smc-Set \alpha} B$  **and**  $v11 (T \downarrow ArrVal)$  **and**  $\mathcal{D}_\circ (T \downarrow ArrVal) = A$   
**shows**  $T : A \mapsto_{mon smc-Set \alpha} B$   
 ⟨proof⟩

**lemma** *smc-Set-is-monic-arrD*:

**assumes**  $T : A \mapsto_{mon smc-Set \alpha} B$   
**shows**  $T : A \mapsto_{smc-Set \alpha} B$  **and**  $v11 (T \downarrow ArrVal)$  **and**  $\mathcal{D}_\circ (T \downarrow ArrVal) = A$   
 ⟨proof⟩

**lemma** *smc-Set-is-monic-arr*:

$T : A \mapsto_{mon smc-Set \alpha} B \leftrightarrow$   
 $T : A \mapsto_{smc-Set \alpha} B \wedge v11 (T \downarrow ArrVal) \wedge \mathcal{D}_\circ (T \downarrow ArrVal) = A$   
 ⟨proof⟩

An epic arrow in *Set* is a total surjective function (see Chapter I-5 in [39]).

**lemma** *smc-Set-is-epic-arrI*:

**assumes**  $T : A \mapsto_{smc-Set \alpha} B$  **and**  $\mathcal{R}_\circ (T \downarrow ArrVal) = B$   
**shows**  $T : A \mapsto_{epi smc-Set \alpha} B$   
 ⟨proof⟩

**lemma** *smc-Set-is-epic-arrD*:

**assumes**  $T : A \mapsto_{epi smc-Set \alpha} B$   
**shows**  $T : A \mapsto_{smc-Set \alpha} B$  **and**  $\mathcal{R}_\circ (T \downarrow ArrVal) = B$   
 ⟨proof⟩

**lemma** *smc-Set-is-epic-arr*:

$T : A \mapsto_{epi smc-Set \alpha} B \leftrightarrow T : A \mapsto_{smc-Set \alpha} B \wedge \mathcal{R}_\circ (T \downarrow ArrVal) = B$   
 ⟨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*  $\alpha$ )  $A \leftrightarrow (\exists B \in_\circ Vset \alpha. A = set \{B\})$

*<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*  $\alpha$ )  $A \leftrightarrow A = 0$

*<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*  $\alpha$ )  $A \leftrightarrow \text{False}$

*<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*:  $T : A \mapsto_{0\text{smc-Set } \alpha} B \leftrightarrow \text{False}$

*<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*  $\alpha =$

[  
 set { $\mathcal{C}$ . digraph  $\alpha$   $\mathcal{C}$ },  
 all-dghms  $\alpha$ ,  
 ( $\lambda \mathfrak{F} \in_{\circ} \text{all-dghms } \alpha$ .  $\mathfrak{F}(\text{HomDom})$ ),  
 ( $\lambda \mathfrak{F} \in_{\circ} \text{all-dghms } \alpha$ .  $\mathfrak{F}(\text{HomCod})$ ),  
 ( $\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (dg\text{-GRPH } \alpha)$ .  $\mathfrak{G} \mathfrak{F}(\mathbb{0}) \circ_{DGHM} \mathfrak{G} \mathfrak{F}(\mathbb{1}_{\mathbb{N}})$ )  
 ]<sub>o</sub>.

Components.

**lemma** *smc-GRPH-components*:

**shows** *smc-GRPH*  $\alpha(\text{Obj}) = \text{set } \{\mathcal{C}$ . digraph  $\alpha$   $\mathcal{C}\}$   
**and** *smc-GRPH*  $\alpha(\text{Arr}) = \text{all-dghms } \alpha$   
**and** *smc-GRPH*  $\alpha(\text{Dom}) = (\lambda \mathfrak{F} \in_{\circ} \text{all-dghms } \alpha$ .  $\mathfrak{F}(\text{HomDom})$ )  
**and** *smc-GRPH*  $\alpha(\text{Cod}) = (\lambda \mathfrak{F} \in_{\circ} \text{all-dghms } \alpha$ .  $\mathfrak{F}(\text{HomCod})$ )  
**and** *smc-GRPH*  $\alpha(\text{Comp}) =$   
 ( $\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (dg\text{-GRPH } \alpha)$ .  $\mathfrak{G} \mathfrak{F}(\mathbb{0}) \circ_{DGHM} \mathfrak{G} \mathfrak{F}(\mathbb{1}_{\mathbb{N}})$ )  
 ⟨proof⟩

Slicing.

**lemma** *smc-dg-GRPH*: *smc-dg* (*smc-GRPH*  $\alpha$ ) = *dg-GRPH*  $\alpha$

⟨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*  $\alpha$ ) = *composable-arrs* (*smc-GRPH*  $\alpha$ )

⟨proof⟩

**lemma** *smc-GRPH-Comp*:

*smc-GRPH*  $\alpha(\text{Comp}) = (\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (smc\text{-GRPH } \alpha)$ .  $\mathfrak{G} \mathfrak{F}(\mathbb{0}) \circ_{DGHM} \mathfrak{G} \mathfrak{F}(\mathbb{1}_{\mathbb{N}})$ )

⟨proof⟩

#### 4.14.4 Composition

**lemma** *smc-GRPH-Comp-app*:

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{\text{smc-GRPH } \alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{\text{smc-GRPH } \alpha} \mathfrak{B}$   
**shows**  $\mathfrak{G} \circ_{A \text{ smc-GRPH } \alpha} \mathfrak{F} = \mathfrak{G} \circ_{DGHM} \mathfrak{F}$   
*<proof>*

**lemma** *smc-GRPH-Comp-vdomain*:

$\mathcal{D}_o(\text{smc-GRPH } \alpha \backslash \text{Comp}) = \text{composable-arrs}(\text{smc-GRPH } \alpha)$   
*<proof>*

#### 4.14.5 GRPH is a semicategory

**lemma** *(in Z) tiny-semicategory-smc-GRPH*:

**assumes**  $Z \beta$  **and**  $\alpha \in_o \beta$   
**shows** *tiny-semicategory*  $\beta$  *(smc-GRPH*  $\alpha$ *)*  
*<proof>*

#### 4.14.6 Initial object

**lemma** *(in Z) smc-GRPH-obj-initialI*: *obj-initial* *(smc-GRPH*  $\alpha$ *)* *dg-0*  
*<proof>*

**lemma** *(in Z) smc-GRPH-obj-initialD*:

**assumes** *obj-initial* *(smc-GRPH*  $\alpha$ *)*  $\mathfrak{A}$   
**shows**  $\mathfrak{A} = \text{dg-0}$   
*<proof>*

**lemma** *(in Z) smc-GRPH-obj-initialE*:

**assumes** *obj-initial* *(smc-GRPH*  $\alpha$ *)*  $\mathfrak{A}$   
**obtains**  $\mathfrak{A} = \text{dg-0}$   
*<proof>*

**lemma** *(in Z) smc-GRPH-obj-initial-iff[smc-GRPH-cs-simps]*:

*obj-initial* *(smc-GRPH*  $\alpha$ *)*  $\mathfrak{A} \longleftrightarrow \mathfrak{A} = \text{dg-0}$   
*<proof>*

#### 4.14.7 Terminal object

**lemma** *(in Z) smc-GRPH-obj-terminalI[smc-GRPH-cs-intros]*:

**assumes**  $a \in_o \text{Vset } \alpha$  **and**  $f \in_o \text{Vset } \alpha$   
**shows** *obj-terminal* *(smc-GRPH*  $\alpha$ *)* *(dg-1*  $a$   $f$ *)*  
*<proof>*

**lemma** *(in Z) smc-GRPH-obj-terminalE*:

**assumes** *obj-terminal* *(smc-GRPH*  $\alpha$ *)*  $\mathfrak{B}$   
**obtains**  $a$  **where**  $a \in_o \text{Vset } \alpha$  **and**  $f \in_o \text{Vset } \alpha$  **and**  $\mathfrak{B} = \text{dg-1 } a$   $f$   
*<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]<sup>8</sup>). 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,  $\alpha$ -*SemiCAT* is defined as a digraph of semicategories and semifunctors in  $V_\alpha$ .

**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*  $\alpha =$

[  
 set { $\mathfrak{C}$ . *semicategory*  $\alpha$   $\mathfrak{C}$ },  
 all-smcfs  $\alpha$ ,  
 ( $\lambda \mathfrak{F} \in_\circ \text{all-smcfs } \alpha. \mathfrak{F}(\text{HomDom})$ ),  
 ( $\lambda \mathfrak{F} \in_\circ \text{all-smcfs } \alpha. \mathfrak{F}(\text{HomCod})$ )  
 ]<sub>o</sub>

Components.

**lemma** *dg-SemiCAT-components*:

**shows** *dg-SemiCAT*  $\alpha(\text{Obj}) = \text{set } \{\mathfrak{C}. \text{semicategory } \alpha \mathfrak{C}\}$

**and** *dg-SemiCAT*  $\alpha(\text{Arr}) = \text{all-smcfs } \alpha$

**and** *dg-SemiCAT*  $\alpha(\text{Dom}) = (\lambda \mathfrak{F} \in_\circ \text{all-smcfs } \alpha. \mathfrak{F}(\text{HomDom}))$

**and** *dg-SemiCAT*  $\alpha(\text{Cod}) = (\lambda \mathfrak{F} \in_\circ \text{all-smcfs } \alpha. \mathfrak{F}(\text{HomCod}))$

*<proof>*

### 4.15.3 Object

**lemma** *dg-SemiCAT-ObjI*:

**assumes** *semicategory*  $\alpha \mathfrak{A}$

**shows**  $\mathfrak{A} \in_\circ \text{dg-SemiCAT } \alpha(\text{Obj})$

*<proof>*

**lemma** *dg-SemiCAT-ObjD*:

**assumes**  $\mathfrak{A} \in_\circ \text{dg-SemiCAT } \alpha(\text{Obj})$

**shows** *semicategory*  $\alpha \mathfrak{A}$

*<proof>*

**lemma** *dg-SemiCAT-ObjE*:

**assumes**  $\mathfrak{A} \in_\circ \text{dg-SemiCAT } \alpha(\text{Obj})$

**obtains** *semicategory*  $\alpha \mathfrak{A}$

*<proof>*

**lemma** *dg-SemiCAT-Obj-iff[ dg-SemiCAT-simps ]*:

$\mathfrak{A} \in_\circ \text{dg-SemiCAT } \alpha(\text{Obj}) \longleftrightarrow \text{semicategory } \alpha \mathfrak{A}$

*<proof>*

### 4.15.4 Domain and codomain

**lemma** [*dg-SemiCAT-simps*]:

**assumes**  $\mathfrak{F} \in_\circ \text{all-smcfs } \alpha$

**shows** *dg-SemiCAT-Dom-app*: *dg-SemiCAT*  $\alpha(\text{Dom})(\mathfrak{F}) = \mathfrak{F}(\text{HomDom})$

<sup>8</sup><https://ncatlab.org/nlab/show/semicategory>

and *dg-SemiCAT-Cod-app*:  $dg\text{-SemiCAT } \alpha(\text{Cod})(\mathfrak{F}) = \mathfrak{F}(\text{HomCod})$   
 ⟨proof⟩

#### 4.15.5 *SemiCAT* is a digraph

**lemma** (in  $\mathcal{Z}$ ) *tiny-digraph-dg-SemiCAT*:  
 assumes  $\mathcal{Z} \beta$  and  $\alpha \in_o \beta$   
 shows *tiny-digraph*  $\beta$  ( $dg\text{-SemiCAT } \alpha$ )  
 ⟨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  $\alpha$ -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*  $\alpha$  =

[  
 set { $\mathcal{C}$ . semicategory  $\alpha$   $\mathcal{C}$ },  
 all-smcfs  $\alpha$ ,  
 ( $\lambda \mathfrak{F} \in_{\circ} \text{all-smcfs } \alpha$ .  $\mathfrak{F}(\text{HomDom})$ ),  
 ( $\lambda \mathfrak{F} \in_{\circ} \text{all-smcfs } \alpha$ .  $\mathfrak{F}(\text{HomCod})$ ),  
 ( $\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (dg\text{-SemiCAT } \alpha)$ .  $\mathfrak{G}\mathfrak{F}(\{0\}) \circ_{SMCF} \mathfrak{G}\mathfrak{F}(\{1_N\})$ )  
 ]<sub>o</sub>.

Components.

**lemma** *smc-SemiCAT-components*:

**shows** *smc-SemiCAT*  $\alpha(\text{Obj})$  = set { $\mathcal{C}$ . semicategory  $\alpha$   $\mathcal{C}$ }

**and** *smc-SemiCAT*  $\alpha(\text{Arr})$  = all-smcfs  $\alpha$

**and** *smc-SemiCAT*  $\alpha(\text{Dom})$  = ( $\lambda \mathfrak{F} \in_{\circ} \text{all-smcfs } \alpha$ .  $\mathfrak{F}(\text{HomDom})$ )

**and** *smc-SemiCAT*  $\alpha(\text{Cod})$  = ( $\lambda \mathfrak{F} \in_{\circ} \text{all-smcfs } \alpha$ .  $\mathfrak{F}(\text{HomCod})$ )

**and** *smc-SemiCAT*  $\alpha(\text{Comp})$  =

( $\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (dg\text{-SemiCAT } \alpha)$ .  $\mathfrak{G}\mathfrak{F}(\{0\}) \circ_{SMCF} \mathfrak{G}\mathfrak{F}(\{1_N\})$ )

*<proof>*

Slicing.

**lemma** *smc-dg-SemiCAT[smc-SemiCAT-simps]*: *smc-dg* (*smc-SemiCAT*  $\alpha$ ) = *dg-SemiCAT*  $\alpha$

*<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*  $\alpha$ ) = *composable-arrs* (*smc-SemiCAT*  $\alpha$ )

*<proof>*

**lemma** *smc-SemiCAT-Comp*:

*smc-SemiCAT*  $\alpha(\text{Comp})$  =

( $\lambda \mathfrak{G} \mathfrak{F} \in_{\circ} \text{composable-arrs } (smc\text{-SemiCAT } \alpha)$ .  $\mathfrak{G}\mathfrak{F}(\{0\}) \circ_{DGHM} \mathfrak{G}\mathfrak{F}(\{1_N\})$ )

*<proof>*

#### 4.16.4 Composition

**lemma** *smc-SemiCAT-Comp-app*[*smc-SemiCAT-simps*]:

**assumes**  $\mathfrak{G} : \mathfrak{B} \mapsto_{\text{smc-SemiCAT } \alpha} \mathfrak{C}$  **and**  $\mathfrak{F} : \mathfrak{A} \mapsto_{\text{smc-SemiCAT } \alpha} \mathfrak{B}$

**shows**  $\mathfrak{G} \circ_A \text{smc-SemiCAT } \alpha \mathfrak{F} = \mathfrak{G} \circ_{SMCF} \mathfrak{F}$

*<proof>*

**lemma** *smc-SemiCAT-Comp-vdomain*[*smc-SemiCAT-simps*]:

$\mathcal{D}_\circ (\text{smc-SemiCAT } \alpha (\text{Comp})) = \text{composable-arrs } (\text{smc-SemiCAT } \alpha)$

*<proof>*

**lemma** *smc-SemiCAT-Comp-vrange*:  $\mathcal{R}_\circ (\text{smc-SemiCAT } \alpha (\text{Comp})) \sqsubseteq_\circ \text{all-smcfs } \alpha$

*<proof>*

#### 4.16.5 *SemiCAT* is a semicategory

**lemma** (*in*  $\mathcal{Z}$ ) *tiny-semicategory-smc-SemiCAT*:

**assumes**  $\mathcal{Z} \beta$  **and**  $\alpha \in_\circ \beta$

**shows** *tiny-semicategory*  $\beta$  (*smc-SemiCAT*  $\alpha$ )

*<proof>*

#### 4.16.6 Initial object

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-initialI*: *obj-initial* (*smc-SemiCAT*  $\alpha$ ) *smc-0*

*<proof>*

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-initialD*:

**assumes** *obj-initial* (*smc-SemiCAT*  $\alpha$ )  $\mathfrak{A}$

**shows**  $\mathfrak{A} = \text{smc-0}$

*<proof>*

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-initialE*:

**assumes** *obj-initial* (*smc-SemiCAT*  $\alpha$ )  $\mathfrak{A}$

**obtains**  $\mathfrak{A} = \text{smc-0}$

*<proof>*

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-initial-iff*[*smc-SemiCAT-simps*]:

*obj-initial* (*smc-SemiCAT*  $\alpha$ )  $\mathfrak{A} \longleftrightarrow \mathfrak{A} = \text{smc-0}$

*<proof>*

#### 4.16.7 Terminal object

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-terminalI*[*smc-SemiCAT-intros*]:

**assumes**  $a \in_\circ \text{Vset } \alpha$  **and**  $f \in_\circ \text{Vset } \alpha$

**shows** *obj-terminal* (*smc-SemiCAT*  $\alpha$ ) (*smc-1*  $a$   $f$ )

*<proof>*

**lemma** (*in*  $\mathcal{Z}$ ) *smc-SemiCAT-obj-terminalE*:

**assumes** *obj-terminal* (*smc-SemiCAT*  $\alpha$ )  $\mathfrak{B}$

**obtains**  $a$   $f$  **where**  $a \in_\circ \text{Vset } \alpha$  **and**  $f \in_\circ \text{Vset } \alpha$  **and**  $\mathfrak{B} = \text{smc-1 } a$   $f$

*<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](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](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. Categorical 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. C acamo 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. C acamo 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. Korn owicz. 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](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. Kamm uller, M. Wenzel, and L. C. Paulson. Locales A Sectioning Concept for Isabelle. In Y. Bertot, G. Dowek, L. Th ery, 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. Korn owicz. 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. Kornilowicz. The Composition of Functors and Transformations in Alternative Categories. *Formalized Mathematics*, 7(1):1–7, 1998.
- [38] A. Kornilowicz. 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.