

# Cubical Categories

Tanguy Massacrier and Georg Struth

March 19, 2025

## Abstract

This AFP entry formalises cubical  $\omega$ -categories and cubical  $\omega$ -categories with connections in the style of single-set categories. Cubical categories, and the cubical sets on which they are based, have their origins and main applications in algebraic topology. Applications in computer science include homotopy type theory, higher-dimensional automata in concurrency theory and higher-dimensional rewriting. The single-set axiomatisation, introduced in these components and a companion paper, allows a formalisation based on Isabelle’s type classes.

## Contents

<b>1</b>	<b>Introductory Remarks</b>	<b>1</b>
<b>2</b>	<b>Indexed Catoids</b>	<b>2</b>
2.1	Indexed catoids and categories . . . . .	2
<b>3</b>	<b>Cubical Categories</b>	<b>4</b>
3.1	Semi-cubical $\omega$ -categories . . . . .	4
3.2	Type classes for cubical $\omega$ -categories . . . . .	8
<b>4</b>	<b>Cubical Categories with Connections</b>	<b>28</b>
<b>5</b>	<b>Cubical <math>(\omega, 0)</math>-Categories with Connections</b>	<b>39</b>

## 1 Introductory Remarks

Based on a formalisation of catoids and single-set categories in the AFP [2] we develop single-set axiomatisations for cubical  $\omega$ -categories with and without connections. A detailed explanation of the single-set approach, the classical approach to cubical  $\omega$ -categories and the proof of equivalence of the single-set and the classical approach can be found in a companion article [1]. Isabelle, with its high degree of proof automation, has been instrumental for developing the single-set axioms introduced in this article.

## 2 Indexed Catoids

**theory** *ICatoids*  
**imports** *Catoids.Catoid*

**begin**

All categories considered in this component are single-set categories.

**no-notation** *src* ( $\langle\sigma\rangle$ )

**notation** *True* ( $\langle tt\rangle$ )

**notation** *False* ( $\langle ff\rangle$ )

**abbreviation** *Fix* :: ( $'a \Rightarrow 'a$ )  $\Rightarrow$   $'a$  set **where**  
*Fix*  $f \equiv \{x. f\ x = x\}$

First we lift locality to powersets.

**lemma** (in *local-catoid*) *locality-lifting*:  $(X \star Y \neq \{\}) = (Tgt\ X \cap Src\ Y \neq \{\})$

**proof**–

**have**  $(X \star Y \neq \{\}) = (\exists x\ y. x \in X \wedge y \in Y \wedge x \odot y \neq \{\})$

**by** (*metis (mono-tags, lifting) all-not-in-conv conv-exp2*)

**also have**  $\dots = (\exists x\ y. x \in X \wedge y \in Y \wedge tgt\ x = src\ y)$

**using** *local.st-local* **by** *auto*

**also have**  $\dots = (Tgt\ X \cap Src\ Y \neq \{\})$

**by** *blast*

**finally show** *?thesis*.

**qed**

The following lemma about functional catoids is useful in proofs.

**lemma** (in *functional-catoid*) *pcomp-def-var4*:  $\Delta\ x\ y \Longrightarrow x \odot y = \{x \otimes y\}$

**using** *local.pcomp-def-var3* **by** *blast*

### 2.1 Indexed catoids and categories

**class** *face-map-op* =

**fixes** *fmap* ::  $nat \Rightarrow bool \Rightarrow 'a \Rightarrow 'a$  ( $\langle\partial\rangle$ )

**begin**

**abbreviation** *Face* ::  $nat \Rightarrow bool \Rightarrow 'a$  set  $\Rightarrow 'a$  set ( $\langle\partial\partial\rangle$ ) **where**

$\partial\partial\ i\ \alpha \equiv image\ (\partial\ i\ \alpha)$

**abbreviation** *face-fix* ::  $nat \Rightarrow 'a$  set **where**

*face-fix*  $i \equiv Fix\ (\partial\ i\ ff)$

**abbreviation** *fFx*  $i\ x \equiv (\partial\ i\ ff\ x = x)$

**abbreviation** *FFx*  $i\ X \equiv (\forall x \in X. fFx\ i\ x)$

**end**

**class** *icomop-op* =  
  **fixes** *icomop* :: 'a  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  'a set ( $\leftarrow \odot \rightarrow$  [70,70,70]70)

**class** *imultisemigroup* = *icomop-op* +  
  **assumes** *iassoc*: ( $\bigcup v \in y \odot_i z. x \odot_i v$ ) = ( $\bigcup v \in x \odot_i y. v \odot_i z$ )

**begin**

**sublocale** *ims*: *multisemigroup*  $\lambda x y. x \odot_i y$   
  **by** *unfold-locales* (*simp add: local.iassoc*)

**abbreviation** *DD*  $\equiv$  *ims*. $\Delta$

**abbreviation** *iconv* :: 'a set  $\Rightarrow$  nat  $\Rightarrow$  'a set  $\Rightarrow$  'a set ( $\leftarrow \star \rightarrow$  [70,70,70]70) **where**  
   $X \star_i Y \equiv$  *ims.conv* *i* *X* *Y*

**end**

**class** *icatoid* = *imultisemigroup* + *face-map-op* +  
  **assumes** *iDst*: *DD* *i* *x* *y*  $\Longrightarrow$   $\partial$  *i* *tt* *x* =  $\partial$  *i* *ff* *y*  
  **and** *is-absorb* [*simp*]: ( $\partial$  *i* *ff* *x*)  $\odot_i$  *x* = {*x*}  
  **and** *it-absorb* [*simp*]: *x*  $\odot_i$  ( $\partial$  *i* *tt* *x*) = {*x*}

**begin**

Every indexed catoid is a catoid.

**sublocale** *icid*: *catoid*  $\lambda x y. x \odot_i y$   $\partial$  *i* *ff*  $\partial$  *i* *tt*  
  **by** *unfold-locales* (*simp-all add: iDst*)

**lemma** *lFace-Src*:  $\partial \partial$  *i* *ff* = *icid*.*Src* *i*  
  **by** *simp*

**lemma** *uFace-Tgt*:  $\partial \partial$  *i* *tt* = *icid*.*Tgt* *i*  
  **by** *simp*

**lemma** *face-fix-sfix*: *face-fix* = *icid*.*sfix*  
  **by** *force*

**lemma** *face-fix-tfix*: *face-fix* = *icid*.*tfix*  
  **using** *icid.stopp.stfix-set* **by** *presburger*

**lemma** *face-fix-prop* [*simp*]:  $x \in$  *face-fix* *i* = ( $\partial$  *i*  $\alpha$  *x* = *x*)  
  **by** (*smt* (*verit*, *del-insts*) *icid.stopp.st-fix mem-Collect-eq*)

**lemma** *fFx-prop*: *fFx* *i* *x* = ( $\partial$  *i*  $\alpha$  *x* = *x*)  
  **by** (*metis* *icid.st-eq1* *icid.st-eq2*)

**end**

**class** *icategory* = *icatoid* +  
**assumes** *locality*:  $\partial i \text{ tt } x = \partial i \text{ ff } y \implies DD i x y$   
**and** *functionality*:  $z \in x \odot_i y \implies z' \in x \odot_i y \implies z = z'$

**begin**

Every indexed category is a (single-set) category.

**sublocale** *icat*: *single-set-category*  $\lambda x y. x \odot_i y \partial i \text{ ff } \partial i \text{ tt}$   
**apply** *unfold-locales*  
**apply** (*simp add: local.functionality*)  
**apply** (*metis dual-order.eq-iff icid.src-local-cond icid.st-locality-locality local.locality*)  
**by** (*metis icid.st-locality-locality local.iDst local.locality order-refl*)

**abbreviation** *ipcomp* :: '*a*  $\Rightarrow$  *nat*  $\Rightarrow$  '*a*  $\Rightarrow$  '*a* ( $\langle \otimes \rangle$ [70,70,70]70) **where**  
 $x \otimes_i y \equiv \text{icat.pcomp } i x y$

**lemma** *iconv-prop*:  $X \star_i Y = \{x \otimes_i y \mid x \in X \wedge y \in Y \wedge DD i x y\}$   
**by** (*rule antisym*) (*clarsimp simp: ims.conv-def, metis local.icat.pcomp-def-var*)+

**abbreviation** *dim-bound*  $k x \equiv (\forall i. k \leq i \longrightarrow fFx i x)$

**abbreviation** *fin-dim*  $x \equiv (\exists k. \text{dim-bound } k x)$

**end**

**end**

### 3 Cubical Categories

**theory** *CubicalCategories*  
**imports** *ICatoids*

**begin**

All categories considered in this component are single-set categories.

#### 3.1 Semi-cubical $\omega$ -categories

We first define a class for cubical  $\omega$ -categories without symmetries.

**class** *semi-cubical-omega-category* = *icategory* +  
**assumes** *face-comm*:  $i \neq j \implies \partial i \alpha \circ \partial j \beta = \partial j \beta \circ \partial i \alpha$   
**and** *face-func*:  $i \neq j \implies DD j x y \implies \partial i \alpha (x \otimes_j y) = \partial i \alpha x \otimes_j \partial i \alpha y$   
**and** *interchange*:  $i \neq j \implies DD i w x \implies DD i y z \implies DD j w y \implies DD j x z$   
 $\implies (w \otimes_i x) \otimes_j (y \otimes_i z) = (w \otimes_j y) \otimes_i (x \otimes_j z)$   
**and** *fin-fix*:  $\exists k. \forall i. k \leq i \longrightarrow fFx i x$

**begin**

**lemma** *pcomp-face-func-DD*:  $i \neq j \implies DD\ j\ x\ y \implies DD\ j\ (\partial\ i\ \alpha\ x)\ (\partial\ i\ \alpha\ y)$   
**by** (*metis comp-apply icat.st-local local.face-comm*)

**lemma** *comp-face-func*:  $i \neq j \implies (\partial\ \partial\ i\ \alpha)\ (x \odot_j\ y) \subseteq \partial\ i\ \alpha\ x \odot_j\ \partial\ i\ \alpha\ y$   
**using** *local.icat.pcomp-def-var local.icat.pcomp-def-var4 local.face-func pcomp-face-func-DD*  
**by** *fastforce*

**lemma** *interchange-var*:

**assumes**  $i \neq j$

**and**  $(w \odot_i\ x) \star_j\ (y \odot_i\ z) \neq \{\}$

**and**  $(w \odot_j\ y) \star_i\ (x \odot_j\ z) \neq \{\}$

**shows**  $(w \odot_i\ x) \star_j\ (y \odot_i\ z) = (w \odot_j\ y) \star_i\ (x \odot_j\ z)$

**proof** –

**have**  $h1$ :  $DD\ i\ w\ x$

**using** *assms(2) local.ims.conv-def* **by** *force*

**have**  $h2$ :  $DD\ i\ y\ z$

**using** *assms(2) multimagma.conv-distl* **by** *force*

**have**  $h3$ :  $DD\ j\ w\ y$

**using** *assms(3) multimagma.conv-def* **by** *force*

**have**  $h4$ :  $DD\ j\ x\ z$

**using** *assms(3) local.icid.stopp.conv-def* **by** *force*

**have**  $(w \odot_i\ x) \star_j\ (y \odot_i\ z) = \{w \otimes_i\ x\} \star_j\ \{y \otimes_i\ z\}$

**using**  $h1\ h2$  *local.icat.pcomp-def-var4* **by** *force*

**also have**  $\dots = \{(w \otimes_i\ x) \otimes_j\ (y \otimes_i\ z)\}$

**using** *assms(2) calculation local.icat.pcomp-def-var4* **by** *force*

**also have**  $\dots = \{(w \otimes_j\ y) \otimes_i\ (x \otimes_j\ z)\}$

**by** (*simp add: assms(1) h1 h2 h3 h4 local.interchange*)

**also have**  $\dots = \{w \otimes_j\ y\} \star_i\ \{x \otimes_j\ z\}$

**by** (*metis assms(3) h3 h4 local.icat.pcomp-def-var4 multimagma.conv-atom*)

**also have**  $\dots = (w \odot_j\ y) \star_i\ (x \odot_j\ z)$

**using**  $h3\ h4$  *local.icat.pcomp-def-var4* **by** *force*

**finally show** *?thesis*.

**qed**

**lemma** *interchange-var2*:

**assumes**  $i \neq j$

**and**  $(\bigcup a \in w \odot_i\ x. \bigcup b \in y \odot_i\ z. a \odot_j\ b) \neq \{\}$

**and**  $(\bigcup c \in w \odot_j\ y. \bigcup d \in x \odot_j\ z. c \odot_i\ d) \neq \{\}$

**shows**  $(\bigcup a \in w \odot_i\ x. \bigcup b \in y \odot_i\ z. a \odot_j\ b) = (\bigcup c \in w \odot_j\ y. \bigcup d \in x \odot_j\ z. c \odot_i\ d)$

**proof** –

**have**  $\{(w \otimes_i\ x) \otimes_j\ (y \otimes_i\ z)\} = \{(w \otimes_j\ y) \otimes_i\ (x \otimes_j\ z)\}$

**using** *assms(1) assms(2) assms(3) local.interchange* **by** *fastforce*

**thus** *?thesis*

**by** (*metis assms(1) assms(2) assms(3) interchange-var multimagma.conv-def*)

**qed**

**lemma** *face-compat*:  $\partial i \alpha \circ \partial i \beta = \partial i \beta$   
**by** (*metis* (*full-types*, *lifting*) *ext comp-def*[*of*  $\partial i tt \partial i tt$ ] *comp-def*[*of*  $\partial i tt \partial i ff$ ]  
*i ff*]  
*comp-def*[*of*  $\partial i ff \partial i tt$ ] *comp-def*[*of*  $\partial i ff \partial i ff$ ] *icid.ts-compat*[*of* *i*]  
*local.icid.stopp.ts-compat*[*of* *i*])

**lemma** *face-compat-var* [*simp*]:  $\partial i \alpha (\partial i \beta x) = \partial i \beta x$   
**by** (*metis* (*full-types*) *icid.ts-compat* *local.icid.stopp.ts-compat*)

**lemma** *face-comm-var*:  $i \neq j \implies \partial i \alpha (\partial j \beta x) = \partial j \beta (\partial i \alpha x)$   
**by** (*meson* *comp-eq-dest* *local.face-comm*)

**lemma** *face-comm-lift*:  $i \neq j \implies \partial \partial i \alpha (\partial \partial j \beta X) = \partial \partial j \beta (\partial \partial i \alpha X)$   
**by** (*simp* *add: image-comp* *local.face-comm*)

**lemma** *face-func-lift*:  $i \neq j \implies (\partial \partial i \alpha) (X \star_j Y) \subseteq \partial \partial i \alpha X \star_j \partial \partial i \alpha Y$   
**using** *ims.conv-def* *comp-face-func* *dual-order.refl* *image-subset-iff* **by** *fastforce*

**lemma** *pcomp-lface*:  $DD i x y \implies \partial i ff (x \otimes_i y) = \partial i ff x$   
**by** (*simp* *add: icat.st-local* *local.icat.sscatml.locall-var*)

**lemma** *pcomp-uface*:  $DD i x y \implies \partial i tt (x \otimes_i y) = \partial i tt y$   
**using** *icat.st-local* *local.icat.sscatml.localr-var* **by** *force*

**lemma** *interchange-DD1*:

**assumes**  $i \neq j$   
**and**  $DD i w x$   
**and**  $DD i y z$   
**and**  $DD j w y$   
**and**  $DD j x z$   
**shows**  $DD j (w \otimes_i x) (y \otimes_i z)$

**proof**–

**have**  $a: \partial j tt (w \otimes_i x) = \partial j tt w \otimes_i \partial j tt x$   
**using** *assms*(1) *assms*(2) *face-func* **by** *simp*  
**also have**  $\dots = \partial j ff y \otimes_i \partial j ff z$   
**using** *assms*(4) *assms*(5) *local.iDst* **by** *simp*  
**also have**  $\dots = \partial j ff (y \otimes_i z)$   
**using** *assms*(1) *assms*(3) *face-func* **by** *simp*  
**finally show** *?thesis*  
**using** *local.locality* **by** *simp*

**qed**

**lemma** *interchange-DD2*:

**assumes**  $i \neq j$   
**and**  $DD i w x$   
**and**  $DD i y z$   
**and**  $DD j w y$   
**and**  $DD j x z$   
**shows**  $DD i (w \otimes_j y) (x \otimes_j z)$

using *assms interchange-DD1* by *simp*

**lemma** *face-idem1*:  $\partial i \alpha x = \partial i \beta y \implies \partial i \alpha x \odot_i \partial i \beta y = \{\partial i \alpha x\}$   
 by (*metis face-compat-var local.it-absorb*)

**lemma** *face-pidem1*:  $\partial i \alpha x = \partial i \beta y \implies \partial i \alpha x \otimes_i \partial i \beta y = \partial i \alpha x$   
 by (*metis face-compat-var local.icat.sscatml.l0-absorb*)

**lemma** *face-pidem2*:  $\partial i \alpha x \neq \partial i \beta y \implies \partial i \alpha x \odot_i \partial i \beta y = \{\}$   
 using *icat.st-local* by *force*

**lemma** *face-fix-comp-var*:  $i \neq j \implies \partial \partial i \alpha (\partial i \alpha x \odot_j \partial i \alpha y) = \partial i \alpha x \odot_j \partial i \alpha y$   
 by (*metis (mono-tags, lifting) comp-face-func empty-is-image face-compat-var local.icat.pcomp-def-var4 subset-singletonD*)

**lemma** *interchange-lift-aux*:  $x \in X \implies y \in Y \implies DD i x y \implies x \otimes_i y \in X \star_i Y$   
 using *local.icat.pcomp-def-var local.ims.conv-exp2* by *blast*

**lemma** *interchange-lift1*:

assumes  $i \neq j$

and  $\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. DD i w x \wedge DD i y z \wedge DD j w y \wedge DD j x z$

shows  $((W \star_i X) \star_j (Y \star_i Z)) \cap ((W \star_j Y) \star_i (X \star_j Z)) \neq \{\}$

**proof**–

obtain  $w x y z$  where  $h1: w \in W \wedge x \in X \wedge y \in Y \wedge z \in Z \wedge DD i w x \wedge DD i y z \wedge DD j w y \wedge DD j x z$

using *assms(2)* by *blast*

have  $h5: (w \otimes_i x) \otimes_j (y \otimes_i z) \in (W \star_i X) \star_j (Y \star_i Z)$

using *assms(1) h1 interchange-lift-aux interchange-DD2* by *presburger*

have  $(w \otimes_j y) \otimes_i (x \otimes_j z) \in (W \star_j Y) \star_i (X \star_j Z)$

by (*simp add: assms(1) h1 interchange-lift-aux interchange-DD2*)

thus *?thesis*

using *assms(1) h1 h5 local.interchange* by *fastforce*

qed

**lemma** *interchange-lift2*:

assumes  $i \neq j$

and  $\forall w \in W. \forall x \in X. \forall y \in Y. \forall z \in Z. DD i w x \wedge DD i y z \wedge DD j w y \wedge DD j x z$

shows  $((W \star_i X) \star_j (Y \star_i Z)) = ((W \star_j Y) \star_i (X \star_j Z))$

**proof**–

{fix  $t$

have  $(t \in (W \star_i X) \star_j (Y \star_i Z)) = (\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. DD i w x \wedge DD i y z \wedge DD j (w \otimes_i x) (y \otimes_i z) \wedge t = (w \otimes_i x) \otimes_j (y \otimes_i z))$

unfolding *iconv-prop* by *force*

also have  $\dots = (\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. DD i w x \wedge DD i y z \wedge DD j w y \wedge DD j x z \wedge t = (w \otimes_i x) \otimes_j (y \otimes_i z))$

using *assms(1) assms(2) interchange-DD2* by *simp*

**also have**  $\dots = (\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. DD\ j\ w\ y \wedge DD\ j\ x\ z \wedge DD\ j\ w\ y \wedge DD\ j\ x\ z \wedge t = (w \otimes_j y) \otimes_i (x \otimes_j z))$   
**by** (*simp add: assms(1) assms(2) local.interchange*)  
**also have**  $\dots = (\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. DD\ j\ w\ y \wedge DD\ j\ x\ z \wedge DD\ i\ (w \otimes_j y)\ (x \otimes_j z) \wedge t = (w \otimes_j y) \otimes_i (x \otimes_j z))$   
**using** *assms(1) assms(2) interchange-DD1* **by** *simp*  
**also have**  $\dots = (t \in (W \star_j Y) \star_i (X \star_j Z))$   
**unfolding** *iconv-prop* **by** *force*  
**finally have**  $(t \in (W \star_i X) \star_j (Y \star_i Z)) = (t \in (W \star_j Y) \star_i (X \star_j Z))$   
**by** *blast*  
**thus** *?thesis*  
**by** *force*  
**qed**

**lemma** *double-fix-prop*:  $(\partial\ i\ \alpha\ (\partial\ j\ \beta\ x) = x) = (fF\ x\ i\ x \wedge fF\ x\ j\ x)$   
**by** (*metis face-comm-var face-compat-var*)

**end**

### 3.2 Type classes for cubical $\omega$ -categories

**abbreviation** *diffSup* ::  $nat \Rightarrow nat \Rightarrow nat \Rightarrow bool$  **where**  
 $diffSup\ i\ j\ k \equiv (i - j \geq k \vee j - i \geq k)$

**class** *symmetry-ops* =  
**fixes** *symmetry* ::  $nat \Rightarrow 'a \Rightarrow 'a$  ( $\langle \sigma \rangle$ )  
**and** *inv-symmetry* ::  $nat \Rightarrow 'a \Rightarrow 'a$  ( $\langle \vartheta \rangle$ )

**begin**

**abbreviation**  $\sigma\sigma\ i \equiv image\ (\sigma\ i)$

**abbreviation**  $\vartheta\vartheta\ i \equiv image\ (\vartheta\ i)$

*symcomp*  $i\ j$  composes the symmetry maps from index  $i$  to index  $i+j-1$ .

**primrec** *symcomp* ::  $nat \Rightarrow nat \Rightarrow 'a \Rightarrow 'a$  ( $\langle \Sigma \rangle$ ) **where**  
 $\Sigma\ i\ 0\ x = x$   
 $|\ \Sigma\ i\ (Suc\ j)\ x = \sigma\ (i + j)\ (\Sigma\ i\ j\ x)$

*inv-symcomp*  $i\ j$  composes the inverse symmetries from  $i+j-1$  to  $i$ .

**primrec** *inv-symcomp* ::  $nat \Rightarrow nat \Rightarrow 'a \Rightarrow 'a$  ( $\langle \Theta \rangle$ ) **where**  
 $\Theta\ i\ 0\ x = x$   
 $|\ \Theta\ i\ (Suc\ j)\ x = \Theta\ i\ j\ (\vartheta\ (i + j)\ x)$

**end**

Next we define a class for cubical  $\omega$ -categories.

**class** *cubical-omega-category* = *semi-cubical-omega-category* + *symmetry-ops* +  
**assumes** *sym-type*:  $\sigma\sigma\ i\ (face\text{-}fix\ i) \subseteq face\text{-}fix\ (i + 1)$



**and** *inv-sym-type*:  $\vartheta \vartheta i (\text{face-fix } (i + 1)) \subseteq \text{face-fix } i$   
**and** *sym-inv-sym*:  $fFx (i + 1) x \implies \sigma i (\vartheta i x) = x$   
**and** *inv-sym-sym*:  $fFx i x \implies \vartheta i (\sigma i x) = x$   
**and** *sym-face1*:  $fFx i x \implies \partial i \alpha (\sigma i x) = \sigma i (\partial (i + 1) \alpha x)$   
**and** *sym-face2*:  $i \neq j \implies i \neq j + 1 \implies fFx j x \implies \partial i \alpha (\sigma j x) = \sigma j (\partial i \alpha x)$   
**and** *sym-func*:  $i \neq j \implies fFx i x \implies fFx i y \implies DD j x y \implies \sigma i (x \otimes_j y) = (\text{if } j = i + 1 \text{ then } \sigma i x \otimes_i \sigma i y \text{ else } \sigma i x \otimes_j \sigma i y)$   
**and** *sym-fix*:  $fFx i x \implies fFx (i + 1) x \implies \sigma i x = x$   
**and** *sym-sym-braid*:  $\text{diffSup } i j 2 \implies fFx i x \implies fFx j x \implies \sigma i (\sigma j x) = \sigma j (\sigma i x)$

**begin**

First we prove variants of the axioms.

**lemma** *sym-type-var*:  $fFx i x \implies fFx (i + 1) (\sigma i x)$   
**by** (*meson image-subset-iff local.face-fix-prop local.sym-type*)

**lemma** *sym-type-var1* [*simp*]:  $\partial (i + 1) \alpha (\sigma i (\partial i \alpha x)) = \sigma i (\partial i \alpha x)$   
**by** (*metis local.face-compat-var sym-type-var*)

**lemma** *sym-type-var2* [*simp*]:  $\partial (i + 1) \alpha \circ \sigma i \circ \partial i \alpha = \sigma i \circ \partial i \alpha$   
**unfolding** *comp-def fun-eq-iff* **using** *sym-type-var1* **by** *simp*

**lemma** *sym-type-var-lift-var* [*simp*]:  $\partial \partial (i + 1) \alpha (\sigma \sigma i (\partial \partial i \alpha X)) = \sigma \sigma i (\partial \partial i \alpha X)$   
**by** (*metis image-comp sym-type-var2*)

**lemma** *sym-type-var-lift* [*simp*]:  
**assumes**  $fFx i X$   
**shows**  $\partial \partial (i + 1) \alpha (\sigma \sigma i X) = \sigma \sigma i X$   
**proof** –  
**have**  $\partial \partial (i + 1) \alpha (\sigma \sigma i X) = \{\partial (i + 1) \alpha (\sigma i x) \mid x. x \in X\}$   
**by** *blast*  
**also have**  $\dots = \{\sigma i x \mid x. x \in X\}$   
**by** (*metis assms local.fFx-prop sym-type-var*)  
**also have**  $\dots = \sigma \sigma i X$   
**by** (*simp add: setcompr-eq-image*)  
**finally show** *?thesis*.  
**qed**

**lemma** *inv-sym-type-var*:  $fFx (i + 1) x \implies fFx i (\vartheta i x)$   
**by** (*meson image-subset-iff local.face-fix-prop local.inv-sym-type*)

**lemma** *inv-sym-type-var1* [*simp*]:  $\partial i \alpha (\vartheta i (\partial (i + 1) \alpha x)) = \vartheta i (\partial (i + 1) \alpha x)$   
**by** (*metis inv-sym-type-var local.face-compat-var*)

**lemma** *inv-sym-type-var2* [*simp*]:  $\partial i \alpha \circ \vartheta i \circ \partial (i + 1) \alpha = \vartheta i \circ \partial (i + 1) \alpha$

**unfolding** *comp-def fun-eq-iff* **using** *inv-sym-type-var1* **by** *simp*

**lemma** *inv-sym-type-lift-var* [*simp*]:  $\partial \partial i \alpha (\vartheta \vartheta i (\partial \partial (i + 1) \alpha X)) = \vartheta \vartheta i (\partial \partial (i + 1) \alpha X)$   
**by** (*metis image-comp inv-sym-type-var2*)

**lemma** *inv-sym-type-lift*:  
**assumes**  $FFx (i + 1) X$   
**shows**  $\partial \partial i \alpha (\vartheta \vartheta i X) = \vartheta \vartheta i X$   
**by** (*smt (verit, best) assms image-cong image-ident inv-sym-type-lift-var local.icid.stopp.ST-compat*)

**lemma** *sym-inv-sym-var1* [*simp*]:  $\sigma i (\vartheta i (\partial (i + 1) \alpha x)) = \partial (i + 1) \alpha x$   
**by** (*simp add: local.sym-inv-sym*)

**lemma** *sym-inv-sym-var2* [*simp*]:  $\sigma i \circ \vartheta i \circ \partial (i + 1) \alpha = \partial (i + 1) \alpha$   
**unfolding** *comp-def fun-eq-iff* **using** *sym-inv-sym-var1* **by** *simp*

**lemma** *sym-inv-sym-lift-var*:  $\sigma \sigma i (\vartheta \vartheta i (\partial \partial (i + 1) \alpha X)) = \partial \partial (i + 1) \alpha X$   
**by** (*metis image-comp sym-inv-sym-var2*)

**lemma** *sym-inv-sym-lift*:  
**assumes**  $FFx (i + 1) X$   
**shows**  $\sigma \sigma i (\vartheta \vartheta i X) = X$   
**proof** –  
**have**  $\sigma \sigma i (\vartheta \vartheta i X) = \{\sigma i (\vartheta i x) \mid x. x \in X\}$   
**by** *blast*  
**thus** *?thesis*  
**using** *assms local.sym-inv-sym* **by** *force*  
**qed**

**lemma** *inv-sym-sym-var1* [*simp*]:  $\vartheta i (\sigma i (\partial i \alpha x)) = \partial i \alpha x$   
**by** (*simp add: local.inv-sym-sym*)

**lemma** *inv-sym-sym-var2* [*simp*]:  $\vartheta i \circ \sigma i \circ \partial i \alpha = \partial i \alpha$   
**unfolding** *comp-def fun-eq-iff* **by** *simp*

**lemma** *inv-sym-sym-lift-var* [*simp*]:  $\vartheta \vartheta i (\sigma \sigma i (\partial \partial i \alpha X)) = \partial \partial i \alpha X$   
**by** (*simp add: image-comp*)

**lemma** *inv-sym-sym-lift*:  
**assumes**  $FFx i X$   
**shows**  $\vartheta \vartheta i (\sigma \sigma i X) = X$   
**by** (*metis assms image-cong image-ident inv-sym-sym-lift-var*)

**lemma** *sym-fix-var1* [*simp*]:  $\sigma i (\partial i \alpha (\partial (i + 1) \beta x)) = \partial i \alpha (\partial (i + 1) \beta x)$   
**by** (*simp add: local.face-comm-var local.sym-fix*)

**lemma** *sym-fix-var2* [*simp*]:  $\sigma i \circ \partial i \alpha \circ \partial (i + 1) \beta = \partial i \alpha \circ \partial (i + 1) \beta$

**unfolding** *comp-def fun-eq-iff* **using** *sym-fix-var1* **by** *simp*

**lemma** *sym-fix-lift-var*:  $\sigma \sigma i (\partial \partial i \alpha (\partial \partial (i + 1) \beta X)) = \partial \partial i \alpha (\partial \partial (i + 1) \beta X)$   
**by** (*metis image-comp sym-fix-var2*)

**lemma** *sym-fix-lift*:  
**assumes**  $FFx i X$   
**and**  $FFx (i + 1) X$   
**shows**  $\sigma \sigma i X = X$   
**using** *assms local.sym-fix* **by** *simp*

**lemma** *sym-face1-var1*:  $\partial i \alpha (\sigma i (\partial i \beta x)) = \sigma i (\partial (i + 1) \alpha (\partial i \beta x))$   
**by** (*simp add: local.sym-face1*)

**lemma** *sym-face1-var2*:  $\partial i \alpha \circ \sigma i \circ \partial i \beta = \sigma i \circ \partial (i + 1) \alpha \circ \partial i \beta$   
**by** (*simp add: comp-def local.sym-face1*)

**lemma** *sym-face1-lift-var*:  $\partial \partial i \alpha (\sigma \sigma i (\partial \partial i \beta X)) = \sigma \sigma i (\partial \partial (i + 1) \alpha (\partial \partial i \beta X))$   
**by** (*metis image-comp sym-face1-var2*)

**lemma** *sym-face1-lift*:  
**assumes**  $FFx i X$   
**shows**  $\partial \partial i \alpha (\sigma \sigma i X) = \sigma \sigma i (\partial \partial (i + 1) \alpha X)$   
**by** (*metis (lifting) ext assms inv-sym-sym-lift inv-sym-type-lift-var sym-face1-lift-var sym-type-var-lift*)

**lemma** *sym-face2-var1*:  
**assumes**  $i \neq j$   
**and**  $i \neq j + 1$   
**shows**  $\partial i \alpha (\sigma j (\partial j \beta x)) = \sigma j (\partial i \alpha (\partial j \beta x))$   
**using** *assms local.sym-face2* **by** *simp*

**lemma** *sym-face2-var2*:  
**assumes**  $i \neq j$   
**and**  $i \neq j + 1$   
**shows**  $\partial i \alpha \circ \sigma j \circ \partial j \beta = \sigma j \circ \partial i \alpha \circ \partial j \beta$   
**unfolding** *comp-def fun-eq-iff* **using** *assms sym-face2-var1* **by** *simp*

**lemma** *sym-face2-lift-var*:  
**assumes**  $i \neq j$   
**and**  $i \neq j + 1$   
**shows**  $\partial \partial i \alpha (\sigma \sigma j (\partial \partial j \beta X)) = \sigma \sigma j (\partial \partial i \alpha (\partial \partial j \beta X))$   
**by** (*metis assms image-comp sym-face2-var2*)

**lemma** *sym-face2-lift*:  
**assumes**  $i \neq j$   
**and**  $i \neq j + 1$

**and**  $FFx\ j\ X$   
**shows**  $\partial\partial\ i\ \alpha\ (\sigma\sigma\ j\ X) = \sigma\sigma\ j\ (\partial\partial\ i\ \alpha\ X)$   
**by** (*smt* (*z3*) *assms image-cong image-image sym-face2-var1*)

**lemma** *sym-sym-braid-var1*:  
**assumes** *diffSup i j 2*  
**shows**  $\sigma\ i\ (\sigma\ j\ (\partial\ i\ \alpha\ (\partial\ j\ \beta\ x))) = \sigma\ j\ (\sigma\ i\ (\partial\ i\ \alpha\ (\partial\ j\ \beta\ x)))$   
**using** *assms local.face-comm-var local.sym-sym-braid* **by** *force*

**lemma** *sym-sym-braid-var2*:  
**assumes** *diffSup i j 2*  
**shows**  $\sigma\ i\ \circ\ \sigma\ j\ \circ\ \partial\ i\ \alpha\ \circ\ \partial\ j\ \beta = \sigma\ j\ \circ\ \sigma\ i\ \circ\ \partial\ i\ \alpha\ \circ\ \partial\ j\ \beta$   
**using** *assms sym-sym-braid-var1* **by** *fastforce*

**lemma** *sym-sym-braid-lift-var*:  
**assumes** *diffSup i j 2*  
**shows**  $\sigma\sigma\ i\ (\sigma\sigma\ j\ (\partial\partial\ i\ \alpha\ (\partial\partial\ j\ \beta\ X))) = \sigma\sigma\ j\ (\sigma\sigma\ i\ (\partial\partial\ i\ \alpha\ (\partial\partial\ j\ \beta\ X)))$

**proof** –

**have**  $\sigma\sigma\ i\ (\sigma\sigma\ j\ (\partial\partial\ i\ \alpha\ (\partial\partial\ j\ \beta\ X))) = \{\sigma\ i\ (\sigma\ j\ (\partial\ i\ \alpha\ (\partial\ j\ \beta\ x))) \mid x. x \in X\}$

**by** *blast*

**also have**  $\dots = \{\sigma\ j\ (\sigma\ i\ (\partial\ i\ \alpha\ (\partial\ j\ \beta\ x))) \mid x. x \in X\}$

**by** (*metis* (*full-types, opaque-lifting*) *assms sym-sym-braid-var1*)

**finally show** *?thesis*

**by** (*simp add: Setcompr-eq-image image-image*)

**qed**

**lemma** *sym-sym-braid-lift*:  
**assumes** *diffSup i j 2*  
**and**  $FFx\ i\ X$   
**and**  $FFx\ j\ X$   
**shows**  $\sigma\sigma\ i\ (\sigma\sigma\ j\ X) = \sigma\sigma\ j\ (\sigma\sigma\ i\ X)$   
**by** (*smt* (*verit, best*) *assms(1,2,3) comp-apply image-comp image-cong local.sym-sym-braid*)

**lemma** *sym-func2*:  
**assumes**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ (i + 1)\ x\ y$   
**shows**  $\sigma\ i\ (x \otimes_{(i+1)} y) = \sigma\ i\ x \otimes_i \sigma\ i\ y$   
**using** *assms local.sym-func* **by** *simp*

**lemma** *sym-func3*:  
**assumes**  $i \neq j$   
**and**  $j \neq i + 1$   
**and**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ j\ x\ y$   
**shows**  $\sigma\ i\ (x \otimes_j y) = \sigma\ i\ x \otimes_j \sigma\ i\ y$   
**using** *assms local.sym-func* **by** *simp*

**lemma** *sym-func2-var1*:

**assumes**  $DD (i + 1) (\partial i \alpha x) (\partial i \beta y)$   
**shows**  $\sigma i (\partial i \alpha x \otimes_{(i+1)} \partial i \beta y) = \sigma i (\partial i \alpha x) \otimes_i \sigma i (\partial i \beta y)$   
**using** *assms local.face-compat-var local.sym-func2* **by** *simp*

**lemma** *sym-func3-var1*:

**assumes**  $i \neq j$   
**and**  $j \neq i + 1$   
**and**  $DD j (\partial i \alpha x) (\partial i \beta y)$   
**shows**  $\sigma i (\partial i \alpha x \otimes_j \partial i \beta y) = \sigma i (\partial i \alpha x) \otimes_j \sigma i (\partial i \beta y)$   
**using** *assms local.face-compat-var local.sym-func3* **by** *simp*

**lemma** *sym-func2-DD*:

**assumes**  $fFx i x$   
**and**  $fFx i y$   
**shows**  $DD (i + 1) x y = DD i (\sigma i x) (\sigma i y)$   
**by** (*metis assms icat.st-local local.face-comm-var local.sym-face1 sym-fix-var1*)

**lemma** *func2-var2*:  $\sigma \sigma i (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = \sigma i (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y)$

**proof** (*cases DD (i + 1) (\partial i \alpha x) (\partial i \beta y)*)

**case** *True*

**have**  $\sigma \sigma i (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = \sigma \sigma i \{ \partial i \alpha x \otimes_{(i+1)} \partial i \beta y \}$

**using** *True local.icat.pcomp-def-var4* **by** *simp*

**also have**  $\dots = \{ \sigma i (\partial i \alpha x \otimes_{(i+1)} \partial i \beta y) \}$

**by** *simp*

**also have**  $\dots = \{ \sigma i (\partial i \alpha x) \otimes_i \sigma i (\partial i \beta y) \}$

**using** *True sym-func2-var1* **by** *simp*

**also have**  $\dots = \sigma i (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y)$

**using** *True local.icat.pcomp-def-var4 sym-func2-DD* **by** *simp*

**finally show** *?thesis*.

**next**

**case** *False*

**thus** *?thesis*

**using** *local.sym-func2-DD* **by** *simp*

**qed**

**lemma** *sym-func2-lift-var1*:  $\sigma \sigma i (\partial \partial i \alpha X \star_{(i+1)} \partial \partial i \beta Y) = \sigma \sigma i (\partial \partial i \alpha X) \star_i \sigma \sigma i (\partial \partial i \beta Y)$

**proof** –

**have**  $\sigma \sigma i (\partial \partial i \alpha X \star_{(i+1)} \partial \partial i \beta Y) = \sigma \sigma i \{ x \otimes_{(i+1)} y \mid x y. x \in \partial \partial i \alpha X \wedge y \in \partial \partial i \beta Y \wedge DD (i + 1) x y \}$

**using** *local.iconv-prop* **by** *presburger*

**also have**  $\dots = \{ \sigma i (\partial i \alpha x \otimes_{(i+1)} \partial i \beta y) \mid x y. x \in X \wedge y \in Y \wedge DD (i + 1) (\partial i \alpha x) (\partial i \beta y) \}$

**by** *blast*

**also have**  $\dots = \{ \sigma i (\partial i \alpha x) \otimes_i \sigma i (\partial i \beta y) \mid x y. x \in X \wedge y \in Y \wedge DD i (\sigma i (\partial i \alpha x)) (\sigma i (\partial i \beta y)) \}$

**using** *func2-var2 sym-func2-var1* **by** *fastforce*  
**also have**  $\dots = \{x \otimes_i y \mid x y. x \in \sigma \sigma i (\partial \partial i \alpha X) \wedge y \in \sigma \sigma i (\partial \partial i \beta Y) \wedge DD i x y\}$   
**by** *blast*  
**also have**  $\dots = \sigma \sigma i (\partial \partial i \alpha X) \star_i \sigma \sigma i (\partial \partial i \beta Y)$   
**using** *local.iconv-prop* **by** *presburger*  
**finally show** *?thesis*.  
**qed**

**lemma** *sym-func2-lift*:

**assumes** *FFx i X*

**and** *FFx i Y*

**shows**  $\sigma \sigma i (X \star_{(i+1)} Y) = \sigma \sigma i X \star_i \sigma \sigma i Y$

**proof** –

**have**  $\sigma \sigma i (X \star_{(i+1)} Y) = \sigma \sigma i (\partial \partial i tt X \star_{(i+1)} \partial \partial i tt Y)$

**by** (*smt (verit) assms image-cong image-ident local.icid.stopp.ST-compat*)

**also have**  $\dots = \sigma \sigma i (\partial \partial i tt X) \star_i \sigma \sigma i (\partial \partial i tt Y)$

**using** *sym-func2-lift-var1* **by** *simp*

**also have**  $\dots = \sigma \sigma i X \star_i \sigma \sigma i Y$

**using** *assms icid.st-eq1* **by** *simp*

**finally show** *?thesis*.

**qed**

**lemma** *func3-var1*:

**assumes**  $i \neq j$

**and**  $j \neq i + 1$

**shows**  $\sigma \sigma i (\partial i \alpha x \odot_j \partial i \beta y) = \sigma i (\partial i \alpha x) \odot_j \sigma i (\partial i \beta y)$

**proof** (*cases DD j (\partial i \alpha x) (\partial i \beta y)*)

**case** *True*

**have**  $\sigma \sigma i (\partial i \alpha x \odot_j \partial i \beta y) = \sigma \sigma i \{\partial i \alpha x \otimes_j \partial i \beta y\}$

**using** *True local.icat.pcomp-def-var4* **by** *simp*

**also have**  $\dots = \{\sigma i (\partial i \alpha x \otimes_j \partial i \beta y)\}$

**by** *simp*

**also have**  $\dots = \{\sigma i (\partial i \alpha x) \otimes_j \sigma i (\partial i \beta y)\}$

**using** *True assms sym-func3-var1* **by** *simp*

**also have**  $\dots = \sigma i (\partial i \alpha x) \odot_j \sigma i (\partial i \beta y)$

**using** *True assms icat.st-local local.icat.pcomp-def-var4 sym-face2-var1* **by** *simp*

**finally show** *?thesis*.

**next**

**case** *False*

**thus** *?thesis*

**by** (*metis assms empty-is-image icat.st-local inv-sym-sym-var1 local.face-comm-var sym-face2-var1*)

**qed**

**lemma** *sym-func3-lift-var1*:

**assumes**  $i \neq j$

**and**  $j \neq i + 1$

**shows**  $\sigma \sigma i (\partial \partial i \alpha X \star_j \partial \partial i \beta Y) = \sigma \sigma i (\partial \partial i \alpha X) \star_j \sigma \sigma i (\partial \partial i \beta Y)$

**proof**–

**have**  $\sigma \sigma i (\partial \partial i \alpha X \star_j \partial \partial i \beta Y) = \sigma \sigma i \{x \otimes_j y \mid x y. x \in \partial \partial i \alpha X \wedge y \in \partial \partial i \beta Y \wedge DD j x y\}$

**using** *local.iconv-prop* **by** *presburger*

**also have**  $\dots = \{\sigma i (\partial i \alpha x \otimes_j \partial i \beta y) \mid x y. x \in X \wedge y \in Y \wedge DD j (\partial i \alpha x) (\partial i \beta y)\}$

**by** *force*

**also have**  $\dots = \{\sigma i (\partial i \alpha x) \otimes_j \sigma i (\partial i \beta y) \mid x y. x \in X \wedge y \in Y \wedge DD j (\sigma i (\partial i \alpha x)) (\sigma i (\partial i \beta y))\}$

**using** *assms func3-var1 sym-func3-var1* **by** *fastforce*

**also have**  $\dots = \{x \otimes_j y \mid x y. x \in \sigma \sigma i (\partial \partial i \alpha X) \wedge y \in \sigma \sigma i (\partial \partial i \beta Y) \wedge DD j x y\}$

**by** *force*

**also have**  $\dots = \sigma \sigma i (\partial \partial i \alpha X) \star_j \sigma \sigma i (\partial \partial i \beta Y)$

**using** *local.iconv-prop* **by** *presburger*

**finally show** *?thesis*.

**qed**

**lemma** *sym-func3-lift*:

**assumes**  $i \neq j$

**and**  $j \neq i + 1$

**and**  $FFx i X$

**and**  $FFx i Y$

**shows**  $\sigma \sigma i (X \star_j Y) = \sigma \sigma i X \star_j \sigma \sigma i Y$

**proof**–

**have**  $\sigma \sigma i (X \star_j Y) = \sigma \sigma i (\partial \partial i tt X \star_j \partial \partial i tt Y)$

**by** (*smt (verit) assms(3) assms(4) image-cong image-ident local.icid.stopp.ST-compat*)

**also have**  $\dots = \sigma \sigma i (\partial \partial i tt X) \star_j \sigma \sigma i (\partial \partial i tt Y)$

**using** *assms(1) assms(2) sym-func3-lift-var1* **by** *presburger*

**also have**  $\dots = \sigma \sigma i X \star_j \sigma \sigma i Y$

**using** *assms(3) assms(4) icid.st-eq1* **by** *simp*

**finally show** *?thesis*.

**qed**

**lemma** *sym-func3-var2*:  $i \neq j \implies \sigma \sigma i (\partial i \alpha x \odot_j \partial i \beta y) = (\text{if } j = i + 1 \text{ then } \sigma i (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y) \text{ else } \sigma i (\partial i \alpha x) \odot_j \sigma i (\partial i \beta y))$

**using** *func2-var2 func3-var1* **by** *simp*

Symmetries and inverse symmetries form a bijective pair on suitable fix-points of the face maps.

**lemma** *sym-inj*: *inj-on*  $(\sigma i)$  (*face-fix i*)

**by** (*smt (verit, del-insts) CollectD inj-onI local.inv-sym-sym*)

**lemma** *sym-inj-var*:

**assumes**  $fFx i x$

**and**  $fFx i y$

**and**  $\sigma i x = \sigma i y$

**shows**  $x = y$

**by** (*metis assms inv-sym-sym-var1*)

**lemma** *inv-sym-inj*:  $\text{inj-on } (\vartheta \ i) \ (\text{face-fix } (i + 1))$   
**by** (*smt* (*verit*, *del-insts*) *CollectD inj-onI local.sym-inv-sym*)

**lemma** *inv-sym-inj-var*:  
**assumes**  $fFx \ (i + 1) \ x$   
**and**  $fFx \ (i + 1) \ y$   
**and**  $\vartheta \ i \ x = \vartheta \ i \ y$   
**shows**  $x = y$   
**by** (*metis assms local.sym-inv-sym*)

**lemma** *surj-sym*:  $\text{image } (\sigma \ i) \ (\text{face-fix } i) = \text{face-fix } (i + 1)$   
**by** (*safe*, *metis sym-type-var1*, *smt* (*verit*, *del-insts*) *imageI inv-sym-type-var1 mem-Collect-eq sym-inv-sym-var1*)

**lemma** *surj-inv-sym*:  $\text{image } (\vartheta \ i) \ (\text{face-fix } (i + 1)) = \text{face-fix } i$   
**by** (*safe*, *metis inv-sym-type-var1*, *smt* (*verit*, *del-insts*) *imageI inv-sym-sym-var1 mem-Collect-eq sym-type-var1*)

**lemma** *sym-adj*:  
**assumes**  $fFx \ i \ x$   
**and**  $fFx \ (i + 1) \ y$   
**shows**  $(\sigma \ i \ x = y) = (x = \vartheta \ i \ y)$   
**using** *assms local.inv-sym-sym local.sym-inv-sym* **by** *force*

Next we list properties for inverse symmetries corresponding to the axioms.

**lemma** *inv-sym*:  
**assumes**  $fFx \ i \ x$   
**and**  $fFx \ (i + 1) \ x$   
**shows**  $\vartheta \ i \ x = x$   
**proof** –  
**have**  $x = \sigma \ i \ x$   
**using** *assms local.sym-fix* **by** *simp*  
**thus** *?thesis*  
**using** *assms sym-adj* **by** *force*  
**qed**

**lemma** *inv-sym-face2*:  
**assumes**  $i \neq j$   
**and**  $i \neq j + 1$   
**and**  $fFx \ (j + 1) \ x$   
**shows**  $\partial \ i \ \alpha \ (\vartheta \ j \ x) = \vartheta \ j \ (\partial \ i \ \alpha \ x)$   
**proof** –  
**have**  $\sigma \ j \ (\partial \ i \ \alpha \ (\vartheta \ j \ x)) = \sigma \ j \ (\partial \ i \ \alpha \ (\partial \ j \ \text{ff} \ (\vartheta \ j \ x)))$   
**using** *assms*( $\mathcal{B}$ ) *inv-sym-type-var* **by** *simp*  
**also have**  $\dots = \partial \ i \ \alpha \ (\sigma \ j \ (\partial \ j \ \alpha \ (\vartheta \ j \ x)))$   
**by** (*metis assms inv-sym-type-var local.fFx-prop sym-face2-var1*)  
**also have**  $\dots = \partial \ i \ \alpha \ (\sigma \ j \ (\vartheta \ j \ x))$   
**using** *assms calculation inv-sym-type-var local.sym-face2* **by** *presburger*



**also have**  $\dots = \partial i \alpha (\partial (j + 1) \alpha x)$   
**by** (*metis* *assms*(3) *local.face-compat-var sym-inv-sym-var1*)  
**finally have**  $\sigma j (\partial i \alpha (\vartheta j x)) = \partial i \alpha (\partial (j + 1) \alpha x)$ .  
**thus** *?thesis*  
**by** (*metis* *assms*(3) *inv-sym-type-var local.fFx-prop local.face-comm-var local.inv-sym-sym*)  
**qed**

**lemma** *sym-braid*:  
**assumes** *fFx i x*  
**and** *fFx (i + 1) x*  
**shows**  $\sigma i (\sigma (i + 1) (\sigma i x)) = \sigma (i + 1) (\sigma i (\sigma (i + 1) x))$   
**using** *assms local.sym-face2 local.sym-fix sym-type-var* **by** *simp*

**lemma** *inv-sym-braid*:  
**assumes** *fFx (i + 1) x*  
**and** *fFx (i + 2) x*  
**shows**  $\vartheta i (\vartheta (i + 1) (\vartheta i x)) = \vartheta (i + 1) (\vartheta i (\vartheta (i + 1) x))$   
**using** *assms inv-sym inv-sym-face2 inv-sym-type-var* **by** *simp*

**lemma** *sym-inv-sym-braid*:  
**assumes** *diffSup i j 2*  
**and** *fFx (j + 1) x*  
**and** *fFx i x*  
**shows**  $\sigma i (\vartheta j x) = \vartheta j (\sigma i x)$   
**by** (*metis* (*no-types, lifting*) *ext assms*(1,2,3) *diff-add-0 diff-add-inverse inv-sym-face2 inv-sym-type-var local.inv-sym-sym local.sym-face2 local.sym-inv-sym not-numeral-le-zero numeral-le-one-iff semiring-norm*(69) *sym-sym-braid-var1*)

**lemma** *sym-func1*:  
**assumes** *fFx i x*  
**and** *fFx i y*  
**and** *DD i x y*  
**shows**  $\sigma i (x \otimes_i y) = \sigma i x \otimes_{(i + 1)} \sigma i y$   
**by** (*metis* *assms icid.ts-compat local.iDst local.icat.sscatml.l0-absorb sym-type-var1*)

**lemma** *sym-func1-var1*:  $\sigma \sigma i (\partial i \alpha x \odot_i \partial i \beta y) = \sigma i (\partial i \alpha x) \odot_{(i + 1)} \sigma i (\partial i \beta y)$   
**by** (*metis* *icid.t-idem image-empty image-insert inv-sym-sym-var1 local.face-compat-var local.icid.stopp.Dst sym-type-var1*)

**lemma** *inv-sym-func2-var1*:  $\vartheta \vartheta i (\partial (i + 1) \alpha x \odot_i \partial (i + 1) \beta y) = \vartheta i (\partial (i + 1) \alpha x) \odot_{(i + 1)} \vartheta i (\partial (i + 1) \beta y)$

**proof**–

**have**  $\sigma \sigma i (\vartheta i (\partial (i + 1) \alpha x) \odot_{(i + 1)} \vartheta i (\partial (i + 1) \beta y)) = \partial (i + 1) \alpha x \odot_i \partial (i + 1) \beta y$   
**by** (*metis* *func2-var2 inv-sym-type-var1 sym-inv-sym-var1*)  
**hence**  $\sigma \sigma i (\partial \partial i \text{ff} (\vartheta i (\partial (i + 1) \alpha x) \odot_{(i + 1)} \vartheta i (\partial (i + 1) \beta y))) = \partial \partial$

$(i + 1) \text{ ff } (\partial (i + 1) \alpha x \odot_i \partial (i + 1) \beta y)$   
**by** (*smt* (*verit*) *diff-add-0* *diff-add-inverse* *inv-sym-type-var1* *local.face-fix-comp-var*  
*local.icid.stopp.ts-compat* *numerals(1)* *zero-neq-numeral*)  
**hence**  $\partial \partial i \text{ ff } (\vartheta i (\partial (i + 1) \alpha x) \odot_{(i + 1)} \vartheta i (\partial (i + 1) \beta y)) = \vartheta \vartheta i (\partial \partial (i + 1) \text{ ff } (\partial (i + 1) \alpha x \odot_i \partial (i + 1) \beta y))$   
**by** (*metis* *inv-sym-sym-lift-var*)  
**thus** *?thesis*  
**by** (*metis* *add-cancel-right-right* *dual-order.eq-iff* *inv-sym-type-var1* *local.face-compat-var*  
*local.face-fix-comp-var* *not-one-le-zero*)  
**qed**

**lemma** *inv-sym-func3-var1*:  $\vartheta \vartheta i ((\partial (i + 1) \alpha x) \odot_{(i + 1)} (\partial (i + 1) \beta y)) = \vartheta i (\partial (i + 1) \alpha x) \odot_i \vartheta i (\partial (i + 1) \beta y)$   
**by** (*smt* (*z3*) *empty-is-image* *image-insert* *inv-sym-type-var1* *local.face-idem1* *local.face-pidem2* *sym-inv-sym-var1*)

**lemma** *inv-sym-func-var1*:

**assumes**  $i \neq j$   
**and**  $j \neq i + 1$   
**shows**  $\vartheta \vartheta i ((\partial (i + 1) \alpha x) \odot_j (\partial (i + 1) \beta y)) = \vartheta i (\partial (i + 1) \alpha x) \odot_j \vartheta i (\partial (i + 1) \beta y)$   
**by** (*smt* (*z3*) *assms(1)* *assms(2)* *inv-sym-sym-lift-var* *inv-sym-type-var1* *local.face-fix-comp-var*  
*local.icid.stopp.ts-compat* *sym-func3-var2* *sym-inv-sym-var1*)

**lemma** *inv-sym-func2*:

**assumes**  $fFx (i + 1) x$   
**and**  $fFx (i + 1) y$   
**and**  $DD i x y$   
**shows**  $\vartheta i (x \otimes_i y) = \vartheta i x \otimes_{(i + 1)} \vartheta i y$

**proof**–

**have**  $\{\vartheta i (x \otimes_i y)\} = \vartheta \vartheta i (x \odot_i y)$   
**using** *assms(3)* *local.icat.pcomp-def-var4* **by** *fastforce*  
**also have**  $\dots = \vartheta i x \odot_{(i + 1)} \vartheta i y$   
**by** (*metis* *assms(1)* *assms(2)* *inv-sym-func2-var1*)  
**also have**  $\dots = \{\vartheta i x \otimes_{(i + 1)} \vartheta i y\}$   
**by** (*metis* *calculation* *insert-not-empty* *local.icat.pcomp-def-var4*)  
**finally show** *?thesis*  
**by** *simp*

**qed**

**lemma** *inv-sym-func3*:

**assumes**  $fFx (i + 1) x$   
**and**  $fFx (i + 1) y$   
**and**  $DD (i + 1) x y$   
**shows**  $\vartheta i (x \otimes_{(i + 1)} y) = \vartheta i x \otimes_i \vartheta i y$   
**by** (*metis* *assms* *icat.st-local* *icid.st-fix* *inv-sym-type-var1* *local.icat.sscatml.l0-absorb*)

**lemma** *inv-sym-func*:

```

assumes  $i \neq j$ 
and  $j \neq i + 1$ 
and  $fFx (i + 1) x$ 
and  $fFx (i + 1) y$ 
and  $DD j x y$ 
shows  $\vartheta i (x \otimes_j y) = \vartheta i x \otimes_j \vartheta i y$ 
proof –
  have  $\{\vartheta i (x \otimes_j y)\} = \vartheta \vartheta i (x \odot_j y)$ 
    using  $assms(5)$  local.icat.pcomp-def-var4 by fastforce
  also have  $\dots = \vartheta i x \odot_j \vartheta i y$ 
    by (metis  $assms(1)$   $assms(2)$   $assms(3)$   $assms(4)$  inv-sym-func-var1)
  also have  $\dots = \{\vartheta i x \otimes_j \vartheta i y\}$ 
    by (metis calculation insert-not-empty local.icat.pcomp-def-var4)
  finally show ?thesis
    by simp
qed

```

The following properties are related to faces and braids.

**lemma** *sym-face3*:

```

assumes  $fFx i x$ 
shows  $\partial (i + 1) \alpha (\sigma i x) = \sigma i (\partial i \alpha x)$ 
by (metis  $assms$  local.fFx-prop sym-type-var1)

```

**lemma** *sym-face3-var1*:  $\partial (i + 1) \alpha (\sigma i (\partial i \beta x)) = \sigma i (\partial i \alpha (\partial i \beta x))$

**proof** –

```

have  $\partial (i + 1) \alpha (\sigma i (\partial i \beta x)) = \partial (i + 1) \alpha (\sigma i (\partial i \alpha (\partial i \beta x)))$ 
  by simp
also have  $\dots = \sigma i (\partial i \alpha (\partial i \beta x))$ 
  using local.sym-type-var1 by fastforce
also have  $\dots = \sigma i (\partial i \beta x)$ 
  by simp
thus ?thesis
  using calculation by simp

```

**qed**

**lemma** *sym-face3-simp* [*simp*]:

```

assumes  $fFx i x$ 
shows  $\partial (i + 1) \alpha (\sigma i x) = \sigma i x$ 
by (metis  $assms$  local.fFx-prop sym-face3)

```

**lemma** *sym-face3-simp-var1* [*simp*]:  $\partial (i + 1) \alpha (\sigma i (\partial i \beta x)) = \sigma i (\partial i \beta x)$

**using** *sym-face3* **by** *simp*

**lemma** *inv-sym-face3*:

```

assumes  $fFx (i + 1) x$ 
shows  $\partial i \alpha (\vartheta i x) = \vartheta i (\partial (i + 1) \alpha x)$ 
by (metis  $assms$  inv-sym-type-var1 local.face-compat-var)

```

**lemma** *inv-sym-face3-var1*:  $\partial i \alpha (\vartheta i (\partial (i + 1) \beta x)) = \vartheta i (\partial (i + 1) \alpha (\partial i \beta x))$

+ 1)  $\beta$  x))  
**by** (metis inv-sym-type-var1 local.face-compat-var)

**lemma** inv-sym-face3-simp:  
**assumes** fFx (i + 1) x  
**shows**  $\partial i \alpha (\vartheta i x) = \vartheta i x$   
**using** assms inv-sym-type-var local.fFx-prop **by** force

**lemma** inv-sym-face3-simp-var1 [simp]:  $\partial i \alpha (\vartheta i (\partial (i + 1) \beta x)) = \vartheta i (\partial (i + 1) \beta x)$   
**using** inv-sym-face3 local.face-compat-var **by** simp

**lemma** inv-sym-face1:  
**assumes** fFx (i + 1) x  
**shows**  $\partial (i + 1) \alpha (\vartheta i x) = \vartheta i (\partial i \alpha x)$   
**by** (metis assms inv-sym-face3-simp inv-sym-sym-var1 local.face-comm-var local.sym-inv-sym sym-face1-var1)

**lemma** inv-sym-face1-var1:  $\partial (i + 1) \alpha (\vartheta i (\partial (i + 1) \beta x)) = \vartheta i (\partial i \alpha (\partial (i + 1) \beta x))$   
**using** inv-sym-face1 local.face-compat-var **by** simp

**lemma** inv-sym-sym-braid:  
**assumes** diffSup i j 2  
**and** fFx j x  
**and** fFx (i + 1) x  
**shows**  $\vartheta i (\sigma j x) = \sigma j (\vartheta i x)$   
**using** assms sym-inv-sym-braid **by** force

**lemma** inv-sym-sym-braid-var1:  $\text{diffSup } i \ j \ 2 \implies \vartheta i (\sigma j (\partial (i + 1) \alpha (\partial j \beta x))) = \sigma j (\vartheta i (\partial (i + 1) \alpha (\partial j \beta x)))$   
**using** local.face-comm-var local.sym-inv-sym-braid **by** force

**lemma** inv-sym-inv-sym-braid:  
**assumes** diffSup i j 2  
**and** fFx (i + 1) x  
**and** fFx (j + 1) x  
**shows**  $\vartheta i (\vartheta j x) = \vartheta j (\vartheta i x)$   
**by** (metis Suc-eq-plus1 add-right-cancel assms inv-sym-face2 inv-sym-face3 inv-sym-sym-braid-var1 local.inv-sym-sym local.sym-inv-sym nat-le-linear not-less-eq-eq)

**lemma** inv-sym-inv-sym-braid-var1:  $\text{diffSup } i \ j \ 2 \implies \vartheta i (\vartheta j (\partial (i + 1) \alpha (\partial (j + 1) \beta x))) = \vartheta j (\vartheta i (\partial (i + 1) \alpha (\partial (j + 1) \beta x)))$   
**using** inv-sym-inv-sym-braid local.face-comm-var **by** force

The following properties are related to symcomp and inv-symcomp.

**lemma** symcomp-type-var:  
**assumes** fFx i x  
**shows** fFx (i + j) ( $\Sigma i \ j \ x$ ) **using** ⟨fFx i x⟩

**apply** (*induct j*)  
**using** *sym-face3* **by** *simp-all*

**lemma** *symcomp-type: image* ( $\Sigma i j$ ) (*face-fix i*)  $\subseteq$  *face-fix (i + j)*  
**using** *symcomp-type-var* **by** *force*

**lemma** *symcomp-type-var1* [*simp*]:  $\partial (i + j) \alpha (\Sigma i j (\partial i \beta x)) = \Sigma i j (\partial i \beta x)$   
**by** (*metis local.face-compat-var symcomp-type-var*)

**lemma** *inv-symcomp-type-var*:  
**assumes** *fFx (i + j) x*  
**shows** *fFx i* ( $\Theta i j x$ ) **using**  $\langle fFx (i + j) x \rangle$   
**by** (*induct j arbitrary: x, simp-all add: inv-sym-type-var*)

**lemma** *inv-symcomp-type: image* ( $\Theta i j$ ) (*face-fix (i + j)*)  $\subseteq$  *face-fix i*  
**using** *inv-symcomp-type-var* **by** *force*

**lemma** *inv-symcomp-type-var1* [*simp*]:  $\partial i \alpha (\Theta i j (\partial (i + j) \beta x)) = \Theta i j (\partial (i + j) \beta x)$   
**by** (*meson inv-symcomp-type-var local.fFx-prop local.face-compat-var*)

**lemma** *symcomp-inv-symcomp*:  
**assumes** *fFx (i + j) x*  
**shows**  $\Sigma i j (\Theta i j x) = x$  **using**  $\langle fFx (i + j) x \rangle$   
**by** (*induct j arbitrary: i x, simp-all add: inv-sym-type-var local.sym-inv-sym*)

**lemma** *inv-symcomp-symcomp*:  
**assumes** *fFx i x*  
**shows**  $\Theta i j (\Sigma i j x) = x$  **using**  $\langle fFx i x \rangle$   
**by** (*induct j arbitrary: i x, simp-all add: local.inv-sym-sym symcomp-type-var*)

**lemma** *symcomp-adj*:  
**assumes** *fFx i x*  
**and** *fFx (i + j) y*  
**shows**  $(\Sigma i j x = y) = (x = \Theta i j y)$   
**using** *assms inv-symcomp-symcomp symcomp-inv-symcomp* **by** *force*

**lemma** *decomp-symcomp1*:  
**assumes**  $k \leq j$   
**and** *fFx i x*  
**shows**  $\Sigma i j x = \Sigma (i + k) (j - k) (\Sigma i k x)$  **using**  $\langle k \leq j \rangle$   
**apply** (*induct j*)  
**using** *Suc-diff-le le-Suc-eq* **by** *force+*

**lemma** *decomp-symcomp2*:  
**assumes**  $1 \leq k$   
**and**  $k \leq j$   
**and** *fFx i x*  
**shows**  $\Sigma i j x = \Sigma (i + k) (j - k) (\sigma (i + k - 1) (\Sigma i (k - 1) x))$

by (metis Nat.add-diff-assoc add-diff-cancel-left' assms decomp-symcomp1 local.symcomp.simps(2) plus-1-eq-Suc)

**lemma** *decomp-symcomp3*:

assumes  $i \leq l$   
and  $l + 1 \leq i + j$   
and  $fFx\ i\ x$   
shows  $\Sigma\ i\ j\ x = \Sigma\ (l + 1)\ (i + j - l - 1)\ (\sigma\ l\ (\Sigma\ i\ (l - i)\ x))$   
by (smt (verit, del-insts) add commute add-le-cancel-left assms decomp-symcomp2 diff-add-inverse2 diff-diff-left le-add1 le-add-diff-inverse)

**lemma** *symcomp-face2*:

assumes  $l < i \vee i + j < l$   
and  $fFx\ i\ x$   
shows  $\partial\ l\ \alpha\ (\Sigma\ i\ j\ x) = \Sigma\ i\ j\ (\partial\ l\ \alpha\ x)$  using  $\langle l < i \vee i + j < l \rangle$   
**proof** (induct j)

case 0  
show ?case  
by simp  
**next**  
case (Suc j)  
have  $\partial\ l\ \alpha\ (\Sigma\ i\ (Suc\ j)\ x) = \partial\ l\ \alpha\ (\sigma\ (i + j)\ (\Sigma\ i\ j\ x))$   
by simp  
also have  $\dots = \sigma\ (i + j)\ (\partial\ l\ \alpha\ (\Sigma\ i\ j\ x))$   
using Suc.prem1 add commute assms(2) local.sym-face2 symcomp-type-var by auto  
also have  $\dots = \sigma\ (i + j)\ (\Sigma\ i\ j\ (\partial\ l\ \alpha\ x))$   
using Suc.hyps Suc.prem1 by fastforce  
also have  $\dots = (\Sigma\ i\ (Suc\ j)\ (\partial\ l\ \alpha\ x))$   
by simp  
**finally show** ?case.  
**qed**

**lemma** *symcomp-face3*:  $fFx\ i\ x \implies \partial\ (i + j)\ \alpha\ (\Sigma\ i\ j\ x) = \Sigma\ i\ j\ (\partial\ i\ \alpha\ x)$   
by (metis local.face-compat-var symcomp-type-var1)

**lemma** *symcomp-face1*:

assumes  $i \leq l$   
and  $l < i + j$   
and  $fFx\ i\ x$   
shows  $\partial\ l\ \alpha\ (\Sigma\ i\ j\ x) = \Sigma\ i\ j\ (\partial\ (l + 1)\ \alpha\ x)$   
**proof** –  
have  $\partial\ l\ \alpha\ (\Sigma\ i\ j\ x) = \partial\ l\ \alpha\ (\Sigma\ (l + 1)\ (i + j - l - 1)\ (\sigma\ l\ (\Sigma\ i\ (l - i)\ x)))$   
using Suc-eq-plus1 Suc-leI assms(1) assms(2) assms(3) decomp-symcomp3 by presburger  
also have  $\dots = \Sigma\ (l + 1)\ (i + j - l - 1)\ (\partial\ l\ \alpha\ (\sigma\ l\ (\Sigma\ i\ (l - i)\ x)))$   
by (metis assms(1) assms(3) less-add-one ordered-cancel-comm-monoid-diff-class.add-diff-inverse sym-type-var symcomp-face2 symcomp-face3)  
also have  $\dots = \Sigma\ (l + 1)\ (i + j - l - 1)\ (\sigma\ l\ (\partial\ (l + 1)\ \alpha\ (\Sigma\ i\ (l - i)\ x)))$

**by** (*metis* *assms*(1) *assms*(3) *local.sym-face1* *ordered-cancel-comm-monoid-diff-class.add-diff-inverse* *symcomp-face3*)  
**also have**  $\dots = \Sigma (l + 1) (i + j - l - 1) (\sigma l (\Sigma i (l - i) (\partial (l + 1) \alpha x)))$   
**by** (*simp* *add: assms*(1) *assms*(3) *symcomp-face2*)  
**also have**  $\dots = \Sigma i j (\partial (l + 1) \alpha x)$   
**by** (*metis* *Suc-eq-plus1* *Suc-leI* *assms*(1) *assms*(2) *assms*(3) *decomp-symcomp3* *local.fFx-prop* *local.face-comm-var*)  
**finally show** *?thesis*.  
**qed**

**lemma** *inv-symcomp-face2*:  
**assumes**  $l < i \vee i + j < l$   
**and** *fFx* (i + j) x  
**shows**  $\partial l \alpha (\Theta i j x) = \Theta i j (\partial l \alpha x)$  **using**  $\langle l < i \vee i + j < l \rangle \langle fFx (i + j) x \rangle$   
**proof** (*induct* j *arbitrary*: x)  
**case** 0  
**show** *?case*  
**using** *local.inv-sym-face2* **by** *force*  
**next**  
**case** (*Suc* j)  
**have**  $\partial l \alpha (\Theta i (Suc j) x) = \Theta i j (\partial l \alpha (\vartheta (i + j) x))$   
**using** *Suc.hyps* *Suc.prem*(1) *Suc.prem*(2) *inv-sym-type-var* **by** *force*  
**also have**  $\dots = \Theta i j (\vartheta (i + j) (\partial l \alpha x))$   
**using** *Suc.prem* *inv-sym-face2* **by** *force*  
**also have**  $\dots = (\Theta i (Suc j) (\partial l \alpha x))$   
**by** *simp*  
**finally show** *?case*.  
**qed**

**lemma** *inv-symcomp-face3*: *fFx* (i + j) x  $\implies \partial i \alpha (\Theta i j x) = \Theta i j (\partial (i + j) \alpha x)$   
**by** (*metis* *inv-symcomp-type-var1* *local.face-compat-var*)

**lemma** *inv-symcomp-face1*:  
**assumes**  $i < l$   
**and**  $l \leq i + j$   
**and** *fFx* (i + j) x  
**shows**  $\partial l \alpha (\Theta i j x) = \Theta i j (\partial (l - 1) \alpha x)$   
**proof** –  
**have**  $(\partial (l - 1) \alpha (\Sigma i j (\Theta i j x))) = \partial (l - 1) \alpha x$   
**using** *assms*(3) *symcomp-inv-symcomp* **by** *force*  
**hence**  $(\Sigma i j (\partial l \alpha (\Theta i j x))) = \partial (l - 1) \alpha x$   
**using** *assms* *inv-symcomp-type-var* *symcomp-face1* **by** *auto*  
**thus** *?thesis*  
**by** (*metis* *assms*(1) *assms*(3) *inv-symcomp-symcomp* *inv-symcomp-type-var* *local.face-comm-var* *nat-neq-iff*)  
**qed**

**lemma** *symcomp-comp1*:  
**assumes**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ i\ x\ y$   
**shows**  $\Sigma\ i\ j\ (x \otimes_i y) = \Sigma\ i\ j\ x \otimes_{(i+j)} \Sigma\ i\ j\ y$   
**by** (*induct j, simp, metis assms local.face-compat-var local.iDst local.icat.sscatml.r0-absorb symcomp-type-var1*)

**lemma** *symcomp-comp2*:  
**assumes**  $k < i$   
**and**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ k\ x\ y$   
**shows**  $\Sigma\ i\ j\ (x \otimes_k y) = \Sigma\ i\ j\ x \otimes_k \Sigma\ i\ j\ y$   
**proof** (*induct j*)  
**case** 0  
**show** ?case  
**by** *simp*  
**next**  
**case** (*Suc j*)  
**have**  $\Sigma\ i\ (Suc\ j)\ (x \otimes_k y) = \sigma\ (i + j)\ ((\Sigma\ i\ j\ x) \otimes_k (\Sigma\ i\ j\ y))$   
**by** (*simp add: Suc*)  
**also have**  $\dots = \sigma\ (i + j)\ (\Sigma\ i\ j\ x) \otimes_k \sigma\ (i + j)\ (\Sigma\ i\ j\ y)$   
**apply** (*rule sym-func3*)  
**using** *assms(1) assms(2) assms(3) symcomp-type-var* **apply** *presburger+*  
**using** *assms local.iDst local.locality symcomp-face2* **by** *presburger*  
**finally show** ?case  
**by** *simp*  
**qed**

**lemma** *symcomp-comp3*:  
**assumes**  $i + j < k$   
**and**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ k\ x\ y$   
**shows**  $\Sigma\ i\ j\ (x \otimes_k y) = \Sigma\ i\ j\ x \otimes_k \Sigma\ i\ j\ y$  **using**  $\langle k > i + j \rangle$   
**proof** (*induct j*)  
**case** 0  
**show** ?case  
**by** *simp*  
**next**  
**case** (*Suc j*)  
**have**  $\Sigma\ i\ (Suc\ j)\ (x \otimes_k y) = \sigma\ (i + j)\ ((\Sigma\ i\ j\ x) \otimes_k (\Sigma\ i\ j\ y))$   
**using** *Suc.hyps Suc.prem1 by force*  
**also have**  $\dots = \sigma\ (i + j)\ (\Sigma\ i\ j\ x) \otimes_k \sigma\ (i + j)\ (\Sigma\ i\ j\ y)$   
**apply** (*rule sym-func3*)  
**using** *Suc.prem1* **apply** *linarith+*  
**using** *assms(2) assms(3) symcomp-type-var* **apply** *presburger+*  
**using** *Suc.prem1 assms(2) assms(3) assms(4) local.icid.ts-msg.st-locality-locality*



*symcomp-face2* **by** *simp*  
**finally show** *?case*  
**by** *simp*  
**qed**

**lemma** *fix-comp*:  
**assumes**  $i \neq j$   
**and**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ j\ x\ y$   
**shows**  $fFx\ i\ (x \otimes_j y)$   
**using** *face-func assms* **by** *simp*

**lemma** *symcomp-comp4*:  
**assumes**  $i < k$   
**and**  $k \leq i + j$   
**and**  $fFx\ i\ x$   
**and**  $fFx\ i\ y$   
**and**  $DD\ k\ x\ y$   
**shows**  $\Sigma\ i\ j\ (x \otimes_k y) = \Sigma\ i\ j\ x \otimes_{(k-1)} \Sigma\ i\ j\ y$   
**using**  $\langle k \leq i + j \rangle \langle fFx\ i\ x \rangle \langle fFx\ i\ y \rangle \langle DD\ k\ x\ y \rangle$   
**proof** (*induct j arbitrary: x y*)  
**case** 0  
**thus** *?case*  
**using** *assms(1)* **by** *linarith*  
**next**  
**case** (*Suc j*)  
**have**  $a: fFx\ i\ (x \otimes_k y)$   
**using** *Suc.prem(2) Suc.prem(3) Suc.prem(4) assms(1) fix-comp* **by** *force*  
**have**  $b: fFx\ (k-1)\ (\Sigma\ i\ (k-1-i)\ x)$   
**using** *Suc.prem(2) assms(1) less-imp-Suc-add symcomp-type-var* **by** *fastforce*  
**have**  $c: fFx\ (k-1)\ (\Sigma\ i\ (k-1-i)\ y)$   
**using** *Suc.prem(3) assms(1) less-imp-Suc-add symcomp-type-var* **by** *fastforce*  
**have**  $d: DD\ k\ (\Sigma\ i\ (k-1-i)\ x)\ (\Sigma\ i\ (k-1-i)\ y)$   
**by** (*metis Suc.prem(2) Suc.prem(3) Suc.prem(4) add-diff-cancel-left' assms(1) lessI less-imp-Suc-add local.iDst local.locality plus-1-eq-Suc symcomp-face2*)  
**have**  $\Sigma\ i\ (Suc\ j)\ (x \otimes_k y) = \Sigma\ k\ (i + j + 1 - k)\ (\sigma\ (k-1)\ (\Sigma\ i\ (k-1-i)\ (x \otimes_k y)))$   
**by** (*smt (verit) Suc.prem(1) Suc-eq-plus1 a add-Suc-right add-le-imp-le-diff assms(1) decomp-symcomp3 diff-diff-left le-add-diff-inverse2 less-eq-Suc-le plus-1-eq-Suc*)  
**also have**  $\dots = \Sigma\ k\ (i + j + 1 - k)\ (\sigma\ (k-1)\ (\Sigma\ i\ (k-1-i)\ x \otimes_k \Sigma\ i\ (k-1-i)\ y))$   
**using** *Suc.prem(2) Suc.prem(3) Suc.prem(4) assms(1) symcomp-comp3* **by** *force*  
**also have**  $\dots = \Sigma\ k\ (i + j + 1 - k)\ (\sigma\ (k-1)\ (\Sigma\ i\ (k-1-i)\ x \otimes_{((k-1)+1)} \Sigma\ i\ (k-1-i)\ y))$   
**using** *assms(1)* **by** *auto*  
**also have**  $\dots = \Sigma\ k\ (i + j + 1 - k)\ (\sigma\ (k-1)\ (\Sigma\ i\ (k-1-i)\ x) \otimes_{(k-1)} \sigma\ (k-1)\ (\Sigma\ i\ (k-1-i)\ y))$

**using** *assms(1) b c d less-iff-Suc-add sym-func2* **by** *fastforce*  
**also have**  $\dots = \Sigma k (i + j + 1 - k) (\sigma (k - 1) (\Sigma i (k - 1 - i) x)) \otimes_{(k - 1)}$   
 $\Sigma k (i + j + 1 - k) (\sigma (k - 1) (\Sigma i (k - 1 - i) y))$   
**apply** (*rule symcomp-comp2*)  
**using** *assms(1) b sym-face3* **apply** *fastforce+*  
**apply** (*metis assms(1) c le-add1 le-add-diff-inverse2 less-imp-Suc-add plus-1-eq-Suc*  
*sym-face3*)  
**by** (*metis assms(1) b c d le-add1 le-add-diff-inverse2 less-imp-Suc-add plus-1-eq-Suc*  
*sym-func2-DD*)  
**also have**  $\dots = \Sigma k (i + j + 1 - k) (\Sigma i (k - i) x) \otimes_{(k - 1)} \Sigma k (i + j + 1$   
 $- k) (\Sigma i (k - i) y)$   
**using** *assms(1) less-imp-Suc-add* **by** *fastforce*  
**also have**  $\dots = (\Sigma i (j + 1) x) \otimes_{(k - 1)} \Sigma k (i + j + 1 - k) (\Sigma i (k - i) y)$   
**by** (*smt (verit, ccfv-SIG) Nat.diff-diff-eq Suc.prem(1) Suc.prem(2) add.comm-neutral*  
*add-left-mono assms(1) decomp-symcomp1 diff-add-inverse diff-le-mono group-cancel.add2*  
*linordered-semidom-class.add-diff-inverse order-less-imp-le order-less-imp-not-less*  
*plus-1-eq-Suc zero-less-Suc*)  
**also have**  $\dots = (\Sigma i (j + 1) x) \otimes_{(k - 1)} (\Sigma i (j + 1) y)$   
**by** (*smt (verit, ccfv-SIG) Nat.add-0-right Nat.diff-diff-eq Suc.prem(1) Suc.prem(3)*  
*add-Suc add-Suc-shift add-diff-inverse-nat add-mono-thms-linordered-semiring(2)*  
*assms(1) decomp-symcomp1 diff-add-inverse diff-le-mono nless-le order.asym plus-1-eq-Suc*  
*trans-less-add2 zero-less-one*)  
**finally show** *?case*  
**by** *simp*  
**qed**

**lemma** *symcomp-comp:*

**assumes** *fFx i x*  
**and** *fFx i y*  
**and** *DD k x y*  
**shows**  $\Sigma i j (x \otimes_k y) = (\text{if } k = i \text{ then } \Sigma i j x \otimes_{(i + j)} \Sigma i j y$   
 $\text{else } (\text{if } (i < k \wedge k \leq i + j) \text{ then } \Sigma i j x \otimes_{(k - 1)} \Sigma i j y$   
 $\text{else } \Sigma i j x \otimes_k \Sigma i j y))$   
**by** (*metis assms linorder-not-le not-less-iff-gr-or-eq symcomp-comp1 symcomp-comp2*  
*symcomp-comp3 symcomp-comp4*)

**lemma** *inv-symcomp-comp1:*

**assumes** *fFx (i + j) x*  
**and** *fFx (i + j) y*  
**and** *DD (i + j) x y*  
**shows**  $\Theta i j (x \otimes_{(i + j)} y) = \Theta i j x \otimes_i \Theta i j y$   
**by** (*metis assms inv-symcomp-type-var local.fFx-prop local.iDst local.icat.sscatml.l0-absorb*)

**lemma** *inv-symcomp-comp2:*

**assumes**  $k < i$   
**and** *fFx (i + j) x*  
**and** *fFx (i + j) y*  
**and** *DD k x y*

shows  $\Theta i j (x \otimes_k y) = \Theta i j x \otimes_k \Theta i j y$   
**proof** –  
 have  $a: DD k (\Theta i j x) (\Theta i j y)$   
 using *assms inv-symcomp-face2 local.iDst local.locality* **by** *presburger*  
 have  $x \otimes_k y = \Sigma i j (\Theta i j x) \otimes_k \Sigma i j (\Theta i j y)$   
 by (*simp add: assms(2) assms(3) symcomp-inv-symcomp*)  
 hence  $x \otimes_k y = \Sigma i j ((\Theta i j x) \otimes_k (\Theta i j y))$   
 using  $a$  *assms(1) assms(2) assms(3) inv-symcomp-type-var symcomp-comp2*  
**by** *presburger*  
 thus *?thesis*  
 using  $a$  *assms(1) assms(2) assms(3) fix-comp inv-symcomp-face3 inv-symcomp-symcomp*  
**by** *simp*  
**qed**

**lemma** *inv-symcomp-comp3*:

assumes  $i + j < k$   
 and *fFx (i + j) x*  
 and *fFx (i + j) y*  
 and *DD k x y*  
 shows  $\Theta i j (x \otimes_k y) = \Theta i j x \otimes_k \Theta i j y$   
**proof** –  
 have  $h: DD k (\Theta i j x) (\Theta i j y)$   
 using *assms inv-symcomp-face2 local.iDst local.locality* **by** *presburger*  
 have  $x \otimes_k y = \Sigma i j (\Theta i j x) \otimes_k \Sigma i j (\Theta i j y)$   
 by (*simp add: assms(2) assms(3) symcomp-inv-symcomp*)  
 hence  $x \otimes_k y = \Sigma i j ((\Theta i j x) \otimes_k (\Theta i j y))$   
 using *assms(1) assms(2) assms(3) h inv-symcomp-face3 symcomp-comp3* **by**  
*simp*  
 thus *?thesis*  
 using *assms(1) assms(2) assms(3) fix-comp h inv-symcomp-face3 inv-symcomp-symcomp*  
**by** *simp*  
**qed**

**lemma** *inv-symcomp-comp4*:

assumes  $i \leq k$   
 and  $k < i + j$   
 and *fFx (i + j) x*  
 and *fFx (i + j) y*  
 and *DD k x y*  
 shows  $\Theta i j (x \otimes_k y) = \Theta i j x \otimes_{(k+1)} \Theta i j y$   
**proof** –  
 have  $h: DD (k + 1) (\Theta i j x) (\Theta i j y)$   
 using *assms(1) assms(2) assms(3) assms(4) assms(5) inv-symcomp-face1*  
*local.icat.sts-msg.st-local* **by** *auto*  
 have  $x \otimes_k y = \Sigma i j (\Theta i j x) \otimes_k \Sigma i j (\Theta i j y)$   
 by (*simp add: assms(3) assms(4) symcomp-inv-symcomp*)  
 hence  $x \otimes_k y = \Sigma i j ((\Theta i j x) \otimes_{(k+1)} (\Theta i j y))$   
 apply (*subst symcomp-comp4*)  
 using *assms h inv-symcomp-type-var* **by** *auto*

```

thus ?thesis
  by (metis Suc-eq-plus1 Suc-n-not-le-n assms(1) assms(3) assms(4) fix-comp h
    inv-symcomp-face3 inv-symcomp-symcomp)
qed

end

end

```

## 4 Cubical Categories with Connections

```

theory CubicalCategoriesConnections
  imports CubicalCategories

```

```

begin

```

All categories considered in this component are single-set categories.

```

class connection-ops =
  fixes connection :: nat  $\Rightarrow$  bool  $\Rightarrow$  'a  $\Rightarrow$  'a ( $\langle \Gamma \rangle$ )

```

```

abbreviation (in connection-ops)  $\Gamma \Gamma \ i \ \alpha \equiv$  image ( $\Gamma \ i \ \alpha$ )

```

We define a class for cubical  $\omega$ -categories with connections.

```

class cubical-omega-category-connections = cubical-omega-category + connection-ops
+
  assumes conn-face1:  $fFx \ j \ x \Longrightarrow \partial \ j \ \alpha \ (\Gamma \ j \ \alpha \ x) = x$ 
  and conn-face2:  $fFx \ j \ x \Longrightarrow \partial \ (j + 1) \ \alpha \ (\Gamma \ j \ \alpha \ x) = \sigma \ j \ x$ 
  and conn-face3:  $i \neq j \Longrightarrow i \neq j + 1 \Longrightarrow fFx \ j \ x \Longrightarrow \partial \ i \ \alpha \ (\Gamma \ j \ \beta \ x) = \Gamma \ j \ \beta$ 
  ( $\partial \ i \ \alpha \ x$ )
  and conn-corner1:  $fFx \ i \ x \Longrightarrow fFx \ i \ y \Longrightarrow DD \ (i + 1) \ x \ y \Longrightarrow \Gamma \ i \ tt \ (x \otimes_{(i + 1)}$ 
   $y) = (\Gamma \ i \ tt \ x \otimes_{(i + 1)} \sigma \ i \ x) \otimes_i (x \otimes_{(i + 1)} \Gamma \ i \ tt \ y)$ 
  and conn-corner2:  $fFx \ i \ x \Longrightarrow fFx \ i \ y \Longrightarrow DD \ (i + 1) \ x \ y \Longrightarrow \Gamma \ i \ ff \ (x \otimes_{(i + 1)}$ 
   $y) = (\Gamma \ i \ ff \ x \otimes_{(i + 1)} y) \otimes_i (\sigma \ i \ y \otimes_{(i + 1)} \Gamma \ i \ ff \ y)$ 
  and conn-corner3:  $j \neq i \wedge j \neq i + 1 \Longrightarrow fFx \ i \ x \Longrightarrow fFx \ i \ y \Longrightarrow DD \ j \ x \ y \Longrightarrow$ 
   $\Gamma \ i \ \alpha \ (x \otimes_j y) = \Gamma \ i \ \alpha \ x \otimes_j \Gamma \ i \ \alpha \ y$ 
  and conn-fix:  $fFx \ i \ x \Longrightarrow fFx \ (i + 1) \ x \Longrightarrow \Gamma \ i \ \alpha \ x = x$ 
  and conn-zigzag1:  $fFx \ i \ x \Longrightarrow \Gamma \ i \ tt \ x \otimes_{(i + 1)} \Gamma \ i \ ff \ x = x$ 
  and conn-zigzag2:  $fFx \ i \ x \Longrightarrow \Gamma \ i \ tt \ x \otimes_i \Gamma \ i \ ff \ x = \sigma \ i \ x$ 
  and conn-conn-braid:  $diffSup \ i \ j \ 2 \Longrightarrow fFx \ j \ x \Longrightarrow fFx \ i \ x \Longrightarrow \Gamma \ i \ \alpha \ (\Gamma \ j \ \beta \ x)$ 
   $= \Gamma \ j \ \beta \ (\Gamma \ i \ \alpha \ x)$ 
  and conn-shift:  $fFx \ i \ x \Longrightarrow fFx \ (i + 1) \ x \Longrightarrow \sigma \ (i + 1) \ (\sigma \ i \ (\Gamma \ (i + 1) \ \alpha \ x))$ 
   $= \Gamma \ i \ \alpha \ (\sigma \ (i + 1) \ x)$ 

```

```

begin

```

```

lemma conn-face4:  $fFx \ j \ x \Longrightarrow \partial \ j \ \alpha \ (\Gamma \ j \ (\neg \alpha) \ x) = \partial \ (j + 1) \ \alpha \ x$ 
  by (smt (z3) local.conn-face1 local.conn-zigzag2 local.face-comm-var local.locality
    local.pcomp-lface local.pcomp-uface local.sym-face1 local.sym-fix-var1)

```

**lemma** *conn-face1-lift*:  $FFx\ j\ X \implies \partial\partial\ j\ \alpha\ (\Gamma\Gamma\ j\ \alpha\ X) = X$   
**by** (*auto simp add: image-iff local.conn-face1*)

**lemma** *conn-face4-lift*:  $FFx\ j\ X \implies \partial\partial\ j\ \alpha\ (\Gamma\Gamma\ j\ (\neg\alpha)\ X) = \partial\partial\ (j + 1)\ \alpha\ X$   
**apply** *safe*  
**apply** (*simp add: local.conn-face4*)  
**by** (*metis image-eqI local.conn-face4*)

**lemma** *conn-face2-lift*:  $FFx\ j\ X \implies \partial\partial\ (j + 1)\ \alpha\ (\Gamma\Gamma\ j\ \alpha\ X) = \sigma\sigma\ j\ X$   
**by** (*smt (z3) comp-apply image-comp image-cong local.conn-face2*)

**lemma** *conn-face3-lift*:  $i \neq j \implies i \neq j + 1 \implies FFx\ j\ X \implies \partial\partial\ i\ \alpha\ (\Gamma\Gamma\ j\ \beta\ X) = \Gamma\Gamma\ j\ \beta\ (\partial\partial\ i\ \alpha\ X)$   
**by** (*smt (z3) image-cong image-image local.conn-face3*)

**lemma** *conn-fix-lift*:  $FFx\ i\ X \implies FFx\ (i + 1)\ X \implies \Gamma\Gamma\ i\ \alpha\ X = X$   
**by** (*simp add: local.conn-fix*)

**lemma** *conn-conn-braid-lift*:  
**assumes** *diffSup i j 2*  
**and** *FFx j X*  
**and** *FFx i X*  
**shows**  $\Gamma\Gamma\ i\ \alpha\ (\Gamma\Gamma\ j\ \beta\ X) = \Gamma\Gamma\ j\ \beta\ (\Gamma\Gamma\ i\ \alpha\ X)$   
**by** (*smt (z3) assms image-cong image-image local.conn-conn-braid*)

**lemma** *conn-sym-braid*:  
**assumes** *diffSup i j 2*  
**and** *fFx i x*  
**and** *fFx j x*  
**shows**  $\Gamma\ i\ \alpha\ (\sigma\ j\ x) = \sigma\ j\ (\Gamma\ i\ \alpha\ x)$   
**by** (*smt (z3) assms add-diff-cancel-left' cancel-comm-monoid-add-class.diff-cancel diff-is-0-eq' icat.st-local le-add1 local.conn-conn-braid local.conn-corner3 local.conn-face1 local.conn-face3 local.conn-zigzag2 numeral-le-one-iff rel-simps(28) semiring-norm(69)*)

**lemma** *conn-zigzag1-var* [*simp*]:  $\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x) \odot_{(i+1)} \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x) = \{\partial\ i\ \alpha\ x\}$   
**proof** (*cases DD (i + 1) (\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x)) (\Gamma\ i\ ff\ (\partial\ i\ \alpha\ x))*)  
**case** *True*  
**hence**  $\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x) \odot_{(i+1)} \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x) = \{\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x) \otimes_{(i+1)} \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x)\}$   
**by** (*metis True local.icat.pcomp-def-var4*)  
**also have**  $\dots = \{\partial\ i\ \alpha\ x\}$   
**using** *local.conn-zigzag1* **by** *simp*  
**finally show** *?thesis*.

**next**  
**case** *False*  
**thus** *?thesis*  
**using** *local.conn-face2 local.locality* **by** *simp*

qed

**lemma** *conn-zigzag1-lift*:

assumes  $FFx\ i\ X$

shows  $\Gamma\Gamma\ i\ tt\ X\ \star_{(i+1)}\ \Gamma\Gamma\ i\ ff\ X = X$

**proof**–

have  $\Gamma\Gamma\ i\ tt\ X\ \star_{(i+1)}\ \Gamma\Gamma\ i\ ff\ X = \{\Gamma\ i\ tt\ x\ \otimes_{(i+1)}\ \Gamma\ i\ ff\ y\ |\ x\ y.\ x \in X \wedge y \in X \wedge DD\ (i+1)\ (\Gamma\ i\ tt\ x)\ (\Gamma\ i\ ff\ y)\}$

unfolding *local.iconv-prop* by *force*

also have  $\dots = \{\Gamma\ i\ tt\ x\ \otimes_{(i+1)}\ \Gamma\ i\ ff\ y\ |\ x\ y.\ x \in X \wedge y \in X \wedge \partial\ (i+1)\ tt\ (\Gamma\ i\ tt\ x) = \partial\ (i+1)\ ff\ (\Gamma\ i\ ff\ y)\}$

using *icat.st-local* by *presburger*

also have  $\dots = \{\Gamma\ i\ tt\ x\ \otimes_{(i+1)}\ \Gamma\ i\ ff\ y\ |\ x\ y.\ x \in X \wedge y \in X \wedge \sigma\ i\ x = \sigma\ i\ y\}$

by (*metis* (*no-types*, *lifting*) *assms local.conn-face2*)

also have  $\dots = \{\Gamma\ i\ tt\ x\ \otimes_{(i+1)}\ \Gamma\ i\ ff\ x\ |\ x.\ x \in X\}$

using *assms local.sym-inj-var* by *blast*

also have  $\dots = \{\Gamma\ i\ tt\ (\partial\ i\ tt\ x)\ \otimes_{(i+1)}\ \Gamma\ i\ ff\ (\partial\ i\ tt\ x)\ |\ x.\ x \in X\}$

by (*metis* (*full-types*) *assms icid.ts-compat*)

also have  $\dots = \{\partial\ i\ tt\ x\ |.\ x \in X\}$

using *local.conn-zigzag1 local.face-compat-var* by *presburger*

also have  $\dots = X$

by (*smt* (*verit*, *del-insts*) *Collect-cong Collect-mem-eq assms local.icid.stopp.st-fix*)

finally show *?thesis*.

qed

**lemma** *conn-zigzag2-var*:  $\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x)\ \odot_i\ \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x) = \{\sigma\ i\ (\partial\ i\ \alpha\ x)\}$

**proof** (*cases DD\ i\ (\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x))\ (\Gamma\ i\ ff\ (\partial\ i\ \alpha\ x))*)

case *True*

hence  $\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x)\ \odot_i\ \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x) = \{\Gamma\ i\ tt\ (\partial\ i\ \alpha\ x)\ \otimes_i\ \Gamma\ i\ ff\ (\partial\ i\ \alpha\ x)\}$

by (*metis True local.icat.pcomp-def-var4*)

also have  $\dots = \{\sigma\ i\ (\partial\ i\ \alpha\ x)\}$

using *local.conn-zigzag2* by *simp*

finally show *?thesis*.

**next**

case *False*

thus *?thesis*

by (*simp add: local.conn-face1 local.locality*)

qed

**lemma** *conn-zigzag2-lift*:

assumes  $FFx\ i\ X$

shows  $\Gamma\Gamma\ i\ tt\ X\ \star_i\ \Gamma\Gamma\ i\ ff\ X = \sigma\sigma\ i\ X$

**proof**–

have  $\Gamma\Gamma\ i\ tt\ X\ \star_i\ \Gamma\Gamma\ i\ ff\ X = \{\Gamma\ i\ tt\ x\ \otimes_i\ \Gamma\ i\ ff\ y\ |.\ x\ y.\ x \in X \wedge y \in X \wedge DD\ i\ (\Gamma\ i\ tt\ x)\ (\Gamma\ i\ ff\ y)\}$

unfolding *local.iconv-prop* by *force*

also have  $\dots = \{\Gamma\ i\ tt\ x\ \otimes_i\ \Gamma\ i\ ff\ y\ |.\ x\ y.\ x \in X \wedge y \in X \wedge \partial\ i\ tt\ (\Gamma\ i\ tt\ x) =$

$\partial i \text{ ff } (\Gamma i \text{ ff } y)$   
**using** *icat.st-local* **by** *presburger*  
**also have**  $\dots = \{\Gamma i \text{ tt } x \otimes_i \Gamma i \text{ ff } x \mid x. x \in X\}$   
**by** (*metis (full-types) assms local.conn-face1*)  
**also have**  $\dots = \{\Gamma i \text{ tt } (\partial i \text{ tt } x) \otimes_i \Gamma i \text{ ff } (\partial i \text{ tt } x) \mid x. x \in X\}$   
**by** (*metis (full-types) assms icid.ts-compat*)  
**also have**  $\dots = \{\sigma i x \mid x. x \in X\}$   
**by** (*metis assms icid.ts-compat local.conn-zigzag2*)  
**also have**  $\dots = \sigma \sigma i X$   
**by** *force*  
**finally show** *?thesis*.  
**qed**

**lemma** *conn-sym-braid-lift*:  $\text{diffSup } i \ j \ 2 \implies \text{FFx } i \ X \implies \text{FFx } j \ X \implies \Gamma \Gamma \ i \ \alpha$   
 $(\sigma \sigma \ j \ X) = \sigma \sigma \ j \ (\Gamma \Gamma \ i \ \alpha \ X)$   
**by** (*smt (z3) image-cong image-image local.conn-sym-braid*)

**lemma** *conn-corner1-DD*:

**assumes** *fFx i x*  
**and** *fFx i y*  
**and** *DD (i + 1) x y*  
**shows** *DD i*  $(\Gamma i \text{ tt } x \otimes_{(i+1)} \sigma i x) (x \otimes_{(i+1)} \Gamma i \text{ tt } y)$

**proof** –

**have** *h1*: *DD (i + 1) (Γ i tt x) (σ i x)*  
**using** *assms(1) local.conn-face2 local.locality local.sym-type-var* **by** *simp*  
**have** *h2*: *DD (i + 1) x (Γ i tt y)*  
**by** (*metis assms(2) assms(3) conn-zigzag1-var icat.st-local icid.src-comp-aux singleton-iff*)  
**have** *h3*:  $\partial i \text{ tt } (\Gamma i \text{ tt } x \otimes_{(i+1)} \sigma i x) = x$   
**by** (*metis assms(1) local.conn-face1 local.conn-face2 local.icat.sscatml.r0-absorb*)  
**have**  $\partial i \text{ ff } (x \otimes_{(i+1)} \Gamma i \text{ tt } y) = \partial i \text{ ff } x \otimes_{(i+1)} \partial i \text{ ff } (\Gamma i \text{ tt } y)$   
**using** *h2 local.face-func* **by** *simp*  
**hence** *h4*:  $\partial i \text{ ff } (x \otimes_{(i+1)} \Gamma i \text{ tt } y) = x \otimes_{(i+1)} \partial (i + 1) \text{ ff } y$   
**by** (*metis (full-types) assms(1) assms(2) local.conn-face4*)  
**have**  $\partial i \text{ tt } (\Gamma i \text{ tt } x \otimes_{(i+1)} \sigma i x) = \partial i \text{ tt } (\Gamma i \text{ tt } x) \otimes_{(i+1)} \partial i \text{ tt } (\sigma i x)$   
**using** *h1 local.face-func* **by** *simp*  
**also have**  $\dots = x \otimes_{(i+1)} \partial (i + 1) \text{ tt } x$   
**using** *calculation h3* **by** *simp*  
**thus** *?thesis*  
**using** *assms(3) h3 h4 local.icat.sts-msg.st-local* **by** *simp*  
**qed**

**lemma** *conn-corner1-var*:  $\Gamma \Gamma \ i \ \text{tt } (\partial i \ \alpha \ x \odot_{(i+1)} \partial i \ \beta \ y) = (\Gamma \ i \ \text{tt } (\partial i \ \alpha \ x) \odot_{(i+1)} \sigma \ i \ (\partial i \ \alpha \ x)) \star_i (\partial i \ \alpha \ x \odot_{(i+1)} \Gamma \ i \ \text{tt } (\partial i \ \beta \ y))$   
**proof** (*cases DD (i + 1) (∂ i α x) (∂ i β y)*)  
**case** *True*  
**have** *h1*: *DD (i + 1) (Γ i tt (∂ i α x)) (σ i (∂ i α x))*  
**by** (*metis local.conn-face2 local.face-compat-var local.locality*)

**have**  $h2: DD (i + 1) (\partial i \alpha x) (\Gamma i tt (\partial i \beta y))$   
**by** (*metis True icid.src-comp-aux insertCI local.conn-zigzag1-var local.iDst local.locality*)  
**have**  $h3: DD i (\Gamma i tt (\partial i \alpha x) \otimes_{(i+1)} \sigma i (\partial i \alpha x)) (\partial i \alpha x \otimes_{(i+1)} \Gamma i tt (\partial i \beta y))$   
**using** *True local.conn-corner1-DD local.face-compat-var* **by** *simp*  
**have**  $\Gamma \Gamma i tt (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = \Gamma \Gamma i tt \{\partial i \alpha x \otimes_{(i+1)} \partial i \beta y\}$   
**using** *True local.icat.pcomp-def-var4* **by** *simp*  
**also have**  $\dots = \{(\Gamma i tt (\partial i \alpha x) \otimes_{(i+1)} \sigma i (\partial i \alpha x)) \otimes_i (\partial i \alpha x \otimes_{(i+1)} \Gamma i tt (\partial i \beta y))\}$   
**using** *True local.conn-corner1 local.face-compat-var* **by** *simp*  
**also have**  $\dots = \{\Gamma i tt (\partial i \alpha x) \otimes_{(i+1)} \sigma i (\partial i \alpha x)\} \star_i \{\partial i \alpha x \otimes_{(i+1)} \Gamma i tt (\partial i \beta y)\}$   
**using**  $h3$  *local.icat.pcomp-def-var4 local.icid.stopp.conv-atom* **by** *simp*  
**also have**  $\dots = (\Gamma i tt (\partial i \alpha x) \odot_{(i+1)} \sigma i (\partial i \alpha x)) \star_i (\partial i \alpha x \odot_{(i+1)} \Gamma i tt (\partial i \beta y))$   
**using**  $h1 h2$  *local.icat.pcomp-def-var4* **by** *simp*  
**finally show** *?thesis*.  
**next**  
**case** *False*  
**thus** *?thesis*  
**by** (*smt (z3) Union-empty empty-is-image icat.st-local icid.ts-compat local.conn-face4 local.face-comm-var local.icid.stopp.ts-compat multimagma.conv-distl*)  
**qed**

**lemma** *conn-corner1-lift-aux*:  $fFx i x \implies \partial (i + 1) ff (\Gamma i tt x) = \partial (i + 1) ff x$   
**by** (*metis conn-zigzag1-var empty-not-insert equalsOI icid.src-comp-aux singletonD*)

**lemma** *conn-corner1-lift*:

**assumes**  $FFx i X$   
**and**  $FFx i Y$   
**shows**  $\Gamma \Gamma i tt (X \star_{(i+1)} Y) = (\Gamma \Gamma i tt X \star_{(i+1)} \sigma \sigma i X) \star_i (X \star_{(i+1)} \Gamma \Gamma i tt Y)$   
**proof**–  
**have**  $h1: \forall y \in Y. \partial (i + 1) ff (\Gamma i tt y) = \partial (i + 1) ff y$   
**by** (*metis assms(2) conn-zigzag1-var local.icid.ts-msg.tgt-comp-aux singletonI*)  
**have**  $h2: \forall xa \in X. DD (i + 1) (\Gamma i tt xa) (\sigma i xa) \longrightarrow \partial i tt (\Gamma i tt xa \otimes_{(i+1)} \sigma i xa) = \partial i tt (\Gamma i tt xa) \otimes_{(i+1)} \partial i tt (\sigma i xa)$   
**by** (*simp add: local.face-func*)  
**have**  $h3: \forall xc \in X. \forall y \in Y. DD (i + 1) xc (\Gamma i tt y) \longrightarrow \partial i ff (xc \otimes_{(i+1)} \Gamma i tt y) = \partial i ff xc \otimes_{(i+1)} \partial i ff (\Gamma i tt y)$   
**by** (*simp add: local.face-func*)  
**have**  $h4: \forall xa \in X. \partial i tt (\Gamma i tt xa) \otimes_{(i+1)} \partial i tt (\sigma i xa) = xa \otimes_{(i+1)} \partial i tt (\partial (i + 1) tt xa)$   
**by** (*smt (z3) assms(1) local.conn-face1 local.fFx-prop local.face-comm-var local.sym-face1-var1 local.sym-fix-var1*)



**have**  $h5: \forall xc \in X. \forall y \in Y. \partial i \text{ ff } xc \otimes_{(i+1)} \partial i \text{ ff } (\Gamma i \text{ tt } y) = xc \otimes_{(i+1)} \partial$   
 $(i+1) \text{ ff } y$   
**by** (*metis (full-types) assms(1) assms(2) local.conn-face4*)  
**have**  $h6: \forall xc \in X. \forall y \in Y. DD (i+1) xc (\partial (i+1) \text{ ff } y) \longrightarrow xc \otimes_{(i+1)} \partial$   
 $(i+1) \text{ ff } y = xc$   
**by** (*metis local.face-compat-var local.icat.sscatml.r0-absorb local.icid.stopp.Dst*)  
**have**  $h7: \forall xa \in X. xa \otimes_{(i+1)} \partial i \text{ tt } (\partial (i+1) \text{ tt } xa) = xa$   
**by** (*metis assms(1) local.face-comm-var local.face-compat-var local.icat.sscatml.r0-absorb*)  
**have**  $h8: \forall x \in X. \forall y \in Y. DD (i+1) xy \longrightarrow (\Gamma i \text{ tt } x \otimes_{(i+1)} \sigma i x) \otimes_i (x$   
 $\otimes_{(i+1)} \Gamma i \text{ tt } y) = \Gamma i \text{ tt } (x \otimes_{(i+1)} y)$   
**using** *assms(1) assms(2) local.conn-corner1 by auto*  
**have**  $(\Gamma \Gamma i \text{ tt } X \star_{(i+1)} \sigma \sigma i X) \star_i (X \star_{(i+1)} \Gamma \Gamma i \text{ tt } Y) = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)}$   
 $\sigma i xb) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xb \text{ } xc \text{ } y. xa \in X \wedge xb \in X \wedge xc \in X \wedge y \in$   
 $Y \wedge DD (i+1) (\Gamma i \text{ tt } xa) (\sigma i xb) \wedge DD (i+1) xc (\Gamma i \text{ tt } y) \wedge DD i (\Gamma i \text{ tt } xa$   
 $\otimes_{(i+1)} \sigma i xb) (xc \otimes_{(i+1)} \Gamma i \text{ tt } y)\}$   
**unfolding** *local.iconv-prop by blast*  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xb) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xb \text{ } xc$   
 $y. xa \in X \wedge xb \in X \wedge xc \in X \wedge y \in Y \wedge \partial (i+1) \text{ tt } (\Gamma i \text{ tt } xa) = \partial (i+1) \text{ ff}$   
 $(\sigma i xb) \wedge \partial (i+1) \text{ tt } xc = \partial (i+1) \text{ ff } (\Gamma i \text{ tt } y) \wedge \partial i \text{ tt } (\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma$   
 $i xb) = \partial i \text{ ff } (xc \otimes_{(i+1)} \Gamma i \text{ tt } y)\}$   
**using** *icat.st-local by presburger*  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xb) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xb \text{ } xc$   
 $y. xa \in X \wedge xb \in X \wedge xc \in X \wedge y \in Y \wedge xa = xb \wedge \partial (i+1) \text{ tt } xc = \partial (i+1)$   
 $\text{ ff } (\Gamma i \text{ tt } y) \wedge \partial i \text{ tt } (\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xb) = \partial i \text{ ff } (xc \otimes_{(i+1)} \Gamma i \text{ tt } y)\}$   
**by** (*smt (verit) Collect-cong assms(1) local.conn-face2 local.sym-type-var*)  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xa) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xc \text{ } y.$   
 $xa \in X \wedge xc \in X \wedge y \in Y \wedge \partial (i+1) \text{ tt } xc = \partial (i+1) \text{ ff } y \wedge \partial i \text{ tt } (\Gamma i \text{ tt } xa$   
 $\otimes_{(i+1)} \sigma i xa) = \partial i \text{ ff } (xc \otimes_{(i+1)} \Gamma i \text{ tt } y)\}$   
**by** (*smt (verit, best) Collect-cong assms(1) h1 local.conn-face3 local.locality*  
*local.sym-type-var*)  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xa) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xc \text{ } y.$   
 $xa \in X \wedge xc \in X \wedge y \in Y \wedge \partial (i+1) \text{ tt } xc = \partial (i+1) \text{ ff } y \wedge \partial i \text{ tt } (\Gamma i \text{ tt } xa$   
 $\otimes_{(i+1)} \partial i \text{ tt } (\sigma i xa) = \partial i \text{ ff } xc \otimes_{(i+1)} \partial i \text{ ff } (\Gamma i \text{ tt } y)\}$   
**by** (*smt (verit, del-insts) h2 h3 Collect-cong assms(1) h1 icat.st-local lo-*  
*cal.conn-face2 local.sym-type-var*)  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xa) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xc \text{ } y.$   
 $xa \in X \wedge xc \in X \wedge y \in Y \wedge \partial (i+1) \text{ tt } xc = \partial (i+1) \text{ ff } y \wedge xa \otimes_{(i+1)} \partial i$   
 $\text{ tt } (\partial (i+1) \text{ tt } xa) = xc \otimes_{(i+1)} \partial (i+1) \text{ ff } y\}$   
**by** (*smt (verit, del-insts) h4 h5 Collect-cong*)  
**also have**  $\dots = \{(\Gamma i \text{ tt } xa \otimes_{(i+1)} \sigma i xa) \otimes_i (xc \otimes_{(i+1)} \Gamma i \text{ tt } y) \mid xa \text{ } xc \text{ } y.$   
 $xa \in X \wedge xc \in X \wedge y \in Y \wedge \partial (i+1) \text{ tt } xc = \partial (i+1) \text{ ff } y \wedge xa = xc\}$   
**by** (*smt (z3) h6 h7 Collect-cong assms(2) icid.st-eq1 local.face-comm-var*)  
**also have**  $\dots = \{\Gamma i \text{ tt } (x \otimes_{(i+1)} y) \mid xy. x \in X \wedge y \in Y \wedge DD (i+1) xy\}$   
**by** (*smt (verit, ccfv-threshold) h8 Collect-cong icat.st-local*)  
**also have**  $\dots = \Gamma \Gamma i \text{ tt } (X \star_{(i+1)} Y)$

**unfolding** *local.iconv-prop* **by** *force*  
**finally show** *?thesis*  
**by** *simp*  
**qed**

**lemma** *conn-corner2-DD*:

**assumes** *fFx i x*  
**and** *fFx i y*  
**and** *DD (i + 1) x y*  
**shows** *DD i (Γ i ff x ⊗<sub>(i+1)</sub> y) (σ i y ⊗<sub>(i+1)</sub> Γ i ff y)*  
**proof** –  
**have** *h1: DD (i + 1) (Γ i ff x) y*  
**by** (*metis assms(1) assms(3) conn-zigzag1-var insertCI local.iDst local.icid.ts-msg.src-comp-aux local.locality*)  
**have** *h2: ∂ i ff (σ i y ⊗<sub>(i+1)</sub> Γ i ff y) = ∂ i ff (σ i y) ⊗<sub>(i+1)</sub> ∂ i ff (Γ i ff y)*  
**using** *assms(2) local.conn-face2 local.face-func local.locality local.sym-face3-simp*  
**by** *auto*  
**have** *∂ i tt (Γ i ff x ⊗<sub>(i+1)</sub> y) = ∂ i tt (Γ i ff x) ⊗<sub>(i+1)</sub> ∂ i tt y*  
**using** *h1 local.face-func* **by** *simp*  
**hence** *h4: ∂ i tt (Γ i ff x ⊗<sub>(i+1)</sub> y) = ∂ (i + 1) tt x ⊗<sub>(i+1)</sub> y*  
**by** (*metis (full-types) assms(1) assms(2) icid.st-eq1 local.conn-face4*)  
**thus** *?thesis*  
**by** (*metis h2 assms(2) assms(3) local.conn-face1 local.conn-face2 local.face-comm-var local.icid.stopp.Dst local.locality*)  
**qed**

**lemma** *conn-corner2-var*:  $\Gamma \Gamma i \text{ ff } (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = (\Gamma i \text{ ff } (\partial i \alpha x) \odot_{(i+1)} \partial i \beta y) \star_i (\sigma i (\partial i \beta y) \odot_{(i+1)} \Gamma i \text{ ff } (\partial i \beta y))$

**proof** (*cases DD (i + 1) (∂ i α x) (∂ i β y)*)  
**case** *True*  
**have** *h1: DD (i + 1) (Γ i ff (∂ i α x)) (∂ i β y)*  
**by** (*metis True insertCI local.conn-zigzag1-var local.iDst local.icid.ts-msg.src-comp-aux local.locality*)  
**have** *h2: DD (i + 1) (σ i (∂ i β y)) (Γ i ff (∂ i β y))*  
**by** (*metis local.conn-face2 local.face-compat-var local.locality*)  
**have** *h3: DD i (Γ i ff (∂ i α x) ⊗<sub>(i+1)</sub> ∂ i β y) (σ i (∂ i β y) ⊗<sub>(i+1)</sub> Γ i ff (∂ i β y))*  
**using** *True local.conn-corner2-DD* **by** *simp*  
**have**  $\Gamma \Gamma i \text{ ff } (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = \{\Gamma i \text{ ff } (\partial i \alpha x \otimes_{(i+1)} \partial i \beta y)\}$   
**by** (*metis (full-types) True local.icat.pcomp-def-var4 image-empty image-insert*)  
  
**also have**  $\dots = \{(\Gamma i \text{ ff } (\partial i \alpha x) \otimes_{(i+1)} (\partial i \beta y)) \otimes_i (\sigma i (\partial i \beta y) \otimes_{(i+1)} \Gamma i \text{ ff } (\partial i \beta y))\}$   
**using** *True conn-corner2 local.face-compat-var* **by** *simp*  
**also have**  $\dots = \{\Gamma i \text{ ff } (\partial i \alpha x) \otimes_{(i+1)} (\partial i \beta y)\} \star_i \{\sigma i (\partial i \beta y) \otimes_{(i+1)} \Gamma i \text{ ff } (\partial i \beta y)\}$   
**using** *h3 local.icat.pcomp-def-var4 local.icid.stopp.conv-atom* **by** *simp*

**also have**  $\dots = (\Gamma \text{ i ff } (\partial \text{ i } \alpha \text{ x}) \odot_{(i+1)} (\partial \text{ i } \beta \text{ y})) \star_i \{\sigma \text{ i } (\partial \text{ i } \beta \text{ y}) \otimes_{(i+1)} \Gamma \text{ i ff } (\partial \text{ i } \beta \text{ y})\}$   
**by** (*metis h1 local.icat.functionality-lem-var local.icat.pcomp-def local.icat.sscatml.r0-absorb local.it-absorb*)  
**also have**  $\dots = (\Gamma \text{ i ff } (\partial \text{ i } \alpha \text{ x}) \odot_{(i+1)} (\partial \text{ i } \beta \text{ y})) \star_i (\sigma \text{ i } (\partial \text{ i } \beta \text{ y}) \odot_{(i+1)} \Gamma \text{ i ff } (\partial \text{ i } \beta \text{ y}))$   
**using** *h2 local.icat.pcomp-def-var4* **by** *simp*  
**finally show** *?thesis*.  
**next**  
**case** *False*  
**thus** *?thesis*  
**by** (*metis UN-empty add-eq-self-zero empty-is-image local.conn-face1 local.face-compat-var local.pcomp-face-func-DD multimagma.conv-def nat-neq-iff zero-less-one*)  
**qed**

**lemma** *conn-corner2-lift*:

**assumes** *FFx i X*

**and** *FFx i Y*

**shows**  $\Gamma \Gamma \text{ i ff } (X \star_{(i+1)} Y) = (\Gamma \Gamma \text{ i ff } X \star_{(i+1)} Y) \star_i (\sigma \sigma \text{ i } Y \star_{(i+1)} \Gamma \Gamma \text{ i ff } Y)$

**proof** –

**have** *h1*:  $\forall x \in X. \forall ya \in Y. \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \longrightarrow \partial \text{ i tt } (\Gamma \text{ i ff } x \otimes_{(i+1)} ya) = \partial \text{ i tt } (\Gamma \text{ i ff } x) \otimes_{(i+1)} \partial \text{ i tt } ya$

**by** (*metis local.face-func add commute add-diff-cancel-right' assms(1) bot-nat-0.extremum-unique cancel-comm-monoid-add-class.diff-cancel conn-zigzag1-var empty-not-insert ex-in-conv icat.st-local local.icid.ts-msg.src-comp-aux not-one-le-zero singletonD*)

**have** *h2*:  $\forall yb \in Y. DD (i+1) (\sigma \text{ i } yb) (\Gamma \text{ i ff } yb) \longrightarrow \partial \text{ i ff } (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb) = \partial \text{ i ff } (\sigma \text{ i } yb) \otimes_{(i+1)} \partial \text{ i ff } (\Gamma \text{ i ff } yb)$

**by** (*simp add: local.face-func*)

**have** *h3*:  $\forall x \in X. \forall y \in Y. DD (i+1) x y \longrightarrow (\Gamma \text{ i ff } x \otimes_{(i+1)} y) \otimes_i (\sigma \text{ i } y \otimes_{(i+1)} \Gamma \text{ i ff } y) = \Gamma \text{ i ff } (x \otimes_{(i+1)} y)$

**using** *assms local.conn-corner2* **by** *simp*

**have** *h4*:  $\forall x \in X. \forall ya \in Y. (\partial (i+1) \text{ tt } (\Gamma \text{ i ff } x) = \partial (i+1) \text{ ff } ya) = (\partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya)$

**by** (*metis assms(1) conn-zigzag1-var local.icid.ts-msg.src-comp-aux singletonI*)

**have** *h5*:  $\forall yb \in Y. \forall yc \in Y. (\partial (i+1) \text{ tt } (\sigma \text{ i } yb) = \partial (i+1) \text{ ff } (\Gamma \text{ i ff } yc)) = (yb = yc)$

**by** (*metis assms(2) local.conn-face2 local.inv-sym-sym local.sym-face3-simp*)

**have** *h6*:  $\forall x \in X. \forall ya \in Y. \forall yb \in Y. (x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge \partial \text{ i tt } (\Gamma \text{ i ff } x \otimes_{(i+1)} ya) = \partial \text{ i ff } (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb)) = (x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge \partial \text{ i tt } (\Gamma \text{ i ff } x) \otimes_{(i+1)} \partial \text{ i tt } ya = \partial \text{ i ff } (\sigma \text{ i } yb) \otimes_{(i+1)} \partial \text{ i ff } (\Gamma \text{ i ff } yb))$

**using** *h1 h2 h5 icat.st-local* **by** *force*

**have**  $(\Gamma \Gamma \text{ i ff } X \star_{(i+1)} Y) \star_i (\sigma \sigma \text{ i } Y \star_{(i+1)} \Gamma \Gamma \text{ i ff } Y) = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yc) \mid x ya yb yc. x \in X \wedge ya \in Y \wedge yb \in Y \wedge yc \in Y \wedge DD (i+1) (\Gamma \text{ i ff } x) ya \wedge DD (i+1) (\sigma \text{ i } yb) (\Gamma \text{ i ff } yc) \wedge DD \text{ i } (\Gamma \text{ i ff } x \otimes_{(i+1)} ya) (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yc)\}$

**unfolding** *local.iconv-prop* **by** *fastforce*  
**also have**  $\dots = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yc) \mid x \text{ ya } yb \text{ yc. } x \in X \wedge ya \in Y \wedge yb \in Y \wedge yc \in Y \wedge \partial (i+1) \text{ tt } (\Gamma \text{ i ff } x) = \partial (i+1) \text{ ff } ya \wedge \partial (i+1) \text{ tt } (\sigma \text{ i } yb) = \partial (i+1) \text{ ff } (\Gamma \text{ i ff } yc) \wedge \partial i \text{ tt } (\Gamma \text{ i ff } x \otimes_{(i+1)} ya) = \partial i \text{ ff } (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yc)\}$   
**using** *icat.st-local* **by** *simp*  
**also have**  $\dots = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb) \mid x \text{ ya } yb. x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge \partial i \text{ tt } (\Gamma \text{ i ff } x \otimes_{(i+1)} ya) = \partial i \text{ ff } (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb)\}$   
**using** *h4 h5* **by** (*smt (verit, del-insts) Collect-cong*)  
**also have**  $\dots = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb) \mid x \text{ ya } yb. x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge \partial i \text{ tt } (\Gamma \text{ i ff } x) \otimes_{(i+1)} \partial i \text{ tt } ya = \partial i \text{ ff } (\sigma \text{ i } yb) \otimes_{(i+1)} \partial i \text{ ff } (\Gamma \text{ i ff } yb)\}$   
**using** *h6* **by** *fastforce*  
**also have**  $\dots = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb) \mid x \text{ ya } yb. x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge \partial (i+1) \text{ tt } x \otimes_{(i+1)} ya = \partial (i+1) \text{ ff } yb \otimes_{(i+1)} yb\}$   
**by** (*smt (z3) Collect-cong assms(1) assms(2) icid.st-eq1 local.conn-face1 local.conn-face4 local.conn-face2 local.face-comm-var*)  
**also have**  $\dots = \{(\Gamma \text{ i ff } x \otimes_{(i+1)} ya) \otimes_i (\sigma \text{ i } yb \otimes_{(i+1)} \Gamma \text{ i ff } yb) \mid x \text{ ya } yb. x \in X \wedge ya \in Y \wedge yb \in Y \wedge \partial (i+1) \text{ tt } x = \partial (i+1) \text{ ff } ya \wedge ya = yb\}$   
**by** *force*  
**also have**  $\dots = \{\Gamma \text{ i ff } (x \otimes_{(i+1)} y) \mid x \text{ y. } x \in X \wedge y \in Y \wedge DD (i+1) x \text{ y}\}$   
**by** (*smt (verit, ccfv-threshold) h3 Collect-cong icat.st-local*)  
**also have**  $\dots = \Gamma \Gamma \text{ i ff } (X \star_{(i+1)} Y)$   
**unfolding** *local.iconv-prop* **by** *force*  
**finally show** *?thesis*  
**by** *simp*  
**qed**

**lemma** *conn-corner3-var*:

**assumes**  $j \neq i \wedge j \neq i + 1$   
**shows**  $\Gamma \Gamma \text{ i } \alpha (\partial \text{ i } \beta x \odot_j \partial \text{ i } \gamma y) = \Gamma \text{ i } \alpha (\partial \text{ i } \beta x) \odot_j \Gamma \text{ i } \alpha (\partial \text{ i } \gamma y)$   
**by** (*smt (z3) assms empty-is-image image-insert local.conn-corner3 local.conn-face1 local.conn-face3 local.face-commat-var local.iDst local.icat.pcomp-def-var4 local.locality local.pcomp-face-func-DD*)

**lemma** *conn-corner3-lift*:

**assumes**  $j \neq i$   
**and**  $j \neq i + 1$   
**and**  $FFx \text{ i } X$   
**and**  $FFx \text{ i } Y$   
**shows**  $\Gamma \Gamma \text{ i } \alpha (X \star_j Y) = \Gamma \Gamma \text{ i } \alpha X \star_j \Gamma \Gamma \text{ i } \alpha Y$

**proof** –

**have**  $h: \forall x \in X. \forall y \in Y. DD \text{ j } (\Gamma \text{ i } \alpha x) (\Gamma \text{ i } \alpha y) = DD \text{ j } x \text{ y}$   
**by** (*metis assms icat.st-local local.conn-face1 local.conn-face3 local.face-comm-var*)  
**have**  $\Gamma \Gamma \text{ i } \alpha X \star_j \Gamma \Gamma \text{ i } \alpha Y = \{\Gamma \text{ i } \alpha x \otimes_j \Gamma \text{ i } \alpha y \mid x \text{ y. } x \in X \wedge y \in Y \wedge DD$

$j (\Gamma i \alpha x) (\Gamma i \alpha y)$   
**unfolding** *local.iconv-prop* **by** *force*  
**also have**  $\dots = \{\Gamma i \alpha x \otimes_j \Gamma i \alpha y \mid x y. x \in X \wedge y \in Y \wedge DD j x y\}$   
**using** *h* **by** *force*  
**also have**  $\dots = \{\Gamma i \alpha (x \otimes_j y) \mid x y. x \in X \wedge y \in Y \wedge DD j x y\}$   
**using** *conn-corner3* *assms* **by** *fastforce*  
**also have**  $\dots = \Gamma \Gamma i \alpha (X \star_j Y)$   
**unfolding** *local.iconv-prop* **by** *force*  
**finally show** *?thesis*  
**by** *simp*  
**qed**

**lemma** *conn-face5* [*simp*]:  $\partial (j + 1) \alpha (\Gamma j (-\alpha) (\partial j \gamma x)) = \partial (j + 1) \alpha (\partial j \gamma x)$   
**by** (*smt* (*verit*, *ccfv-SIG*) *icid.s-absorb-var* *local.conn-corner1-lift-aux* *local.conn-zigzag1-var* *local.face-compat-var* *local.icid.ts-msg.src-comp-cond* *local.is-absorb* *singleton-insert-inj-eq'*)

**lemma** *conn-inv-sym-braid*:  
**assumes** *diffSup* *i j 2*  
**shows**  $\Gamma i \alpha (\vartheta j (\partial i \beta (\partial (j + 1) \gamma x))) = \vartheta j (\Gamma i \alpha (\partial i \beta (\partial (j + 1) \gamma x)))$   
**by** (*smt* (*z3*) *add-diff-cancel-left'* *assms* *diff-add-0* *diff-is-0-eq'* *local.conn-face3* *local.conn-sym-braid* *local.face-comm-var* *local.face-compat-var* *local.inv-sym-face2* *local.inv-sym-sym-var1* *local.inv-sym-type-var* *local.sym-inv-sym* *nat-1-add-1* *nle-le* *rel-simps(28)*)

**lemma** *conn-corner4*:  $\Gamma \Gamma i tt (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = (\Gamma i tt (\partial i \alpha x) \odot_i \partial i \alpha x) \star_{(i+1)} (\sigma i (\partial i \alpha x) \odot_i \Gamma i tt (\partial i \beta y))$   
**proof** (*cases* *DD* (*i + 1*) ( $\partial i \alpha x$ ) ( $\partial i \beta y$ ))  
**case** *True*  
**have** *h1*:  $\partial \partial (i+1) tt (\Gamma i tt (\partial i \alpha x) \odot_i \partial i \alpha x) = \{\sigma i (\partial i \alpha x)\}$   
**by** (*metis* *image-empty* *image-insert* *local.conn-face1* *local.conn-face2* *local.face-compat-var* *local.it-absorb*)  
**have**  $\partial (i+1) tt (\partial i \alpha x) = \partial (i+1) ff (\partial i \beta y)$   
**using** *True* *local.iDst* **by** *simp*  
**hence** *h2*:  $\partial \partial (i+1) ff (\sigma i (\partial i \alpha x) \odot_i \Gamma i tt (\partial i \beta y)) = \{\sigma i (\partial i \alpha x)\}$   
**by** (*smt* (*z3*) *add-eq-self-zero* *conn-face4* *conn-face5* *icat.st-local* *image-is-empty* *local.comp-face-func* *local.conn-face2* *local.face-comm-var* *local.face-compat-var* *local.it-absorb* *subset-singletonD* *zero-neq-one*)  
**hence**  $\partial \partial (i+1) tt (\Gamma i tt (\partial i \alpha x) \odot_i \partial i \alpha x) \cap \partial \partial (i+1) ff (\sigma i (\partial i \alpha x) \odot_i \Gamma i tt (\partial i \beta y)) \neq \{\}$   
**using** *h1* **by** *simp*  
**thus** *?thesis*  
**by** (*smt* (*z3*) *True* *add-cancel-right-right* *dual-order.eq-iff* *empty-is-image* *h1* *h2* *icat.locality-lifting* *local.conn-corner1-var* *local.icat.pcomp-def-var4* *local.interchange-var* *multimagma.conv-atom* *not-one-le-zero*)  
**next**  
**case** *False*  
**thus** *?thesis*  
**by** (*smt* (*z3*) *Union-empty* *add-eq-self-zero* *dual-order.eq-iff* *icat.st-local* *im-*

*age-empty local.conn-face4 local.conn-face2 local.face-comm-var local.face-compat-var  
multimagma.conv-distl not-one-le-zero)*

**qed**

**lemma** *conn-corner5*:  $\Gamma \Gamma i \text{ ff } (\partial i \alpha x \odot_{(i+1)} \partial i \beta y) = (\Gamma i \text{ ff } (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y)) \star_{(i+1)} (\partial i \beta y \odot_i \Gamma i \text{ ff } (\partial i \beta y))$

**proof** (*cases DD (i + 1) (\partial i \alpha x) (\partial i \beta y)*)

**case** *True*

**have** *h1*:  $\partial \partial (i+1) \text{ ff } (\partial i \beta y \odot_i \Gamma i \text{ ff } (\partial i \beta y)) = \{\sigma i (\partial i \beta y)\}$

**by** (*metis image-empty image-insert local.conn-face1 local.conn-face2 local.face-compat-var local.is-absorb*)

**have**  $\partial (i+1) \text{ tt } (\partial i \alpha x) = \partial (i+1) \text{ ff } (\partial i \beta y)$

**using** *True local.iDst* **by** *simp*

**hence** *h2*:  $\partial \partial (i+1) \text{ tt } (\Gamma i \text{ ff } (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y)) = \{\sigma i (\partial i \beta y)\}$

**by** (*smt (z3) conn-face4 conn-face5 h1 icat.st-local image-insert image-is-empty local.comp-face-func local.conn-face2 local.face-comm-var local.face-compat-var local.icat.functionality-lem-var local.it-absorb subset-singletonD*)

**hence**  $\partial \partial (i+1) \text{ ff } (\partial i \beta y \odot_i \Gamma i \text{ ff } (\partial i \beta y)) \cap \partial \partial (i+1) \text{ tt } (\Gamma i \text{ ff } (\partial i \alpha x) \odot_i \sigma i (\partial i \beta y)) \neq \{\}$

**using** *h1* **by** *simp*

**thus** *?thesis*

**by** (*smt (z3) True add-cancel-right-right dual-order.eq-iff empty-is-image h1 h2 icat.locality-lifting local.conn-corner2-var local.icat.functionality-lem-var local.interchange-var multimagma.conv-atom not-one-le-zero*)

**next**

**case** *False*

**thus** *?thesis*

**by** (*smt (z3) UN-empty add-cancel-right-right dual-order.eq-iff image-empty local.conn-face2 local.face-compat-var local.pcomp-face-func-DD local.sym-func2-DD local.sym-type-var multimagma.conv-def not-one-le-zero*)

**qed**

**lemma** *conn-corner3-alt*:  $j \neq i \implies j \neq i + 1 \implies \Gamma \Gamma i \alpha (\partial i \beta x \odot_j \partial i \gamma y) = \Gamma i \alpha (\partial i \beta x) \odot_j \Gamma i \alpha (\partial i \gamma y)$

**by** (*simp add: local.conn-corner3-var*)

**lemma** *conn-shift2*:

**assumes** *fFx i x*

**and** *fFx (i + 2) x*

**shows**  $\vartheta i (\vartheta (i + 1) (\Gamma i \alpha x)) = \Gamma (i + 1) \alpha (\vartheta (i + 1) x)$

**proof**–

**have**  $\Gamma i \alpha x = \sigma (i + 1) (\sigma i (\Gamma (i + 1) \alpha (\vartheta (i + 1) x)))$

**using** *assms local.conn-shift local.inv-sym-face2 local.inv-sym-face3-simp local.sym-inv-sym* **by** *simp*

**thus** *?thesis*

**using** *assms local.conn-face3 local.inv-sym-face2 local.inv-sym-sym local.inv-sym-type-var local.sym-type-var* **by** *simp*

**qed**

end

end

## 5 Cubical $(\omega, 0)$ -Categories with Connections

**theory** *CubicalOmegaZeroCategoriesConnections*

**imports** *CubicalCategoriesConnections*

**begin**

All categories considered in this component are single-set categories.

First we define shell-invertibility.

**abbreviation** (in *cubical-omega-category-connections*) *ri-inv i x y*  $\equiv (DD\ i\ x\ y \wedge DD\ i\ y\ x \wedge x \otimes_i y = \partial\ i\ ff\ x \wedge y \otimes_i x = \partial\ i\ tt\ x)$

**abbreviation** (in *cubical-omega-category-connections*) *ri-inv-shell k i x*  $\equiv (\forall j\ \alpha.\ j + 1 \leq k \wedge j \neq i \longrightarrow (\exists y.\ ri\text{-inv}\ i\ (\partial\ j\ \alpha\ x)\ y))$

Next we define the class of cubical  $(\omega, 0)$ -categories with connections.

**class** *cubical-omega-zero-category-connections* = *cubical-omega-category-connections* +  
**assumes** *ri-inv*:  $k \geq 1 \implies i \leq k - 1 \implies dim\text{-bound}\ k\ x \implies ri\text{-inv-shell}\ k\ i\ x \implies \exists y.\ ri\text{-inv}\ i\ x\ y$

**begin**

Finally, to show our axiomatisation at work we prove Proposition 2.4.7 from our companion paper, namely that every cell in an  $(\omega, 0)$ -category is ri-invertible for each natural number  $i$ . This requires some background theory engineering.

**lemma** *ri-inv-fix*:

**assumes** *fFx i x*

**shows**  $\exists y.\ ri\text{-inv}\ i\ x\ y$

**by** (*metis assms icat.st-local local.face-compat-var local.icat.sscatml.l0-absorb*)

**lemma** *ri-inv2*:

**assumes**  $k \geq 1$

**assumes** *dim-bound k x*

**and** *ri-inv-shell k i x*

**shows**  $\exists y.\ ri\text{-inv}\ i\ x\ y$

**proof** (*cases i ≤ k - 1*)

**case** *True*

**thus** *?thesis*

**using** *assms local.ri-inv* **by** *simp*

**next**

**case** *False*  
**hence**  $fFx\ i\ x$   
**using** *assms(2)* **by** *fastforce*  
**thus** *?thesis*  
**using** *ri-inv-fix* **by** *simp*  
**qed**

**lemma** *ri-inv3*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $\exists y. ri\text{-}inv\ i\ x\ y$   
**proof** (*cases k = 0*)  
**case** *True*  
**thus** *?thesis*  
**using** *assms(1) less-eq-nat.simps(1) ri-inv-fix* **by** *simp*  
**next**  
**case** *False*  
**hence**  $k \geq 1$   
**by** *simp*  
**thus** *?thesis*  
**using** *assms ri-inv2* **by** *simp*  
**qed**

**lemma** *ri-unique*:  $(\exists y. ri\text{-}inv\ i\ x\ y) = (\exists!y. ri\text{-}inv\ i\ x\ y)$   
**by** (*metis local.icat.pcomp-assoc local.icat.sscatml.assoc-defined local.icat.sscatml.l0-absorb local.icat.sts-msg.st-local local.pcomp-uface*)

**lemma** *ri-unique-var*:  $ri\text{-}inv\ i\ x\ y \implies ri\text{-}inv\ i\ x\ z \implies y = z$   
**using** *ri-unique* **by** *fastforce*

**definition**  $ri\ i\ x = (THE\ y. ri\text{-}inv\ i\ x\ y)$

**lemma** *ri-inv-ri*:  $ri\text{-}inv\ i\ x\ y \implies (y = ri\ i\ x)$   
**proof**–  
**assume**  $a: ri\text{-}inv\ i\ x\ y$   
**hence**  $\exists!y. ri\text{-}inv\ i\ x\ y$   
**using** *ri-unique* **by** *blast*  
**thus**  $y = ri\ i\ x$   
**unfolding** *ri-def*  
**by** (*smt (verit, ccfv-threshold) a the-equality*)  
**qed**

**lemma** *ri-def-prop*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $DD\ i\ x\ (ri\ i\ x) \wedge DD\ i\ (ri\ i\ x)\ x \wedge x \otimes_i (ri\ i\ x) = \partial\ i\ ff\ x \wedge (ri\ i\ x) \otimes_i x = \partial\ i\ tt\ x$   
**proof**–  
**have**  $\exists y. ri\text{-}inv\ i\ x\ y$



**using** *assms ri-inv3* **by** *blast*  
**hence**  $\exists!y. DD\ i\ x\ y \wedge DD\ i\ y\ x \wedge x \otimes_i y = \partial\ i\ ff\ x \wedge y \otimes_i x = \partial\ i\ tt\ x$   
**by** (*simp add: ri-unique*)  
**hence**  $DD\ i\ x\ (ri\ i\ x) \wedge DD\ i\ (ri\ i\ x)\ x \wedge x \otimes_i (ri\ i\ x) = \partial\ i\ ff\ x \wedge (ri\ i\ x) \otimes_i$   
 $x = \partial\ i\ tt\ x$   
**unfolding** *ri-def* **by** (*smt (verit, del-insts) theI'*)  
**thus** *?thesis*  
**by** *simp*  
**qed**

**lemma** *ri-right*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $x \otimes_i ri\ i\ x = \partial\ i\ ff\ x$   
**using** *assms ri-def-prop* **by** *simp*

**lemma** *ri-right-set*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $x \odot_i ri\ i\ x = \{\partial\ i\ ff\ x\}$   
**using** *assms local.icat.pcomp-def-var3 ri-def-prop* **by** *blast*

**lemma** *ri-left*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $ri\ i\ x \otimes_i x = \partial\ i\ tt\ x$   
**using** *assms ri-def-prop* **by** *simp*

**lemma** *ri-left-set*:  
**assumes** *dim-bound k x*  
**and** *ri-inv-shell k i x*  
**shows**  $ri\ i\ x \odot_i x = \{\partial\ i\ tt\ x\}$   
**using** *assms local.icat.pcomp-def-var3 ri-def-prop* **by** *blast*

**lemma** *dim-face*:  $dim\text{-bound}\ k\ x \implies dim\text{-bound}\ k\ (\partial\ i\ \alpha\ x)$   
**by** (*metis local.double-fix-prop local.face-comm-var*)

**lemma** *dim-ri-inv*:  
**assumes** *dim-bound k x*  
**and** *ri-inv i x y*  
**shows** *dim-bound k y*

**proof** –

**{fix**  $l\ \alpha$   
**assume**  $ha: l \geq k$   
**have**  $h1: DD\ i\ x\ (\partial\ l\ \alpha\ y)$   
**by** (*smt (verit, ccfv-threshold) assms ha icat.st-local icid.s-absorb-var3 local.pcomp-face-func-DD*)  
**have**  $h2: DD\ i\ (\partial\ l\ \alpha\ y)\ x$   
**by** (*metis (full-types) assms ha icid.ts-compat local.iDst local.locality local.pcomp-face-func-DD*)

```

have  $\partial l \alpha (x \otimes_i y) = \partial l \alpha x \otimes_i \partial l \alpha y$ 
by (metis ha assms(1) assms(2) local.fFx-prop local.face-func local.icat.sscatml.r0-absorb
local.pcomp-uface)
hence h3:  $\partial l \alpha (x \otimes_i y) = x \otimes_i \partial l \alpha y$ 
by (metis assms(1) ha local.face-compat-var)
have  $\partial l \alpha (y \otimes_i x) = \partial l \alpha y \otimes_i \partial l \alpha x$ 
by (metis ha assms(1) assms(2) local.fFx-prop local.face-func local.icat.sscatml.r0-absorb
local.pcomp-uface)
hence  $\partial l \alpha (y \otimes_i x) = \partial l \alpha y \otimes_i x$ 
by (metis assms(1) ha local.face-compat-var)
hence ri-inv i x ( $\partial l \alpha y$ )
by (metis assms(1,2) h1 h2 h3 ha local.face-comm-var local.face-compat-var)
hence  $\partial l \alpha y = y$ 
using ri-unique-var assms(2) by blast}
thus ?thesis
by simp
qed

```

```

lemma every-dim-k-ri-inv:
assumes dim-bound k x
shows  $\forall i. \exists y. ri\text{-inv } i \ x \ y$  using <dim-bound k x>
proof (induct k arbitrary: x)
case 0
thus ?case
using ri-inv-fix by simp
next
case (Suc k)
{fix i
have  $\exists y. ri\text{-inv } i \ x \ y$ 
proof (cases Suc k  $\leq$  i)
case True
thus ?thesis
using Suc.prem1 ri-inv-fix by simp
next
case False
{fix j  $\alpha$ 
assume h:  $j \leq k \wedge j \neq i$ 
hence a: dim-bound k ( $\Sigma j (k - j) (\partial j \alpha x)$ )
by (smt (z3) Suc.prem2 antisym-conv2 le-add-diff-inverse local.face-comm-var
local.face-compat-var local.symcomp-face2 local.symcomp-type-var nle-le not-less-eq-eq)
have  $\exists y. ri\text{-inv } i \ (\partial j \alpha x) \ y$ 
proof (cases j < i)
case True
obtain y where b: ri-inv (i - 1) ( $\Sigma j (k - j) (\partial j \alpha x)$ ) y
using Suc.hyps a by force
have c: dim-bound k y
apply (rule dim-ri-inv[where x =  $\Sigma j (k - j) (\partial j \alpha x)$ ])
using a b by simp-all
hence d: DD i ( $\partial j \alpha x$ ) ( $\Theta j (k - j) y$ )

```

```

    by (smt (verit) False True a b h icid.ts-compat le-add-diff-inverse local.iDst
local.icid.stopp.ts-compat local.inv-symcomp-face1 local.inv-symcomp-symcomp lo-
cal.locality nle-le not-less-eq-eq)
    hence e: DD i (Θ j (k - j) y) (∂ j α x)
    by (smt (verit) False True b c dual-order.refl h icid.ts-compat le-add-diff-inverse
local.iDst local.icid.stopp.ts-compat local.inv-symcomp-face1 local.inv-symcomp-symcomp
local.locality local.symcomp-type-var not-less-eq-eq)
    have f: ∂ j α x ⊗i Θ j (k - j) y = Θ j (k - j) (Σ j (k - j) (∂ j α x)
⊗(i - 1) y)
    apply (subst inv-symcomp-comp4)
    using True local.symcomp-type-var1 c False One-nat-def b local.face-compat-var
local.inv-symcomp-symcomp a by auto
    have Θ j (k - j) y ⊗i ∂ j α x = Θ j (k - j) (y ⊗(i - 1) Σ j (k - j) (∂
j α x))
    apply (subst inv-symcomp-comp4)
    using True local.symcomp-type-var1 b c False local.face-compat-var
local.inv-symcomp-symcomp a by simp-all
    thus ?thesis
    by (metis False True b c d dual-order.refl e f h le-add-diff-inverse
local.icid.stopp.Dst local.inv-symcomp-face1 not-less-eq-eq)
next
case False
obtain y where b: ri-inv i (Σ j (k - j) (∂ j α x)) y
using Suc.hyps a by presburger
have c: dim-bound k y
apply (rule dim-ri-inv[where x = Σ j (k - j) (∂ j α x)])
using a b by simp-all
hence d: DD i (∂ j α x) (Θ j (k - j) y)
by (smt (verit) False a b dual-order.refl h icid.ts-compat le-add-diff-inverse
linorder-neqE-nat local.iDst local.icid.stopp.ts-compat local.inv-symcomp-face2 lo-
cal.inv-symcomp-symcomp local.locality)
hence e: DD i (Θ j (k - j) y) (∂ j α x)
by (smt (z3) False add commute b c dual-order.refl h le-add-diff-inverse2
linorder-neqE-nat local.face-comm-var local.face-compat-var local.iDst local.inv-symcomp-face2
local.inv-symcomp-symcomp local.locality local.symcomp-face2)
have f: ∂ j α x ⊗i Θ j (k - j) y = Θ j (k - j) (Σ j (k - j) (∂ j α x) ⊗i
y)
    apply (subst inv-symcomp-comp2)
    using False h nat-neq-iff local.symcomp-type-var1 b c a local.face-compat-var
local.inv-symcomp-symcomp by simp-all
    have Θ j (k - j) y ⊗i ∂ j α x = Θ j (k - j) (y ⊗i Σ j (k - j) (∂ j α x))
    apply (subst inv-symcomp-comp2)
    using False h a b c local.inv-symcomp-symcomp by simp-all
    thus ?thesis
    by (metis False antisym-conv3 b d e f h local.face-compat-var lo-
cal.inv-symcomp-face2 local.inv-symcomp-symcomp local.symcomp-type-var1)
qed}
thus ?thesis
apply (intro ri-inv[where k = k + 1])

```

```
      using False Suc.prems by simp-all
    qed}
  thus ?case
    by simp
qed
```

We can now show that every cell is ri-invertible in every direction i.

```
lemma every-ri-inv:  $\exists y. ri\text{-inv } i \ x \ y$ 
  using every-dim-k-ri-inv local.fin-fix by blast
```

```
end
```

```
end
```

## References

- [1] P. Malbos, T. Massacrier, and G. Struth. Single-set cubical categories and their formalisation with a proof assistant. 2024. <http://arxiv.org/abs/2401.10553v1>.
- [2] G. Struth. Catoids, categories, groupoids. *Arch. Formal Proofs*, 2023, 2023.