

# Higher Globular Catoids and Quantaes

Cameron Calk and Georg Struth

April 18, 2024

## Abstract

We formalise strict 2-catoids, 2-categories, 2-Kleene algebras and 2-quantaes, as well as their  $\omega$ -variants. Due to strictness, the cells of these higher categories have globular shape. We use a single-set approach, generalised to catoids and based on type classes. The higher Kleene algebras and quantaes introduced extend features of modal and concurrent Kleene algebras and quantaes. They arise for instance as powerset extensions of higher catoids, and have been used in algebraic confluence proofs in higher-dimensional rewriting. Details are described in two companion articles.

## Contents

<b>1</b>	<b>Introductory remarks</b>	<b>2</b>
<b>2</b>	<b>2-Catoids</b>	<b>2</b>
2.1	0-Structures and 1-structures . . . . .	3
2.2	2-Catoids . . . . .	6
2.3	2-Catoids and single-set 2-categories . . . . .	11
2.4	Reduced axiomatisations . . . . .	13
<b>3</b>	<b>2-Kleene algebras</b>	<b>22</b>
3.1	Copies for 0-structures . . . . .	22
3.2	Copies for 1-structures . . . . .	25
3.3	Globular 2-semirings . . . . .	27
3.4	Globular 2-Kleene algebras . . . . .	37
<b>4</b>	<b>2-Quantaes</b>	<b>39</b>
<b>5</b>	<b>Lifting 2-Catoids to powerset 2-quantaes</b>	<b>48</b>
<b>6</b>	<b>2-Catoids with (too) strong homomorphisms</b>	<b>50</b>
6.1	2-st-Multimagmas with strong homomorphism laws . . . . .	50
6.2	2-Catoids with (too) strong homomorphisms . . . . .	52
6.3	Single-set 2-categories with (too) strong homomorphisms . . .	52

6.4	2-lr-Multimagmas with strong interchange law . . . . .	53
<b>7</b>	<b><math>\omega</math>-Catoids</b>	<b>55</b>
7.1	Indexed catoids. . . . .	55
7.2	$\omega$ -Catoids . . . . .	57
7.3	$\omega$ -Catoids and single-set $\omega$ -categories . . . . .	63
7.4	Reduced axiomatisations . . . . .	64
<b>8</b>	<b><math>\omega</math>-Kleene algebras</b>	<b>68</b>
8.1	Copies for i-structures . . . . .	68
8.2	Globular $\omega$ -semirings . . . . .	70
8.3	Globular $\omega$ -Kleene algebras . . . . .	78
<b>9</b>	<b><math>\omega</math>-Quantales</b>	<b>79</b>
<b>10</b>	<b>Lifting <math>\omega</math>-catoids to powerset <math>\omega</math>-quantales</b>	<b>86</b>

## 1 Introductory remarks

We extend formalisations of catoids, categories and quantales from the AFP [5, 4, 3] to higher variants, as described in a companion article [2]. The categories, in particular, are formalised in a single-set approach. They are strict so that their cells have globular shape. We formalise the cases of 2 and  $\omega$  separately. First, strict 2-categories are important in category theory: the category of all small categories, for example, forms such a category. Second, strict  $\omega$ -categories are simply given by pairs of strict 2-categories in all dimensions, so that many properties for  $\omega$  generalise easily from 2-properties. Fourth, Isabelle’s Nitpick tool can find interesting counterexamples at dimension 2, but not for  $\omega$ . Finally, in the type classes formalising our  $\omega$ -structures, the numerical indices of higher operations cannot simply be instantiated to a fixed value such as 2. Applications of higher Kleene algebras and quantales in higher-dimensional rewriting are explained in [1], where these structures were introduced.

With higher catoids, the partial compositions of cells in higher categories are relaxed to multioperations, which assign each pair of elements to a set of elements, so that mapping to the empty set captures partiality. In addition, a composition of two elements may be undefined even though the target of the first equals the source of the second in a given dimension.

## 2 2-Catoids

```
theory Two-Catoid
  imports Catoids.Catoid
```

**begin**

We define 2-catoids and in particular (strict) 2-categories as local functional 2-catoids. With Isabelle we first need to make two copies of catoids for the 0-structure and 1-structure.

## 2.1 0-Structures and 1-structures.

**class** *multimagma0* =  
  **fixes** *mcomp0* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a set (**infixl**  $\odot_0$  70)

**begin**

**sublocale** *mm0*: *multimagma mcomp0*.

**abbreviation**  $\Delta_0 \equiv mm0.\Delta$

**abbreviation** *conv0* :: 'a set  $\Rightarrow$  'a set  $\Rightarrow$  'a set (**infixl**  $*_0$  70) **where**  
   $X *_0 Y \equiv mm0.conv X Y$

**lemma**  $X *_0 Y = (\bigcup x \in X. \bigcup y \in Y. x \odot_0 y)$   
  **by** (*simp add: mm0.conv-def*)

**end**

**class** *multimagma1* =  
  **fixes** *mcomp1* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a set (**infixl**  $\odot_1$  70)

**begin**

**sublocale** *mm1*: *multimagma mcomp1*.

**abbreviation**  $\Delta_1 \equiv mm1.\Delta$

**abbreviation** *conv1* :: 'a set  $\Rightarrow$  'a set  $\Rightarrow$  'a set (**infixl**  $*_1$  70) **where**  
   $X *_1 Y \equiv mm1.conv X Y$

**end**

**class** *multisemigroup0* = *multimagma0* +  
  **assumes** *assoc*:  $(\bigcup v \in y \odot_0 z. x \odot_0 v) = (\bigcup v \in x \odot_0 y. v \odot_0 z)$

**sublocale** *multisemigroup0*  $\subseteq$  *msg0*: *multisemigroup mcomp0*  
  **by** (*unfold-locales, simp add: local.assoc*)

**class** *multisemigroup1* = *multimagma1* +  
  **assumes** *assoc*:  $(\bigcup v \in y \odot_1 z. x \odot_1 v) = (\bigcup v \in x \odot_1 y. v \odot_1 z)$

**sublocale** *multisemigroup1*  $\subseteq$  *msg1*: *multisemigroup mcomp1*

```

    by (unfold-locales, simp add: local.assoc)

class st-multimagma0 = multimagma0 +
fixes  $\sigma_0 :: 'a \Rightarrow 'a$ 
    and  $\tau_0 :: 'a \Rightarrow 'a$ 
    assumes  $Dst0: x \odot_0 y \neq \{\} \implies \tau_0 x = \sigma_0 y$ 
    and  $src0-absorb [simp]: \sigma_0 x \odot_0 x = \{x\}$ 
    and  $tgt0-absorb [simp]: x \odot_0 \tau_0 x = \{x\}$ 

begin

sublocale  $stmm0: st-multimagma mcomp0 \sigma_0 \tau_0$ 
    by (unfold-locales, simp-all add: local.Dst0)

abbreviation  $s0fix \equiv stmm0.sfix$ 

abbreviation  $t0fix \equiv stmm0.tfix$ 

abbreviation  $Src_0 \equiv stmm0.Src$ 

abbreviation  $Tgt_0 \equiv stmm0.Tgt$ 

end

class st-multimagma1 = multimagma1 +
fixes  $\sigma_1 :: 'a \Rightarrow 'a$ 
    and  $\tau_1 :: 'a \Rightarrow 'a$ 
    assumes  $Dst1: x \odot_1 y \neq \{\} \implies \tau_1 x = \sigma_1 y$ 
    and  $src1-absorb [simp]: \sigma_1 x \odot_1 x = \{x\}$ 
    and  $tgt1-absorb [simp]: x \odot_1 \tau_1 x = \{x\}$ 

begin

sublocale  $stmm1: st-multimagma mcomp1 \sigma_1 \tau_1$ 
    by (unfold-locales, simp-all add: local.Dst1)

abbreviation  $s1fix \equiv stmm1.sfix$ 

abbreviation  $t1fix \equiv stmm1.tfix$ 

abbreviation  $Src_1 \equiv stmm1.Src$ 

abbreviation  $Tgt_1 \equiv stmm1.Tgt$ 

end

class catoid0 = st-multimagma0 + multisemigroup0

sublocale  $catoid0 \subseteq stmsg0: catoid mcomp0 \sigma_0 \tau_0..$ 

```

```

class catoid1 = st-multimagma1 + multisemigroup1

sublocale catoid1  $\subseteq$  stmsg1: catoid mcomp1  $\sigma_1 \tau_1$ ..

class local-catoid0 = catoid0 +
  assumes src0-local:  $\text{Src}_0(x \odot_0 \sigma_0 y) \subseteq \text{Src}_0(x \odot_0 y)$ 
  and tgt0-local:  $\text{Tgt}_0(\tau_0 x \odot_0 y) \subseteq \text{Tgt}_0(x \odot_0 y)$ 

class local-catoid1 = catoid1 +
  assumes l1-local:  $\text{Src}_1(x \odot_1 \sigma_1 y) \subseteq \text{Src}_1(x \odot_1 y)$ 
  and r1-local:  $\text{Tgt}_1(\tau_1 x \odot_1 y) \subseteq \text{Tgt}_1(x \odot_1 y)$ 

sublocale local-catoid0  $\subseteq$  ssmsg0: local-catoid mcomp0  $\sigma_0 \tau_0$ 
  apply unfold-locales using local.src0-local local.tgt0-local by auto

sublocale local-catoid1  $\subseteq$  stmsg1: local-catoid mcomp1  $\sigma_1 \tau_1$ 
  apply unfold-locales using local.l1-local local.r1-local by auto

class functional-magma0 = multimagma0 +
  assumes functionality0:  $x \in y \odot_0 z \implies x' \in y \odot_0 z \implies x = x'$ 

sublocale functional-magma0  $\subseteq$  pm0: functional-magma mcomp0
  by (unfold-locales, simp add: local.functionality0)

class functional-magma1 = multimagma1 +
  assumes functionality1:  $x \in y \odot_1 z \implies x' \in y \odot_1 z \implies x = x'$ 

sublocale functional-magma1  $\subseteq$  pm1: functional-magma mcomp1
  by (unfold-locales, simp add: local.functionality1)

class functional-semigroup0 = functional-magma0 + multisemigroup0

sublocale functional-semigroup0  $\subseteq$  psg0: functional-semigroup mcomp0..

class functional-semigroup1 = functional-magma1 + multisemigroup1

sublocale functional-semigroup1  $\subseteq$  psg1: functional-semigroup mcomp1..

class functional-catoid0 = functional-semigroup0 + catoid0

sublocale functional-catoid0  $\subseteq$  psg0: functional-catoid mcomp0  $\sigma_0 \tau_0$ ..

class functional-catoid1 = functional-semigroup1 + catoid1

sublocale functional-catoid1  $\subseteq$  psg1: functional-catoid mcomp1  $\sigma_1 \tau_1$ ..

class single-set-category0 = functional-catoid0 + local-catoid0

```

**sublocale** *single-set-category0*  $\subseteq$  *sscat0*: *single-set-category mcomp0*  $\sigma_0 \tau_0$ ..

**class** *single-set-category1* = *functional-catoid1* + *local-catoid1*

**sublocale** *single-set-category1*  $\subseteq$  *sscat1*: *single-set-category mcomp1*  $\sigma_1 \tau_1$ ..

## 2.2 2-Catoids

We define 2-catoids and 2-categories.

**class** *two-st-multimagma* = *st-multimagma0* + *st-multimagma1* +

**assumes** *comm-s0s1*:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$

**and** *comm-s0t1*:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$

**and** *comm-t0s1*:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$

**and** *comm-t0t1*:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$

**assumes** *interchange*:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$

**and** *s1-hom*:  $\text{Src}_1 (x \odot_0 y) \subseteq \sigma_1 x \odot_0 \sigma_1 y$

**and** *t1-hom*:  $\text{Tgt}_1 (x \odot_0 y) \subseteq \tau_1 x \odot_0 \tau_1 y$

**and** *s0-hom*:  $\text{Src}_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**and** *t0-hom*:  $\text{Tgt}_0 (x \odot_1 y) \subseteq \tau_0 x \odot_1 \tau_0 y$

**and** *s1s0* [*simp*]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$

**and** *s1t0* [*simp*]:  $\sigma_1 (\tau_0 x) = \tau_0 x$

**and** *t1s0* [*simp*]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$

**and** *t1t0* [*simp*]:  $\tau_1 (\tau_0 x) = \tau_0 x$

**class** *two-st-multimagma-strong* = *two-st-multimagma* +

**assumes** *s1-hom-strong*:  $\text{Src}_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$

**and** *t1-hom-strong*:  $\text{Tgt}_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

**context** *two-st-multimagma*

**begin**

**sublocale** *twolropp*: *two-st-multimagma*  $\lambda x y. y \odot_0 x \tau_0 \sigma_0 \lambda x y. y \odot_1 x \tau_1 \sigma_1$

**apply** *unfold-locales*

**apply** (*simp-all add: stmm0.stopp.Dst stmm1.stopp.Dst comm-t0t1 comm-t0s1 comm-s0t1 comm-s0s1 s1-hom t1-hom s0-hom t0-hom*)

**by** (*metis local.interchange local.stmm0.stopp.conv-exp local.stmm1.stopp.conv-exp multimagma.conv-exp*)

**lemma** *s0s1* [*simp*]:  $\sigma_0 (\sigma_1 x) = \sigma_0 x$

**by** (*simp add: local.comm-s0s1*)

**lemma** *s0t1* [*simp*]:  $\sigma_0 (\tau_1 x) = \sigma_0 x$

**by** (*simp add: local.comm-s0t1*)

**lemma** *t0s1* [*simp*]:  $\tau_0 (\sigma_1 x) = \tau_0 x$

**by** (*simp add: local.comm-t0s1*)

**lemma** *t1t1* [*simp*]:  $\tau_0 (\tau_1 x) = \tau_0 x$

**by** (*simp add: local.comm-t0t1*)

**lemma** *src0-comp1*:  $\Delta_1 x y \implies \text{Src}_0 (x \odot_1 y) = \{\sigma_0 x\}$   
**by** (*metis empty-is-image local.Dst1 local.comm-s0t1 local.s1s0 local.src1-absorb local.t1s0 s0s1 subset-singleton-iff twolropp.t0-hom*)

**lemma** *src0-comp1-var*:  $\Delta_1 x y \implies \text{Src}_0 (x \odot_1 y) = \{\sigma_0 y\}$   
**by** (*metis local.Dst1 s0s1 s0t1 src0-comp1*)

**lemma** *tgt0-comp1*:  $\Delta_1 x y \implies \text{Tgt}_0 (x \odot_1 y) = \{\tau_0 x\}$   
**by** (*metis empty-is-image local.Dst1 local.comm-t0t1 local.s1t0 local.src1-absorb local.t1t0 subset-singleton-iff t0s1 twolropp.s0-hom*)

**lemma** *tgt0-comp1-var*:  $\Delta_1 x y \implies \text{Tgt}_0 (x \odot_1 y) = \{\tau_0 y\}$   
**by** (*metis local.Dst1 t0s1 t1t1 tgt0-comp1*)

We lift the axioms to the powerset level.

**lemma** *comm-S0S1*:  $\text{Src}_0 (\text{Src}_1 X) = \text{Src}_1 (\text{Src}_0 X)$   
**by** (*simp add: image-image*)

**lemma** *comm-T0T1*:  $\text{Tgt}_0 (\text{Tgt}_1 X) = \text{Tgt}_1 (\text{Tgt}_0 X)$   
**by** (*metis (mono-tags, lifting) image-cong image-image local.comm-t0t1*)

**lemma** *comm-S0T1*:  $\text{Src}_0 (\text{Tgt}_1 x) = \text{Tgt}_1 (\text{Src}_0 x)$   
**by** (*simp add: image-image*)

**lemma** *comm-T0S1*:  $\text{Tgt}_0 (\text{Src}_1 x) = \text{Src}_1 (\text{Tgt}_0 x)$   
**by** (*metis (mono-tags, lifting) image-cong image-image local.comm-t0s1*)

**lemma** *interchange-lifting*:  $(W *_1 X) *_0 (Y *_1 Z) \subseteq (W *_0 Y) *_1 (X *_0 Z)$

**proof**–

{**fix** *a*  
**assume**  $a \in (W *_1 X) *_0 (Y *_1 Z)$   
**hence**  $\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. a \in (w \odot_1 x) *_0 (y \odot_1 z)$   
**using** *local.mm0.conv-exp2 local.mm1.conv-exp2* **by** *fastforce*  
**hence**  $\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. a \in (w \odot_0 y) *_1 (x \odot_0 z)$   
**using** *local.interchange* **by** *blast*  
**hence**  $a \in (W *_0 Y) *_1 (X *_0 Z)$   
**using** *local.mm0.conv-exp2 local.mm1.conv-exp2* **by** *auto*}  
**thus** *?thesis..*

**qed**

**lemma** *Src1-hom*:  $\text{Src}_1 (X *_0 Y) \subseteq \text{Src}_1 X *_0 \text{Src}_1 Y$

**proof**–

{**fix** *a*  
**have**  $(a \in \text{Src}_1 (X *_0 Y)) = (\exists b \in X *_0 Y. a = \sigma_1 b)$   
**by** *blast*  
**also have**  $\dots = (\exists b. \exists c \in X. \exists d \in Y. a = \sigma_1 b \wedge b \in c \odot_0 d)$   
**by** (*metis multimagma.conv-exp2*)  
**also have**  $\dots = (\exists c \in X. \exists d \in Y. a \in \text{Src}_1 (c \odot_0 d))$

by *blast*  
 also have  $\dots \longrightarrow (\exists c \in X. \exists d \in Y. a \in \sigma_1 c \odot_0 \sigma_1 d)$   
 using *local.s1-hom* by *fastforce*  
 also have  $\dots = (\exists c \in \text{Src}_1 X. \exists d \in \text{Src}_1 Y. a \in c \odot_0 d)$   
 by *blast*  
 also have  $\dots = (a \in \text{Src}_1 X *_0 \text{Src}_1 Y)$   
 using *local.mm0.conv-exp2* by *auto*  
 finally have  $(a \in \text{Src}_1 (X *_0 Y)) \longrightarrow (a \in \text{Src}_1 X *_0 \text{Src}_1 Y).$   
 thus *?thesis*  
 by *force*  
 qed

lemma *Tgt1-hom*:  $Tgt_1 (X *_0 Y) \subseteq Tgt_1 X *_0 Tgt_1 Y$

proof –  
 {fix *a*  
 have  $(a \in Tgt_1 (X *_0 Y)) = (\exists c \in X. \exists d \in Y. a \in Tgt_1 (c \odot_0 d))$   
 by (*smt (verit, best) image-iff multimagma.conv-exp2*)  
 also have  $\dots \longrightarrow (\exists c \in X. \exists d \in Y. a \in \tau_1 c \odot_0 \tau_1 d)$   
 using *local.t1-hom* by *fastforce*  
 also have  $\dots = (a \in Tgt_1 X *_0 Tgt_1 Y)$   
 using *local.mm0.conv-exp2* by *auto*  
 finally have  $(a \in Tgt_1 (X *_0 Y)) \longrightarrow (a \in Tgt_1 X *_0 Tgt_1 Y).$   
 thus *?thesis*  
 by *force*  
 qed

lemma *Src0-hom*:  $\text{Src}_0 (X *_1 Y) \subseteq \text{Src}_0 X *_1 \text{Src}_0 Y$

proof –  
 {fix *a*  
 assume  $a \in \text{Src}_0 (X *_1 Y)$   
 hence  $\exists c \in X. \exists d \in Y. a \in \text{Src}_0 (c \odot_1 d)$   
 using *local.mm1.conv-exp2* by *fastforce*  
 hence  $\exists c \in X. \exists d \in Y. a \in \sigma_0 c \odot_1 \sigma_0 d$   
 using *local.s0-hom* by *blast*  
 hence  $a \in \text{Src}_0 X *_1 \text{Src}_0 Y$   
 using *local.mm1.conv-exp2* by *auto*}  
 thus *?thesis*  
 by *force*  
 qed

lemma *Tgt0-hom*:  $Tgt_0 (X *_1 Y) \subseteq Tgt_0 X *_1 Tgt_0 Y$

proof –  
 {fix *a*  
 assume  $a \in Tgt_0 (X *_1 Y)$   
 hence  $\exists c \in X. \exists d \in Y. a \in Tgt_0 (c \odot_1 d)$   
 using *local.mm1.conv-exp2* by *fastforce*  
 hence  $\exists c \in X. \exists d \in Y. a \in \tau_0 c \odot_1 \tau_0 d$   
 using *local.t0-hom* by *blast*  
 hence  $a \in Tgt_0 X *_1 Tgt_0 Y$



```

    using local.mm1.conv-exp2 by auto}
  thus ?thesis
    by force
qed

lemma S1S0 [simp]: Src1 (Src0 X) = Src0 X
  by force

lemma S1T0 [simp]: Src1 (Tgt0 X) = Tgt0 X
  by force

lemma T1S0 [simp]: Tgt1 (Src0 X) = Src0 X
  by force

lemma T1T0 [simp]: Tgt1 (Tgt0 X) = Tgt0 X
  by force

lemma (in two-st-multimagma)
  s1fix *0 s1fix ⊆ s1fix

oops

lemma id1-comp0-eq: s1fix ⊆ s1fix *0 s1fix
  by (metis S1S0 local.stmm0.stopp.conv-isor local.stmm0.stopp.conv-uns local.stmm0.stopp.stfix-set
    local.stmm0.stopp.tfix-im local.stmm1.stopp.Tgt-subid)

lemma (in two-st-multimagma) id01:
  s0fix ⊆ s1fix
proof -
  {fix a
    have (a ∈ s0fix) = (∃ b. a = σ0 b)
      by (metis imageE local.stmm0.stopp.tfix-im rangeI)
    hence (a ∈ s0fix) = (∃ b. a = σ1 (σ0 b))
      by fastforce
    hence (a ∈ s0fix) ⇒ (∃ b. a = σ1 b)
      by blast
    hence (a ∈ s0fix) ⇒ (a ∈ s1fix)
      using local.stmm1.stopp.tfix-im by blast}
  thus ?thesis
    by blast
qed

end

context two-st-multimagma-strong
begin

lemma Src1-hom-strong: Src1 (X *0 Y) = Src1 X *0 Src1 Y
proof -

```

```

{fix a
have (a ∈ Src1 (X *0 Y)) = (∃ b ∈ X *0 Y. a = σ1 b)
  by blast
also have ... = (∃ b. ∃ c ∈ X. ∃ d ∈ Y. a = σ1 b ∧ b ∈ c ⊙0 d)
  by (metis multimagma.conv-exp2)
also have ... = (∃ c ∈ X. ∃ d ∈ Y. a ∈ Src1 (c ⊙0 d))
  by blast
also have ... = (∃ c ∈ X. ∃ d ∈ Y. a ∈ σ1 c ⊙0 σ1 d)
  using local.s1-hom-strong by fastforce
also have ... = (∃ c ∈ Src1 X. ∃ d ∈ Src1 Y. a ∈ c ⊙0 d)
  by blast
also have ... = (a ∈ Src1 X *0 Src1 Y)
  using local.mm0.conv-exp2 by auto
finally have (a ∈ Src1 (X *0 Y)) = (a ∈ Src1 X *0 Src1 Y).}
thus ?thesis
  by force
qed

```

**lemma** *Tgt1-hom-strong*:  $Tgt_1 (X *_{0} Y) = Tgt_1 X *_{0} Tgt_1 Y$

**proof**–

```

{fix a
have (a ∈ Tgt1 (X *0 Y)) = (∃ c ∈ X. ∃ d ∈ Y. a ∈ Tgt1 (c ⊙0 d))
  by (smt (verit, best) image-iff multimagma.conv-exp2)
also have ... = (∃ c ∈ X. ∃ d ∈ Y. a ∈ τ1 c ⊙0 τ1 d)
  using local.t1-hom-strong by fastforce
also have ... = (a ∈ Tgt1 X *0 Tgt1 Y)
  using local.mm0.conv-exp2 by auto
finally have (a ∈ Tgt1 (X *0 Y)) = (a ∈ Tgt1 X *0 Tgt1 Y).}
thus ?thesis
  by force
qed

```

**lemma** *id1-comp0*:  $s1fix *_{0} s1fix \subseteq s1fix$

**proof**–

```

{fix a
have (a ∈ s1fix *0 s1fix) = (∃ b ∈ s1fix. ∃ c ∈ s1fix. a ∈ b ⊙0 c)
  by (meson local.mm0.conv-exp2)
also have ... = (∃ b c. a ∈ σ1 b ⊙0 σ1 c)
  by (metis image-iff local.stmm1.stopp.tfix-im rangeI)
finally have (a ∈ s1fix *0 s1fix) = (∃ b c. a ∈ Src1 (b ⊙0 c))
  using local.s1-hom-strong by presburger
hence (a ∈ s1fix *0 s1fix) ⇒ (∃ b. a = σ1 b)
  by blast
hence (a ∈ s1fix *0 s1fix) ⇒ (a ∈ s1fix)
  using local.stmm1.stopp.Tgt-subid by blast}
thus ?thesis
  by blast
qed

```

**lemma** *id1-comp0-eq* [*simp*]:  $s1fx *_0 s1fx = s1fx$   
**using** *local.id1-comp0 local.id1-comp0-eq* **by force**

**end**

### 2.3 2-Catoids and single-set 2-categories

**class** *two-catoid* = *two-st-multimagma* + *catoid0* + *catoid1*

**lemma** (**in** *two-catoid*)  $\Delta_0 x y \implies Src_1 (x \odot_0 y) = \{\sigma_1 x\}$

**oops**

**lemma** (**in** *two-catoid*)  $\Delta_0 x y \implies Tgt_1 (x \odot_0 y) = \{\tau_1 x\}$

**oops**

**class** *two-catoid-strong* = *two-st-multimagma-strong* + *catoid0* + *catoid1*

**class** *local-two-catoid* = *two-st-multimagma* + *local-catoid0* + *local-catoid1*

**begin**

local 2-catoids need not be strong

**lemma**  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$

**oops**

**lemma**  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

**oops**

**lemma**  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y \vee Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

**oops**

**end**

**class** *functional-two-catoid* = *two-st-multimagma* + *functional-catoid0* + *functional-catoid1*

**begin**

**lemma**  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$

**oops**

**lemma**  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

```

oops

lemma  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y \vee Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$ 

oops

end

class local-two-catoid-strong = two-st-multimagma-strong + local-catoid0 + local-catoid1

class two-category = two-st-multimagma + single-set-category0 + single-set-category1

begin

lemma s1-hom-strong [simp]:  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$ 
proof cases
  assume  $\sigma_1 x \odot_0 \sigma_1 y = \{\}$ 
  thus ?thesis
  using local.twolropp.t1-hom by blast
next
  assume  $h: \sigma_1 x \odot_0 \sigma_1 y \neq \{\}$ 
  hence  $(\tau_0 (\sigma_1 x) = \sigma_0 (\sigma_1 y))$ 
  using local.Dst0 by blast
  hence  $\tau_0 x = \sigma_0 y$ 
  by auto
  hence  $x \odot_0 y \neq \{\}$ 
  by (simp add: ssmsg0.st-local)
  thus ?thesis
  by (metis h image-is-empty local.pm0.fun-in-sgl local.pm0.functionality-lem local.twolropp.t1-hom subset-singletonD)
qed

lemma s1-hom-strong-delta:  $\Delta_0 x y = \Delta_0 (\sigma_1 x) (\sigma_1 y)$ 
  by (simp add: ssmsg0.st-local)

lemma t1-hom-strong [simp]:  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$ 
  by (metis (no-types, lifting) empty-is-image local.pm0.functionality-lem-var local.s0t1 local.t1t1 local.twolropp.s1-hom ssmsg0.st-local subset-singleton-iff)

lemma t1-hom-strong-delta:  $\Delta_0 x y = \Delta_0 (\tau_1 x) (\tau_1 y)$ 
  by (simp add: ssmsg0.st-local)

lemma conv0-sgl:  $a \in x \odot_0 y \implies \{a\} = x \odot_0 y$ 
  using local.functionality0 by fastforce

lemma conv1-sgl:  $a \in \{x\} *_1 \{y\} \implies \{a\} = \{x\} *_1 \{y\}$ 
  using local.functionality1 local.mm1.conv-exp by force

```

Next we derive some simple globular properties.

**lemma** *strong-interchange-St1*:

**assumes**  $a \in (w \odot_0 x) *_1 (y \odot_0 z)$

**shows**  $Tgt_1 (w \odot_0 x) = Src_1 (y \odot_0 z)$

**by** (*smt (verit, ccfv-threshold) assms empty-iff image-insert image-is-empty insertE local.Dst1 local.mm1.conv-exp2 local.pm0.functionality-lem-var*)

**lemma** *strong-interchange-ll0*:

**assumes**  $a \in (w \odot_0 x) *_1 (y \odot_0 z)$

**shows**  $\sigma_0 w = \sigma_0 y$

**by** (*metis assms empty-iff local.Dst1 local.s0s1 local.s0t1 local.stmm1.stopp.conv-exp2 stmsg0.src-comp-aux*)

There is no strong interchange law, and the homomorphism laws for zero sources and targets stay weak, too.

**lemma**  $(w \odot_1 y) *_0 (x \odot_1 z) = (w \odot_0 x) *_1 (y \odot_0 z)$

**oops**

**lemma**  $R_0 (x \odot_1 y) = r_0 x \odot_1 r_0 y$

**oops**

**lemma**  $L_0 (x \odot_1 y) = l_0 x \odot_1 l_0 y$

**oops**

**lemma**  $(W *_0 Y) *_1 (X *_0 Z) = (W *_1 X) *_0 (Y *_1 Z)$

**oops**

**lemma**  $\Delta_0 x y \implies Src_1 (x \odot_0 y) = \{\sigma_1 x\}$

**oops**

**lemma**  $\Delta_0 x y \implies Tgt_1 (x \odot_0 y) = \{\tau_1 x\}$

**oops**

**end**

## 2.4 Reduced axiomatisations

**class** *two-st-multimagma-red* = *st-multimagma0* + *st-multimagma1* +

**assumes** *interchange*:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$

**assumes** *src1-hom*:  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$

**and** *tgt1-hom*:  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

**and** *src0-weak-hom*:  $Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**and** *tgt0-weak-hom*:  $Tgt_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**begin**

**lemma** *s0t1s0* [*simp*]:  $\sigma_0 (\tau_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

**have**  $\{\tau_1 (\sigma_0 x)\} = Tgt_1 (\sigma_0 x \odot_0 \sigma_0 x)$   
  **by** *simp*  
**also have**  $\dots = \tau_1 (\sigma_0 x) \odot_0 \tau_1 (\sigma_0 x)$   
  **by** (*meson local.tgt1-hom*)  
**also have**  $\dots = \tau_1 (\sigma_0 x) \odot_0 \tau_1 (\tau_1 (\sigma_0 x))$   
  **by** *simp*  
**also have**  $\dots = Tgt_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
  **by** (*simp add: local.tgt1-hom*)  
**finally have**  $Tgt_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x)) \neq \{\}$   
  **by** *force*  
**hence**  $\sigma_0 x \odot_0 \tau_1 (\sigma_0 x) \neq \{\}$   
  **by** *blast*  
**thus** *?thesis*  
  **using** *stmm0.s-absorb-var3* **by** *auto*

**qed**

**lemma** *t0s1s0* [*simp*]:  $\tau_0 (\sigma_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

**have**  $\{\sigma_1 (\sigma_0 x)\} = Src_1 (\sigma_0 x \odot_0 \sigma_0 x)$   
  **by** *simp*  
**also have**  $\dots = \sigma_1 (\sigma_0 x) \odot_0 \sigma_1 (\sigma_0 x)$   
  **by** (*meson local.src1-hom*)  
**also have**  $\dots = \sigma_1 (\sigma_1 (\sigma_0 x)) \odot_0 \sigma_1 (\sigma_0 x)$   
  **by** *simp*  
**also have**  $\dots = Src_1 (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x)$   
  **using** *local.src1-hom* **by** *force*  
**finally have**  $Src_1 (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) \neq \{\}$   
  **by** *force*  
**hence**  $\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x \neq \{\}$   
  **by** *blast*  
**thus** *?thesis*  
  **by** (*simp add: local.Dst0*)

**qed**

**lemma** *s1s0* [*simp*]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$

**proof**–

**have**  $\{\sigma_0 x\} = \sigma_0 x \odot_0 \sigma_0 x$   
  **by** *simp*  
**also have**  $\dots = (\sigma_1 (\sigma_0 x) \odot_1 \sigma_0 x) *_0 (\sigma_0 x \odot_1 \tau_1 (\sigma_0 x))$   
  **by** (*simp add: multimagma.conv-atom*)  
**also have**  $\dots \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
  **using** *local.interchange* **by** *blast*  
**also have**  $\dots = (\sigma_1 (\sigma_0 x) \odot_0 \tau_0 (\sigma_1 (\sigma_0 x))) *_1 (\sigma_0 (\tau_1 (\sigma_0 x)) \odot_0 \tau_1 (\sigma_0 x))$   
  **by** *simp*

**also have**  $\dots = \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$   
**using** *local.mm1.conv-atom local.src0-absorb local.tgt0-absorb* **by** *presburger*  
**finally have**  $\{\sigma_0 x\} \subseteq \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$ .  
**thus** *?thesis*  
**by** (*metis empty-iff insert-subset singletonD stmm1.st-comm stmm1.st-prop stmm1.t-idem*)  
**qed**

**lemma** *s1t0* [*simp*]:  $\sigma_1 (\tau_0 x) = \tau_0 x$   
**by** (*metis local.s1s0 local.stmm0.stopp.ts-compat*)

**lemma** *t1s0* [*simp*]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$   
**by** (*simp add: stmm1.st-fix*)

**lemma** *t1t0* [*simp*]:  $\tau_1 (\tau_0 x) = \tau_0 x$   
**by** (*simp add: stmm1.st-fix*)

**lemma** *comm-s0s1*:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$   
**proof** –

**have**  $\{\sigma_1 x\} = \sigma_1 (\sigma_0 x) \odot_0 \sigma_1 x$   
**by** (*metis image-empty image-insert local.src0-absorb local.src1-hom*)  
**also have**  $\dots = \sigma_0 x \odot_0 \sigma_1 x$   
**by** *simp*  
**finally have**  $\sigma_0 x \odot_0 \sigma_1 x \neq \{\}$   
**by** *force*  
**hence**  $\tau_0 (\sigma_0 x) = \sigma_0 (\sigma_1 x)$   
**by** (*meson local.Dst0*)  
**hence**  $\sigma_0 x = \sigma_0 (\sigma_1 x)$   
**by** *simp*  
**thus** *?thesis*  
**by** *simp*  
**qed**

**lemma** *comm-s0t1*:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$

**proof** –  
**have**  $\{\tau_1 x\} = \tau_1 (\sigma_0 x) \odot_0 \tau_1 x$   
**by** (*metis local.src0-absorb local.t1s0 local.tgt1-hom stmm0.s-absorb-var*)  
**hence**  $\tau_1 (\sigma_0 x) \odot_0 \tau_1 x \neq \{\}$   
**by** *force*  
**hence**  $\tau_0 (\tau_1 (\sigma_0 x)) = \sigma_0 (\tau_1 x)$   
**using** *local.Dst0* **by** *blast*  
**thus** *?thesis*  
**by** *simp*  
**qed**

**lemma** *comm-t0s1*:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$

**proof** –  
**have**  $\{\sigma_1 x\} = \sigma_1 x \odot_0 \sigma_1 (\tau_0 x)$   
**by** (*metis local.s1t0 local.src1-hom local.stmm0.stopp.s-absorb-var local.tgt0-absorb*)

**hence**  $\sigma_1 x \odot_0 \sigma_1 (\tau_0 x) \neq \{\}$   
**by** *force*  
**hence**  $\tau_0 (\sigma_1 x) = \tau_0 (\sigma_1 (\tau_0 x))$   
**by** (*metis local.s1t0 local.stmm0.stopp.s-absorb-var stmm0.tt-idem*)  
**thus** *?thesis*  
**by** *simp*  
**qed**

**lemma** *comm-t0t1*:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$   
**by** (*metis local.s1t0 local.stmm0.stopp.s-absorb-var3 local.tgt1-hom stmm1.st-fix*)

**lemma**  $\sigma_0 x = \sigma_1 x$

**oops**

**lemma**  $\sigma_0 x = \tau_1 x$

**oops**

**lemma**  $\tau_0 x = \tau_1 x$

**oops**

**lemma**  $\sigma_0 x = \tau_0 x$

**oops**

**lemma**  $\sigma_1 x = \tau_1 x$

**oops**

**lemma**  $x \odot_0 y = x \odot_1 y$

**oops**

**lemma**  $x \odot_0 y = y \odot_0 x$

**oops**

**lemma**  $x \odot_1 y = y \odot_1 x$

**oops**

**end**

**class** *two-catoid-red* = *catoid0* + *catoid1* +  
**assumes** *interchange*:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$   
**and** *s1-hom*:  $\text{Src}_1 (x \odot_0 y) \subseteq \sigma_1 x \odot_0 \sigma_1 y$   
**and** *t1-hom*:  $\text{Tgt}_1 (x \odot_0 y) \subseteq \tau_1 x \odot_0 \tau_1 y$



**begin**

**lemma** *s0t1s0* [*simp*]:  $\sigma_0 (\tau_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

**have**  $\{\sigma_0 x\} = (\sigma_1 (\sigma_0 x) \odot_1 \sigma_0 x) *_0 (\sigma_0 x \odot_1 \tau_1 (\sigma_0 x))$   
**by** *simp*

**also have**  $\dots \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
**using** *local.interchange* **by** *blast*

**finally have**  $\{\sigma_0 x\} \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$ .

**hence**  $(\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x)) \neq \{\}$   
**by** *fastforce*

**hence**  $\sigma_0 x \odot_0 \tau_1 (\sigma_0 x) \neq \{\}$   
**using** *local.mm1.conv-exp2* **by** *force*

**thus** *?thesis*  
**by** (*simp add: stmm0.s-absorb-var3*)

**qed**

**lemma** *t0s1s0* [*simp*]:  $\tau_0 (\sigma_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

**have**  $\{\sigma_0 x\} = (\sigma_1 (\sigma_0 x) \odot_1 \sigma_0 x) *_0 (\sigma_0 x \odot_1 \tau_1 (\sigma_0 x))$   
**by** *simp*

**also have**  $\dots \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
**using** *local.interchange* **by** *blast*

**finally have**  $\{\sigma_0 x\} \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$ .

**hence**  $(\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x)) \neq \{\}$   
**by** *fastforce*

**hence**  $\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x \neq \{\}$   
**using** *local.mm1.conv-exp2* **by** *force*

**thus** *?thesis*  
**by** (*simp add: local.Dst0*)

**qed**

**lemma** *s1s0* [*simp*]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$

**proof**–

**have**  $\{\sigma_0 x\} = \sigma_0 x \odot_0 \sigma_0 x$   
**by** *simp*

**also have**  $\dots = (\sigma_1 (\sigma_0 x) \odot_1 \sigma_0 x) *_0 (\sigma_0 x \odot_1 \tau_1 (\sigma_0 x))$   
**by** (*simp add: multimagma.conv-atom*)

**also have**  $\dots \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
**using** *local.interchange* **by** *blast*

**also have**  $\dots = (\sigma_1 (\sigma_0 x) \odot_0 \tau_0 (\sigma_1 (\sigma_0 x))) *_1 (\sigma_0 (\tau_1 (\sigma_0 x)) \odot_0 \tau_1 (\sigma_0 x))$   
**by** (*metis calculation empty-iff insert-subset local.t0s1s0 multimagma.conv-exp2*

*stmm0.s-absorb-var*)

**also have**  $\dots = \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$

**using** *local.mm1.conv-atom local.src0-absorb local.tgt0-absorb* **by** *presburger*

**finally have**  $\{\sigma_0 x\} \subseteq \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$ .

**thus** *?thesis*  
**using** *local.stmm1.stopp.Dst* **by** *fastforce*

qed

**lemma** *s1t0* [*simp*]:  $\sigma_1 (\tau_0 x) = \tau_0 x$   
by (*metis local.s1s0 local.stmm0.stopp.ts-compat*)

**lemma** *t1s0* [*simp*]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$   
by (*simp add: stmm1.st-fix*)

**lemma** *t1t0* [*simp*]:  $\tau_1 (\tau_0 x) = \tau_0 x$   
by (*simp add: stmm1.st-fix*)

**lemma** *comm-s0s1*:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$   
by (*metis image-empty image-insert local.s1-hom local.s1s0 local.src0-absorb order-class.order-eq-iff stmm0.s-absorb-var3*)

**lemma** *comm-s0t1*:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$   
by (*metis local.src0-absorb local.src1-absorb local.stmsg1.ts-msg.src-comp-cond local.t1-hom local.t1s0 order-antisym-conv stmm0.s-absorb-var3 subset-insertI*)

**lemma** *comm-t0s1*:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$   
by (*metis equalityI image-empty image-insert local.s1-hom local.s1t0 local.stmm0.stopp.s-absorb local.stmm0.stopp.s-absorb-var2*)

**lemma** *comm-t0t1*:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$   
by (*metis empty-is-image local.src1-absorb local.stmm0.stopp.s-absorb-var2 local.stmsg1.ts-msg.src-comp-cond local.t1-hom local.t1t0 local.tgt0-absorb subset-antisym*)

**lemma** *s0-hom*:  $Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**proof** *cases*

assume  $Src_0 (x \odot_1 y) = \{\}$

thus *?thesis*

by *auto*

**next**

assume  $h: Src_0 (x \odot_1 y) \neq \{\}$

hence  $h1: \tau_1 x = \sigma_1 y$

by (*simp add: local.Dst1*)

hence  $Src_0 (x \odot_1 y) = Src_0 (Src_1 (x \odot_1 y))$

unfolding *image-def* using *local.comm-s0s1* by *auto*

also have  $\dots = Src_0 (Src_1 (x \odot_1 \sigma_1 y))$

using *h stmsg1.src-local-cond* by *auto*

also have  $\dots = Src_0 (Src_1 (x \odot_1 \tau_1 x))$

using *h1* by *presburger*

also have  $\dots = \{\sigma_0 x\}$

by (*simp add: local.comm-s0s1*)

also have  $\dots = \sigma_0 x \odot_1 \tau_1 (\sigma_0 x)$

using *local.tgt1-absorb* by *presburger*

also have  $\dots = \sigma_0 x \odot_1 \sigma_0 (\tau_1 x)$

by (*simp add: local.comm-s0t1*)

also have  $\dots = \sigma_0 x \odot_1 \sigma_0 (\sigma_1 y)$

by (*simp add: h1*)  
 also have  $\dots = \sigma_0 x \odot_1 \sigma_0 y$   
 by (*simp add: local.comm-s0s1*)  
 finally show *?thesis*  
 by *blast*  
 qed

**lemma** *t0-hom*:  $Tgt_0 (x \odot_1 y) \subseteq \tau_0 x \odot_1 \tau_0 y$   
 by (*metis equals0D image-subsetI local.Dst1 local.comm-t0s1 local.comm-t0t1 local.stmsg1.ts-msg.src-comp-aux local.t1t0 local.tgt1-absorb singletonI*)

end

**class** *two-catoid-red-strong* = *catoid0* + *catoid1* +  
 assumes *interchange*:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$   
 and *s1-hom-strong*:  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$   
 and *t1-hom-strong*:  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$

begin

**lemma** *s0t1s0* [*simp*]:  $\sigma_0 (\tau_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

have  $\{\tau_1 (\sigma_0 x)\} = Tgt_1 (\sigma_0 x \odot_0 \sigma_0 x)$   
 by *simp*  
 also have  $\dots = \tau_1 (\sigma_0 x) \odot_0 \tau_1 (\sigma_0 x)$   
 using *local.t1-hom-strong* by *blast*  
 also have  $\dots = \tau_1 (\sigma_0 x) \odot_0 \tau_1 (\tau_1 (\sigma_0 x))$   
 by *simp*  
 also have  $\dots = Tgt_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$   
 by (*simp add: local.t1-hom-strong*)  
 finally have  $Tgt_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x)) \neq \{\}$   
 by *force*  
 hence  $\sigma_0 x \odot_0 \tau_1 (\sigma_0 x) \neq \{\}$   
 by *blast*  
 thus *?thesis*  
 using *stmm0.s-absorb-var3* by *blast*

qed

**lemma** *t0s1s0* [*simp*]:  $\tau_0 (\sigma_1 (\sigma_0 x)) = \sigma_0 x$

**proof**–

have  $\{\sigma_1 (\sigma_0 x)\} = Src_1 (\sigma_0 x \odot_0 \sigma_0 x)$   
 by *simp*  
 also have  $\dots = \sigma_1 (\sigma_0 x) \odot_0 \sigma_1 (\sigma_0 x)$   
 using *local.s1-hom-strong* by *blast*  
 also have  $\dots = \sigma_1 (\sigma_1 (\sigma_0 x)) \odot_0 \sigma_1 (\sigma_0 x)$   
 by *simp*  
 also have  $\dots = Src_1 (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x)$   
 using *local.s1-hom-strong* by *auto*  
 finally have  $Src_1 (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) \neq \{\}$

by *force*  
 hence  $\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x \neq \{\}$   
 by *blast*  
 thus *?thesis*  
 by (*simp add: local.Dst0*)  
**qed**

**lemma** *s1s0* [*simp*]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$

**proof**–

have  $\{\sigma_0 x\} = \sigma_0 x \odot_0 \sigma_0 x$

by *simp*

also have  $\dots = (\sigma_1 (\sigma_0 x) \odot_1 \sigma_0 x) *_0 (\sigma_0 x \odot_1 \tau_1 (\sigma_0 x))$

by (*simp add: multimagma.conv-atom*)

also have  $\dots \subseteq (\sigma_1 (\sigma_0 x) \odot_0 \sigma_0 x) *_1 (\sigma_0 x \odot_0 \tau_1 (\sigma_0 x))$

using *local.interchange* by *blast*

also have  $\dots = (\sigma_1 (\sigma_0 x) \odot_0 \tau_0 (\sigma_1 (\sigma_0 x))) *_1 (\sigma_0 (\tau_1 (\sigma_0 x)) \odot_0 \tau_1 (\sigma_0 x))$

by *simp*

also have  $\dots = \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$

using *local.mm1.conv-atom* *local.src0-absorb* *local.tgt0-absorb* by *presburger*

finally have  $\{\sigma_0 x\} \subseteq \sigma_1 (\sigma_0 x) \odot_1 \tau_1 (\sigma_0 x)$ .

thus *?thesis*

using *local.stmm1.stopp.Dst* by *fastforce*

**qed**

**lemma** *s1t0* [*simp*]:  $\sigma_1 (\tau_0 x) = \tau_0 x$

by (*metis local.s1s0 local.stmm0.stopp.ts-compat*)

**lemma** *t1s0* [*simp*]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$

by (*simp add: stmm1.st-fix*)

**lemma** *t1t0* [*simp*]:  $\tau_1 (\tau_0 x) = \tau_0 x$

by (*simp add: stmm1.st-fix*)

**lemma** *comm-s0s1*:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$

by (*metis local.s1-hom-strong local.s1s0 stmm0.s-absorb-var*)

**lemma** *comm-s0t1*:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$

by (*metis local.t1-hom-strong local.t1s0 stmm0.s-absorb-var*)

**lemma** *comm-t0s1*:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$

by (*metis empty-not-insert local.Dst0 local.s1-hom-strong local.s1t0 local.tgt0-absorb*)

**lemma** *comm-t0t1*:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$

using *local.t1-hom-strong* *local.stmm0.stopp.s-absorb-var2* by *fastforce*

**lemma** *s0-weak-hom*:  $Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**proof** *cases*

assume  $Src_0 (x \odot_1 y) = \{\}$

thus *?thesis*

```

    by auto
next
  assume h: Src0 (x ⊙1 y) ≠ {}
  hence h1: τ1 x = σ1 y
    by (simp add: local.Dst1)
  hence Src0 (x ⊙1 y) = Src0 (Src1 (x ⊙1 y))
    unfolding image-def using local.comm-s0s1 by auto
  also have ... = Src0 (Src1 (x ⊙1 σ1 y))
    using h stmsg1.src-local-cond by auto
  also have ... = Src0 (Src1 (x ⊙1 τ1 x))
    using h1 by presburger
  also have ... = {σ0 x}
    by (simp add: local.comm-s0s1)
  also have ... = σ0 x ⊙1 τ1 (σ0 x)
    using local.tgt1-absorb by presburger
  also have ... = σ0 x ⊙1 σ0 (τ1 x)
    by (simp add: local.comm-s0t1)
  also have ... = σ0 x ⊙1 σ0 (σ1 y)
    by (simp add: h1)
  also have ... = σ0 x ⊙1 σ0 y
    by (simp add: local.comm-s0s1)
  finally show ?thesis
    by blast
qed

```

**lemma** *t0-weak-hom*:  $Tgt_0 (x \odot_1 y) \subseteq \tau_0 x \odot_1 \tau_0 y$   
 by (metis equals0D image-subsetI local.Dst1 local.comm-t0s1 local.comm-t0t1 local.stmsg1.ts-msg.src-comp-aux local.t1t0 local.tgt1-absorb singletonI)

end

```

class two-catoid-red2 = single-set-category0 + single-set-category1 +
  assumes comm-s0s1: σ0 (σ1 x) = σ1 (σ0 x)
  and comm-s0t1: σ0 (τ1 x) = τ1 (σ0 x)
  and comm-t0s1: τ0 (σ1 x) = σ1 (τ0 x)
  and comm-t0t1: τ0 (τ1 x) = τ1 (τ0 x)
  and s1s0 [simp]: σ1 (σ0 x) = σ0 x
  and s1t0 [simp]: σ1 (τ0 x) = τ0 x
  and t1s0 [simp]: τ1 (σ0 x) = σ0 x
  and t1t0 [simp]: τ1 (τ0 x) = τ0 x

```

begin

**lemma**  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$

oops

**lemma**  $Src_1 (x \odot_0 y) \subseteq \sigma_1 x \odot_0 \sigma_1 y$

**oops**

**lemma**  $Tgt_1 (x \odot_0 y) \subseteq \tau_1 x \odot_0 \tau_1 y$

**oops**

**lemma**  $s0\text{-hom}: Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$

**by** (*smt* (*verit*, *ccfv-SIG*) *image-subsetI insertCI local.Dst1 local.comm-s0s1 local.cal.comm-s0t1 local.src0-absorb local.t1s0 local.tgt1-absorb stmm0.s-absorb-var3 stmsg1.src-twisted-aux*)

**lemma**  $t0\text{-hom}: Tgt_0 (x \odot_1 y) \subseteq \tau_0 x \odot_1 \tau_0 y$

**by** (*metis equals0D image-subsetI insertI1 local.comm-t0s1 local.comm-t0t1 local.stmm1.stopp.Dst local.t1t0 local.tgt1-absorb stmsg1.tgt-comp-aux*)

**end**

**class** *two-catoid-red3* = *catoid0* + *catoid1* +  
**assumes** *interchange*:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$   
**and** *s1-hom*:  $Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$   
**and** *t1-hom*:  $Tgt_0 (x \odot_1 y) \subseteq \tau_0 x \odot_1 \tau_0 y$

**lemma** (**in** *two-catoid-red3*)

$Src_1 (x \odot_0 y) \subseteq \sigma_1 x \odot_0 \sigma_1 y$

**oops**

**lemma** (**in** *two-catoid-red3*)

$Tgt_1 (x \odot_0 y) \subseteq \tau_1 x \odot_0 \tau_1 y$

**oops**

**end**

## 3 2-Kleene algebras

**theory** *Two-Kleene-Algebra*

**imports** *Quantales-Converse.Modal-Kleene-Algebra-Var*

**begin**

Here we develop (globular) 2-semigroups and (globular) 2-Kleene algebras. These should eventually be extended to n-structures and omega-structures.

### 3.1 Copies for 0-structures

**class** *comp0-op* =

**fixes** *comp0* ::  $'a \Rightarrow 'a \Rightarrow 'a$  (**infixl**  $\cdot_0$  70)

```

class id0-op =
  fixes id0 :: 'a (1_0)

class star0-op =
  fixes star0 :: 'a ⇒ 'a

class dom0-op =
  fixes dom0 :: 'a ⇒ 'a

class cod0-op =
  fixes cod0 :: 'a ⇒ 'a

class monoid-mult0 = comp0-op + id0-op +
  assumes par-assoc0:  $x \cdot_0 (y \cdot_0 z) = (x \cdot_0 y) \cdot_0 z$ 
  and comp0-unl:  $1_0 \cdot_0 x = x$ 
  and comp0-unr:  $x \cdot_0 1_0 = x$ 

class dioid0 = monoid-mult0 + join-semilattice-zero +
  assumes distl0:  $x \cdot_0 (y + z) = x \cdot_0 y + x \cdot_0 z$ 
  and distr0:  $(x + y) \cdot_0 z = x \cdot_0 z + y \cdot_0 z$ 
  and annil0:  $0 \cdot_0 x = 0$ 
  and annir0:  $x \cdot_0 0 = 0$ 

class kleene-algebra0 = dioid0 + star0-op +
  assumes star0-unfoldl:  $1_0 + x \cdot_0 \text{star0 } x \leq \text{star0 } x$ 
  and star0-unfoldr:  $1_0 + \text{star0 } x \cdot_0 x \leq \text{star0 } x$ 
  and star0-inductl:  $z + x \cdot_0 y \leq y \implies \text{star0 } x \cdot_0 z \leq y$ 
  and star0-inductr:  $z + y \cdot_0 x \leq y \implies z \cdot_0 \text{star0 } x \leq y$ 

class domain-semiring0 = dioid0 + dom0-op +
  assumes d0-absorb:  $x \leq \text{dom0 } x \cdot_0 x$ 
  and d0-local:  $\text{dom0 } (x \cdot_0 \text{dom0 } y) = \text{dom0 } (x \cdot_0 y)$ 
  and d0-add:  $\text{dom0 } (x + y) = \text{dom0 } x + \text{dom0 } y$ 
  and d0-subid:  $\text{dom0 } x \leq 1_0$ 
  and d0-zero:  $\text{dom0 } 0 = 0$ 

class codomain-semiring0 = dioid0 + cod0-op +
  assumes cod0-absorb:  $x \leq x \cdot_0 \text{cod0 } x$ 
  and cod0-local:  $\text{cod0 } (\text{cod0 } x \cdot_0 y) = \text{cod0 } (x \cdot_0 y)$ 
  and cod0-add:  $\text{cod0 } (x + y) = \text{cod0 } x + \text{cod0 } y$ 
  and cod0-subid:  $\text{cod0 } x \leq 1_0$ 
  and cod0-zero:  $\text{cod0 } 0 = 0$ 

class modal-semiring0 = domain-semiring0 + codomain-semiring0 +
  assumes dc-compat0:  $\text{dom0 } (\text{cod0 } x) = \text{cod0 } x$ 
  and cd-compat0:  $\text{cod0 } (\text{dom0 } x) = \text{dom0 } x$ 

class modal-kleene-algebra0 = modal-semiring0 + kleene-algebra0

```

**sublocale** *monoid-mult0*  $\subseteq$  *mm0*: *monoid-mult*  $1_0$   $(\cdot_0)$   
**by** (*unfold-locales*, *simp-all* *add*: *local.par-assoc0* *local.comp0-unl* *local.comp0-unr*)

**sublocale** *diod0*  $\subseteq$  *d0*: *diod-one-zero*  $(\cdot_0)$   $1_0$  - - -  
**by** (*unfold-locales*, *simp-all* *add*: *local.distl0* *local.distr0* *annil0* *annir0*)

**sublocale** *diod0*  $\subseteq$  *dd0*: *diod0* - - - -  $\lambda x y. y \cdot_0 x$  -  
**by** *unfold-locales* (*simp-all* *add*: *local.mm0.mult-assoc* *local.d0.distrib-left*)

**sublocale** *kleene-algebra0*  $\subseteq$  *k0*: *kleene-algebra*  $(\cdot_0)$   $1_0$  - - - *star0*  
**apply** *unfold-locales*  
**using** *local.star0-unfoldl* **apply** *blast*  
**by** (*simp-all* *add*: *local.star0-inductl* *local.star0-inductr* *local.star0-unfoldl*)

**sublocale** *kleene-algebra0*  $\subseteq$  *dk0*: *kleene-algebra0* - - - -  $\lambda x y. y \cdot_0 x$  - -  
**by** (*unfold-locales*, *simp-all* *add*: *local.star0-inductr* *local.star0-inductl*)

**sublocale** *domain-semiring0*  $\subseteq$  *d0*: *domain-semiring*  $(\cdot_0)$   $1_0$  - *dom0* - -  
**apply** *unfold-locales*  
**apply** (*simp* *add*: *local.d0-absorb* *local.join.sup-absorb2*)  
**apply** (*simp* *add*: *local.d0-local*)  
**apply** (*simp* *add*: *local.d0-subid* *local.join.sup-absorb2*)  
**apply** (*simp* *add*: *local.d0-zero*)  
**by** (*simp* *add*: *local.d0-add*)

**sublocale** *codomain-semiring0*  $\subseteq$  *cs0*: *range-semiring*  $(\cdot_0)$   $1_0$  - *cod0* - -  
**apply** *unfold-locales*  
**apply** (*simp* *add*: *local.cod0-absorb* *local.join.sup-absorb2*)  
**apply** (*simp* *add*: *local.cod0-local*)  
**apply** (*simp* *add*: *local.cod0-subid* *local.join.sup-absorb2*)  
**apply** (*simp* *add*: *local.cod0-zero*)  
**by** (*simp* *add*: *local.cod0-add*)

**sublocale** *codomain-semiring0*  $\subseteq$  *ds0dual*: *domain-semiring0* - - - -  $\lambda x y. y \cdot_0 x$  -  
*cod0*  
**by** *unfold-locales* *simp-all*

**sublocale** *modal-semiring0*  $\subseteq$  *msr0*: *dr-modal-semiring*  $(\cdot_0)$   $1_0$  - *dom0* - - *cod0*  
**by** (*unfold-locales*, *simp-all* *add*: *local.dc-compat0* *local.cd-compat0*)

**sublocale** *modal-semiring0*  $\subseteq$  *msr0dual*: *modal-semiring0* *dom0* - - - -  $\lambda x y. y \cdot_0$   
*x - cod0*  
**by** *unfold-locales* *simp-all*

**sublocale** *modal-kleene-algebra0*  $\subseteq$  *mka0*: *dr-modal-kleene-algebra*  $(\cdot_0)$   $1_0$  - - -  
*star0* *dom0* *cod0*..

**sublocale** *modal-kleene-algebra0*  $\subseteq$  *mka0dual*: *modal-kleene-algebra0* - - - -  $\lambda x y.$   
 $y \cdot_0 x$  - - *dom0* *cod0*..



### 3.2 Copies for 1-structures

```

class comp1-op =
  fixes comp1 :: 'a ⇒ 'a ⇒ 'a (infixl ·1 70)

class id1-op =
  fixes id1 :: 'a (11)

class star1-op =
  fixes star1 :: 'a ⇒ 'a

class dom1-op =
  fixes dom1 :: 'a ⇒ 'a

class cod1-op =
  fixes cod1 :: 'a ⇒ 'a

class monoid-mult1 = comp1-op + id1-op +
  assumes par-assoc1: x ·1 (y ·1 z) = (x ·1 y) ·1 z
  and comp1-unl: 11 ·1 x = x
  and comp1-unr: x ·1 11 = x

class dioid1 = monoid-mult1 + join-semilattice-zero +
  assumes distl1: x ·1 (y + z) = x ·1 y + x ·1 z
  and distr1: (x + y) ·1 z = x ·1 z + y ·1 z
  and annil1: 0 ·1 x = 0
  and annir1: x ·1 0 = 0

class kleene-algebra1 = dioid1 + star1-op +
  assumes star1-unfoldl: 11 + x ·1 star1 x ≤ star1 x
  and star1-unfoldr: 11 + star1 x ·1 x ≤ star1 x
  and star1-inductl: z + x ·1 y ≤ y ⇒ star1 x ·1 z ≤ y
  and star1-inductr: z + y ·1 x ≤ y ⇒ z ·1 star1 x ≤ y

class domain-semiring1 = dioid1 + dom1-op +
  assumes d1-absorb: x ≤ dom1 x ·1 x
  and d1-local: dom1 (x ·1 dom1 y) = dom1 (x ·1 y)
  and d1-add: dom1 (x + y) = dom1 x + dom1 y
  and d1-subid: dom1 x ≤ 11
  and d1-zero: dom1 0 = 0

class codomain-semiring1 = dioid1 + cod1-op +
  assumes cod1-absorb: x ≤ x ·1 cod1 x
  and cod1-local: cod1 (cod1 x ·1 y) = cod1 (x ·1 y)
  and cod1-add: cod1 (x + y) = cod1 x + cod1 y
  and cod1-subid: cod1 x ≤ 11
  and cod1-zero: cod1 0 = 0

class modal-semiring1 = domain-semiring1 + codomain-semiring1 +
  assumes dc-compat1: dom1 (cod1 x) = cod1 x

```

```

and cd-compat1:  $\text{cod}_1 (\text{dom}_1 x) = \text{dom}_1 x$ 

class modal-kleene-algebra1 = modal-semiring1 + kleene-algebra1

sublocale monoid-mult1  $\subseteq$  mm1: monoid-mult  $1_1 (\cdot_1)$ 
  by (unfold-locales, simp-all add: local.par-assoc1 comp1-unl comp1-unr)

sublocale diod1  $\subseteq$  d1: diod-one-zero -  $(\cdot_1) 1_1 - - -$ 
  by (unfold-locales, simp-all add: local.distl1 local.distr1 local.annil1 local.annir1)

sublocale diod1  $\subseteq$  dd1: diod1 - - - -  $\lambda x y. y \cdot_1 x 1_1$ 
  apply unfold-locales
  apply simp-all
  apply (simp add: local.mm1.mult-assoc)
  by (simp add: local.d1.distrib-left)

sublocale kleene-algebra1  $\subseteq$  k1: kleene-algebra -  $(\cdot_1) 1_1 - - - \text{star1}$ 
  apply unfold-locales
  using local.star1-unfold1 apply blast
  apply (simp add: local.star1-inductl)
  by (simp add: local.star1-inductr)

sublocale kleene-algebra1  $\subseteq$  dk1: kleene-algebra1 - - - -  $\lambda x y. y \cdot_1 x 1_1 \text{star1}$ 
  by (unfold-locales, simp-all add: local.star1-inductr local.star1-inductl)

sublocale domain-semiring1  $\subseteq$  dsr1: domain-semiring -  $(\cdot_1) 1_1 - \text{dom}_1 - -$ 
  apply unfold-locales
  using local.d1-absorb local.join.sup-absorb2 apply force
  apply (simp add: local.d1-local)
  using local.d1-subid local.join.sup-absorb2 apply force
  using local.d1-zero apply fastforce
  by (simp add: local.d1-add)

sublocale codomain-semiring1  $\subseteq$  csr1: range-semiring -  $(\cdot_1) 1_1 - \text{cod}_1 - -$ 
  apply unfold-locales
  apply (simp add: local.cod1-absorb local.join.sup-absorb2)
  apply (simp add: local.cod1-local)
  apply (simp add: local.cod1-subid local.join.sup-absorb2)
  using local.cod1-zero apply fastforce
  by (simp add: local.cod1-add)

sublocale codomain-semiring1  $\subseteq$  ds1dual: domain-semiring1 - - - -  $\lambda x y. y \cdot_1 x - \text{cod}_1$ 
  by (unfold-locales, simp-all add: local.cod1-absorb local.cod1-local local.cod1-add local.cod1-subid)

sublocale modal-semiring1  $\subseteq$  msr1: dr-modal-semiring -  $(\cdot_1) 1_1 - \text{dom}_1 - - \text{cod}_1$ 
  apply unfold-locales
  apply (simp add: local.dc-compat1)

```

**by** (*simp add: local.cd-compat1*)

**sublocale** *modal-semiring1*  $\subseteq$  *msr1dual: modal-semiring1*  $\text{dom}_1$  - - -  $\lambda x y. y \cdot_1 x - \text{cod}_1$   
**by** *unfold-locales simp-all*

**sublocale** *modal-kleene-algebra1*  $\subseteq$  *mka1: dr-modal-kleene-algebra - (\cdot\_1) 1\_1* - - -  
*star1 dom\_1 cod\_1..*

**sublocale** *modal-kleene-algebra1*  $\subseteq$  *mka1dual: modal-kleene-algebra1* - - -  $\lambda x y. y \cdot_1 x - \text{dom}_1 \text{cod}_1..$

### 3.3 Globular 2-semirings

**class** *two-semiring* = *modal-semiring0* + *modal-semiring1* +  
**assumes** *interchange*:  $(w \cdot_1 x) \cdot_0 (y \cdot_1 z) \leq (w \cdot_0 y) \cdot_1 (x \cdot_0 z)$   
**and** *d1-hom*:  $\text{dom}_1 (x \cdot_0 y) \leq \text{dom}_1 x \cdot_0 \text{dom}_1 y$   
**and** *c1-hom*:  $\text{cod}_1 (x \cdot_0 y) \leq \text{cod}_1 x \cdot_0 \text{cod}_1 y$   
**and** *d0-hom*:  $\text{dom}_0 (x \cdot_1 y) \leq \text{dom}_0 x \cdot_1 \text{dom}_0 y$   
**and** *c0-hom*:  $\text{cod}_0 (x \cdot_1 y) \leq \text{cod}_0 x \cdot_1 \text{cod}_0 y$   
**and** *d1d0 [simp]*:  $\text{dom}_1 (\text{dom}_0 x) = \text{dom}_0 x$

**class** *strong-two-semiring* = *two-semiring* +  
**assumes** *d1-strong-hom [simp]*:  $\text{dom}_1 (x \cdot_0 y) = \text{dom}_1 x \cdot_0 \text{dom}_1 y$   
**and** *c1-strong-hom*:  $\text{cod}_1 (x \cdot_0 y) = \text{cod}_1 x \cdot_0 \text{cod}_1 y$

**sublocale** *two-semiring*  $\subseteq$  *tgsdual: two-semiring*  $\text{dom}_0$  - - -  $\lambda x y. y \cdot_0 x - \text{cod}_0$   
 $\text{dom}_1 \lambda x y. y \cdot_1 x - \text{cod}_1$   
**apply** *unfold-locales*  
**apply** (*simp-all add: local.interchange local.d0-hom local.c0-hom local.c1-hom local.d1-hom*)  
**by** (*metis local.cd-compat1 local.d1d0 local.dc-compat0*)

**sublocale** *strong-two-semiring*  $\subseteq$  *stgsdual: strong-two-semiring*  $\text{dom}_0$  - - -  $\lambda x y. y \cdot_0 x - \text{cod}_0$   
 $\text{dom}_1 \lambda x y. y \cdot_1 x - \text{cod}_1$   
**apply** *unfold-locales by (simp-all add: local.c1-strong-hom)*

**context** *two-semiring*  
**begin**

**lemma** *c1d0 [simp]*:  $\text{cod}_1 (\text{dom}_0 x) = \text{dom}_0 x$   
**proof** –  
**have**  $\text{cod}_1 (\text{dom}_0 x) = \text{cod}_1 (\text{dom}_1 (\text{dom}_0 x))$   
**by** *simp*  
**also have**  $\dots = \text{dom}_1 (\text{dom}_0 x)$   
**using** *local.cd-compat1* **by** *presburger*  
**also have**  $\dots = \text{dom}_0 x$   
**by** *simp*  
**finally show** *?thesis*.

qed

**lemma** *d1c0* [*simp*]:  $dom_1 (cod_0 x) = cod_0 x$   
by (*simp add: msr1.d-cod-fix*)

**lemma** *c1c0* [*simp*]:  $cod_1 (cod_0 x) = cod_0 x$   
by *simp*

**lemma**  $1_1 \cdot_0 1_1 \leq 1_1$

oops

**lemma** *id1-comp0-var*:  $1_1 \leq 1_1 \cdot_0 1_1$

**proof** –

have  $1_1 = 1_1 \cdot_0 1_0$

by *simp*

also have  $\dots = (1_1 \cdot_1 1_1) \cdot_0 (1_0 \cdot_1 1_1)$

by *simp*

also have  $\dots \leq (1_1 \cdot_0 1_0) \cdot_1 (1_1 \cdot_0 1_1)$

using *local.interchange* by *presburger*

also have  $\dots = 1_1 \cdot_1 (1_1 \cdot_0 1_1)$

by *simp*

also have  $\dots = 1_1 \cdot_0 1_1$

by *simp*

finally show *?thesis*.

qed

**lemma**  $1_1 \cdot_0 1_1 = 1_1$

oops

**lemma** *id0-le-id1*:  $1_0 \leq 1_1$

**proof** –

have  $1_0 = 1_0 \cdot_0 1_0$

by *simp*

also have  $\dots = (1_1 \cdot_1 1_0) \cdot_0 (1_0 \cdot_1 1_1)$

by *simp*

also have  $\dots \leq (1_1 \cdot_0 1_0) \cdot_1 (1_0 \cdot_0 1_1)$

using *local.interchange* by *blast*

also have  $\dots = 1_1 \cdot_1 1_1$

by *simp*

also have  $\dots = 1_1$

by *simp*

finally show *?thesis*.

qed

**lemma** *id0-comp1-eq* [*simp*]:  $1_0 \cdot_1 1_0 = 1_0$

**proof** –

have  $1_0 \cdot_1 1_0 = dom_0 1_0 \cdot_1 dom_0 1_0$

by *simp*  
 also have  $\dots = \text{dom}_1 (\text{dom}_0 1_0) \cdot_1 \text{dom}_0 1_0$   
 using *local.d1d0* by *presburger*  
 also have  $\dots = \text{dom}_0 1_0$   
 by *simp*  
 finally show *?thesis*  
 by *simp*  
 qed

**lemma** *d1-id0* [*simp*]:  $\text{dom}_1 1_0 = 1_0$   
**proof**–  
 have  $\text{dom}_1 1_0 = \text{dom}_1 (\text{dom}_0 1_0)$   
 by *simp*  
 also have  $\dots = \text{dom}_0 1_0$   
 using *local.d1d0* by *blast*  
 also have  $\dots = 1_0$   
 by *simp*  
 finally show *?thesis*.  
 qed

**lemma** *d0-id1* [*simp*]:  $\text{dom}_0 1_1 = 1_0$   
**proof**–  
 have  $a: \text{dom}_0 1_1 \leq 1_0$   
 by *simp*  
 have  $1_0 \leq 1_1$   
 by (*simp add: local.id0-le-id1*)  
 hence  $1_0 \leq \text{dom}_0 1_1$   
 using *local.dsr0.d-iso* by *fastforce*  
 thus *?thesis*  
 by (*simp add: local.dual-order.antisym*)  
 qed

**lemma** *c0-id1*:  $\text{cod}_0 1_1 = 1_0$   
 using *id0-le-id1 local.csr0.rdual.dom-iso local.dual-order.antisym* by *fastforce*

**lemma** *c0-id0*:  $\text{cod}_1 1_0 = 1_0$   
 using *c1d0 d0-id1* by *blast*

**lemma** *comm-d0d1*:  $\text{dom}_0 (\text{dom}_1 x) = \text{dom}_1 (\text{dom}_0 x)$   
**proof**–  
 have  $\text{dom}_0 (\text{dom}_1 x) = \text{dom}_0 (\text{dom}_1 (\text{dom}_0 x \cdot_0 x))$   
 by *simp*  
 also have  $\dots \leq \text{dom}_0 (\text{dom}_1 (\text{dom}_0 x) \cdot_0 \text{dom}_1 x)$   
 using *local.d1-hom local.dsr0.d-iso* by *blast*  
 also have  $\dots = \text{dom}_0 (\text{dom}_0 x \cdot_0 \text{dom}_1 x)$   
 by *simp*  
 also have  $\dots = \text{dom}_0 x \cdot_0 \text{dom}_0 (\text{dom}_1 x)$   
 by *simp*  
 also have  $\dots = \text{dom}_1 (\text{dom}_0 x) \cdot_0 \text{dom}_0 (\text{dom}_1 x)$

by *simp*  
 also have  $\dots \leq \text{dom}_1 (\text{dom}_0 x) \cdot_0 1_0$   
 using *d0.mult-isol local.d0-subid* by *blast*  
 finally have  $a: \text{dom}_0 (\text{dom}_1 x) \leq \text{dom}_1 (\text{dom}_0 x)$   
 by *simp*  
 have  $\text{dom}_1 (\text{dom}_0 x) = \text{dom}_0 x$   
 by *simp*  
 also have  $\dots = \text{dom}_0 (\text{dom}_1 x \cdot_1 x)$   
 by *simp*  
 also have  $\dots \leq \text{dom}_0 (\text{dom}_1 x) \cdot_1 \text{dom}_0 x$   
 using *local.d0-hom* by *blast*  
 also have  $\dots \leq \text{dom}_0 (\text{dom}_1 x) \cdot_1 1_0$   
 by (*simp add: d1.mult-isol*)  
 also have  $\dots \leq \text{dom}_0 (\text{dom}_1 x) \cdot_1 1_1$   
 using *d1.mult-isol local.id0-le-id1* by *presburger*  
 finally have  $\text{dom}_1 (\text{dom}_0 x) \leq \text{dom}_0 (\text{dom}_1 x)$   
 by *simp*  
 thus *?thesis*  
 using *a* by *auto*  
 qed

lemma *comm-d0c1*:  $\text{dom}_0 (\text{cod}_1 x) = \text{cod}_1 (\text{dom}_0 x)$

proof–

have  $\text{dom}_0 (\text{cod}_1 x) = \text{dom}_0 (\text{cod}_1 (\text{dom}_0 x \cdot_0 x))$   
 by *simp*  
 also have  $\dots \leq \text{dom}_0 (\text{cod}_1 (\text{dom}_0 x) \cdot_0 \text{cod}_1 x)$   
 using *local.c1-hom local.dsr0.d-iso* by *blast*  
 also have  $\dots = \text{dom}_0 (\text{dom}_0 x \cdot_0 \text{cod}_1 x)$   
 by *simp*  
 also have  $\dots = \text{dom}_0 x \cdot_0 \text{dom}_0 (\text{cod}_1 x)$   
 by *simp*  
 also have  $\dots = \text{cod}_1 (\text{dom}_0 x) \cdot_0 \text{dom}_0 (\text{cod}_1 x)$   
 by *simp*  
 also have  $\dots \leq \text{cod}_1 (\text{dom}_0 x) \cdot_0 1_0$   
 using *d0.mult-isol local.d0-subid* by *blast*  
 finally have  $a: \text{dom}_0 (\text{cod}_1 x) \leq \text{cod}_1 (\text{dom}_0 x)$   
 by *simp*  
 have  $\text{cod}_1 (\text{dom}_0 x) = \text{dom}_0 x$   
 by *simp*  
 also have  $\dots = \text{dom}_0 (x \cdot_1 \text{cod}_1 x)$   
 by *simp*  
 also have  $\dots \leq \text{dom}_0 x \cdot_1 \text{dom}_0 (\text{cod}_1 x)$   
 using *local.d0-hom* by *blast*  
 also have  $\dots \leq 1_0 \cdot_1 \text{dom}_0 (\text{cod}_1 x)$   
 by (*simp add: d1.mult-isol*)  
 also have  $\dots \leq 1_1 \cdot_1 \text{dom}_0 (\text{cod}_1 x)$   
 using *d1.mult-isol local.id0-le-id1* by *blast*  
 finally have  $\text{cod}_1 (\text{dom}_0 x) \leq \text{dom}_0 (\text{cod}_1 x)$   
 by *simp*

**thus** *?thesis*  
**using** *a* **by** *auto*  
**qed**

**lemma** *comm-c0c1*:  $\text{cod}_0 (\text{cod}_1 x) = \text{cod}_1 (\text{cod}_0 x)$   
**by** (*metis c1c0 local.csr0.rdual.dom-llp local.csr0.rdual.dom-ord local.csr0.rdual.dsg1 local.csr0.rdual.dsg4 local.csr1.rdual.dom-llp local.csr1.rdual.dsg1 local.tgsdual.d0-hom local.tgsdual.d1-hom*)

**lemma** *comm-c0d1*:  $\text{cod}_0 (\text{dom}_1 x) = \text{dom}_1 (\text{cod}_0 x)$   
**by** (*metis d1c0 local.c0-hom local.csr0.rdual.dom-subid-aux2 local.csr0.rdual.domain-1'' local.csr0.rdual.domain-invol local.csr0.rdual.dsg1 local.d1-hom local.dsr1.dom-subid-aux2 local.dsr1.dom-subid-aux2'' local.dsr1.dsg1 local.dual-order.antisym*)

We prove further lemmas that are not related to the globular structure.

**lemma** *d0-comp1-idem* [*simp*]:  $\text{dom}_0 x \cdot_1 \text{dom}_0 x = \text{dom}_0 x$

**proof** –

**have**  $\text{dom}_0 x \cdot_1 \text{dom}_0 x = \text{dom}_1 (\text{dom}_0 x) \cdot_1 \text{dom}_1 (\text{dom}_0 x)$

**by** *simp*

**also have**  $\dots = \text{dom}_1 (\text{dom}_0 x)$

**using** *local.dsr1.dom-el-idem* **by** *blast*

**also have**  $\dots = \text{dom}_0 x$

**by** *simp*

**finally show** *?thesis*.

**qed**

**lemma** *cod0-comp1-idem*:  $\text{cod}_0 x \cdot_1 \text{cod}_0 x = \text{cod}_0 x$

**by** (*metis d1c0 local.dsr1.dsg1*)

**lemma** *dom01-loc* [*simp*]:  $\text{dom}_0 (x \cdot_1 \text{dom}_1 y) = \text{dom}_0 (x \cdot_1 y)$

**proof** –

**have**  $\text{dom}_0 (x \cdot_1 y) = \text{dom}_0 (\text{dom}_1 (x \cdot_1 y))$

**by** (*simp add: local.comm-d0d1*)

**also have**  $\dots = \text{dom}_0 (\text{dom}_1 (x \cdot_1 \text{dom}_1 y))$

**by** *simp*

**also have**  $\dots = \text{dom}_0 (x \cdot_1 \text{dom}_1 y)$

**using** *local.comm-d0d1 local.d1d0* **by** *presburger*

**finally show** *?thesis..*

**qed**

**lemma** *cod01-loc1*:  $\text{cod}_0 (\text{cod}_1 x \cdot_1 y) = \text{cod}_0 (x \cdot_1 y)$

**by** (*metis c1c0 comm-c0c1 local.cod1-local*)

**lemma** *dom01-exp* [*simp*]:  $\text{dom}_0 (\text{cod}_1 x \cdot_1 y) = \text{dom}_0 (x \cdot_1 y)$

**by** (*metis local.c1d0 local.cod1-local local.comm-d0c1*)

**lemma** *cod01-exo*:  $\text{cod}_0 (x \cdot_1 \text{dom}_1 y) = \text{cod}_0 (x \cdot_1 y)$

**by** (*metis comm-c0d1 d1c0 local.d1-local*)

**lemma** *dom01-loc-var* [*simp*]:  $\text{dom}_0 (x \cdot_0 \text{dom}_1 y) = \text{dom}_0 (x \cdot_0 y)$

**proof** –

**have**  $\text{dom}_0 (x \cdot_0 y) = \text{dom}_0 (x \cdot_0 \text{dom}_0 y)$   
**by** *simp*  
**also have**  $\dots = \text{dom}_0 (x \cdot_0 \text{dom}_1 (\text{dom}_0 y))$   
**by** *simp*  
**also have**  $\dots = \text{dom}_0 (x \cdot_0 \text{dom}_0 (\text{dom}_1 y))$   
**by** (*simp add: local.comm-d0d1*)  
**also have**  $\dots = \text{dom}_0 (x \cdot_0 \text{dom}_1 y)$   
**by** *simp*  
**finally show** *?thesis.*  
**qed**

**lemma** *cod01-loc-var*:  $\text{cod}_0 (\text{cod}_1 x \cdot_0 y) = \text{cod}_0 (x \cdot_0 y)$

**by** (*metis c1c0 comm-c0c1 local.cod0-local*)

**lemma** *dom0cod1-exp*:  $\text{dom}_0 (x \cdot_0 y) \leq \text{dom}_0 (\text{cod}_1 x \cdot_0 y)$

**proof** –

**have**  $\text{dom}_0 (x \cdot_0 y) = \text{dom}_0 (\text{cod}_1 (x \cdot_0 y))$   
**by** (*simp add: local.comm-d0c1*)  
**also have**  $\dots \leq \text{dom}_0 (\text{cod}_1 x \cdot_0 \text{cod}_1 y)$   
**by** (*simp add: local.c1-hom local.dsr0.d-iso*)  
**also have**  $\dots = \text{dom}_0 (\text{cod}_1 x \cdot_0 \text{dom}_0 (\text{cod}_1 y))$   
**by** *simp*  
**also have**  $\dots = \text{dom}_0 (\text{cod}_1 x \cdot_0 \text{dom}_0 y)$   
**by** (*simp add: local.comm-d0c1*)  
**also have**  $\dots = \text{dom}_0 (\text{cod}_1 x \cdot_0 y)$   
**by** *simp*  
**finally show** *?thesis.*  
**qed**

**lemma** *cod0dom1-exp*:  $\text{cod}_0 (x \cdot_0 y) \leq \text{cod}_0 (x \cdot_0 \text{dom}_1 y)$

**by** (*metis comm-c0d1 d1c0 local.cod0-local local.csr0.rdual.dom-iso local.d1-hom*)

**lemma** (*in two-semiring*) *d0-comp1*:  $\text{dom}_0 x \cdot_0 (y \cdot_1 z) \leq (\text{dom}_0 x \cdot_0 y) \cdot_1 (\text{dom}_0 x \cdot_0 z)$

**proof** –

**have**  $\text{dom}_0 x \cdot_0 (y \cdot_1 z) = (\text{dom}_0 x \cdot_1 \text{dom}_0 x) \cdot_0 (y \cdot_1 z)$   
**by** *simp*  
**also have**  $\dots \leq (\text{dom}_0 x \cdot_0 y) \cdot_1 (\text{dom}_0 x \cdot_0 z)$   
**using** *local.interchange* **by** *presburger*  
**finally show** *?thesis.*  
**qed**

**lemma** *d1-comp1*:  $\text{dom}_1 x \cdot_0 (y \cdot_1 z) \leq (\text{dom}_1 x \cdot_0 y) \cdot_1 (\text{dom}_1 x \cdot_0 z)$

**by** (*metis local.dsr1.dom-el-idem local.tgsdual.interchange*)

**lemma** *d01-export*:  $\text{dom}_0 (\text{dom}_1 x \cdot_1 y) \leq \text{dom}_0 x \cdot_1 \text{dom}_0 y$

**proof** –



**have**  $dom_0 (dom_1 x \cdot_1 y) \leq dom_0 (dom_1 x) \cdot_1 dom_0 y$   
**by** (*simp add: local.d0-hom*)  
**also have**  $\dots = dom_0 x \cdot_1 dom_0 y$   
**by** (*simp add: local.comm-d0d1*)  
**finally show** *?thesis*.  
**qed**

**lemma** *cod01-export*:  $cod_0 (x \cdot_1 cod_1 y) \leq cod_0 x \cdot_1 cod_0 y$   
**by** (*metis local.c0-hom local.c1c0 local.comm-c0c1*)

**lemma** *d10-export* [*simp*]:  $dom_1 (dom_0 x \cdot_1 y) = dom_0 x \cdot_1 dom_1 y$   
**by** (*metis local.d1d0 local.dsr1.dsg3*)

**lemma** *cod10-export*:  $cod_1 (x \cdot_1 cod_0 y) = cod_1 x \cdot_1 cod_0 y$   
**by** (*metis local.c1c0 local.csr1.rdual.dsg3*)

**lemma** *d0-comp10*:  $dom_0 x \cdot_1 dom_0 y = dom_0 x \cdot_0 dom_0 y$   
**proof** (*rule order.antisym*)

**have**  $dom_0 x \cdot_1 dom_0 y \leq dom_0 (dom_0 x \cdot_1 dom_0 y) \cdot_0 (dom_0 x \cdot_1 dom_0 y)$   
**by** *simp*  
**also have**  $\dots \leq (dom_0 (dom_0 x) \cdot_1 dom_0 (dom_0 y)) \cdot_0 (dom_0 x \cdot_1 dom_0 y)$   
**using** *d0.mult-isor local.tgsdual.c0-hom* **by** *blast*  
**also have**  $\dots \leq (dom_0 x \cdot_1 1_0) \cdot_0 (1_0 \cdot_1 dom_0 y)$   
**by** (*simp add: local.d0.mult-isol-var local.d1.mult-isol-var*)  
**also have**  $\dots \leq (dom_0 x \cdot_1 1_1) \cdot_0 (1_1 \cdot_1 dom_0 y)$   
**using** *local.d0.mult-isol-var local.dd1.d1.mult-isol local.dd1.d1.mult-isor local.id0-le-id1* **by** *presburger*  
**also have**  $\dots \leq dom_0 x \cdot_0 dom_0 y$   
**by** *simp*  
**finally show**  $dom_0 x \cdot_1 dom_0 y \leq dom_0 x \cdot_0 dom_0 y$ .

**next**

**have**  $dom_0 x \cdot_0 dom_0 y = (dom_0 x \cdot_1 dom_0 x) \cdot_0 (dom_0 y \cdot_1 dom_0 y)$   
**by** *simp*  
**also have**  $\dots \leq (dom_0 x \cdot_0 dom_0 y) \cdot_1 (dom_0 x \cdot_0 dom_0 y)$   
**using** *local.interchange* **by** *blast*  
**also have**  $\dots \leq (dom_0 x \cdot_0 1_0) \cdot_1 (1_0 \cdot_0 dom_0 y)$   
**by** (*simp add: local.d1.mult-isol-var local.dsr0.dom-subid-aux2 local.dsr0.dom-subid-aux2''*)  
**also have**  $\dots = dom_0 x \cdot_1 dom_0 y$   
**by** *simp*  
**finally show**  $dom_0 x \cdot_0 dom_0 y \leq dom_0 x \cdot_1 dom_0 y$ .  
**qed**

**lemma** *dom-exchange-strong*:  $(dom_0 w \cdot_1 dom_0 x) \cdot_0 (dom_0 y \cdot_1 dom_0 z) = (dom_0 w \cdot_0 dom_0 y) \cdot_1 (dom_0 x \cdot_0 dom_0 z)$

**proof** –

**have**  $(dom_0 w \cdot_1 dom_0 x) \cdot_0 (dom_0 y \cdot_1 dom_0 z) = (dom_0 w \cdot_0 dom_0 x) \cdot_0 (dom_0 y \cdot_0 dom_0 z)$   
**by** (*simp add: local.d0-comp10*)  
**also have**  $\dots = (dom_0 w \cdot_0 dom_0 y) \cdot_0 (dom_0 x \cdot_0 dom_0 z)$

by (*metis local.dd0.mm0.mult-assoc local.dsr0.dsg4*)  
 also have  $\dots = \text{dom}_0 (\text{dom}_0 w \cdot_0 \text{dom}_0 y) \cdot_0 \text{dom}_0 (\text{dom}_0 x \cdot_0 \text{dom}_0 z)$   
 by *simp*  
 also have  $\dots = \text{dom}_0 (\text{dom}_0 w \cdot_0 \text{dom}_0 y) \cdot_1 \text{dom}_0 (\text{dom}_0 x \cdot_0 \text{dom}_0 z)$   
 using *local.d0-comp10* by *presburger*  
 also have  $\dots = (\text{dom}_0 w \cdot_0 \text{dom}_0 y) \cdot_1 (\text{dom}_0 x \cdot_0 \text{dom}_0 z)$   
 by *simp*  
 finally show *?thesis*.  
 qed

end

context *strong-two-semiring*  
 begin

lemma *id1-comp0*:  $1_1 \cdot_0 1_1 \leq 1_1$   
 proof –  
 have  $1_1 \cdot_0 1_1 = \text{dom}_1 1_1 \cdot_0 \text{dom}_1 1_1$   
 by *simp*  
 also have  $\dots = \text{dom}_1 (1_1 \cdot_0 1_1)$   
 by *simp*  
 also have  $\dots \leq 1_1$   
 using *local.d1-subid* by *blast*  
 finally show *?thesis*.  
 qed

lemma *id1-comp0-eq [simp]*:  $1_1 \cdot_0 1_1 = 1_1$   
 proof –  
 have  $1_1 = 1_1 \cdot_0 1_0$   
 by *simp*  
 also have  $\dots = (1_1 \cdot_1 1_1) \cdot_0 (1_0 \cdot_1 1_1)$   
 by *simp*  
 also have  $\dots \leq (1_1 \cdot_0 1_0) \cdot_1 (1_1 \cdot_0 1_1)$   
 using *local.interchange* by *presburger*  
 also have  $\dots = 1_1 \cdot_1 (1_1 \cdot_0 1_1)$   
 by *simp*  
 also have  $\dots = 1_1 \cdot_0 1_1$   
 by *simp*  
 finally have  $1_1 \leq 1_1 \cdot_0 1_1$ .  
 thus *?thesis*  
 by (*simp add: local.antisym-conv local.id1-comp0*)  
 qed

lemma  $1_0 = 1_1$

oops

lemma *dom0cod1-exp*:  $\text{dom}_0 (x \cdot_0 y) = \text{dom}_0 (\text{cod}_1 x \cdot_0 y)$   
 proof –

```

have  $dom_0 (x \cdot_0 y) = dom_0 (cod_1 (x \cdot_0 y))$ 
  using local.c1d0 local.comm-d0c1 by presburger
also have  $\dots = dom_0 (cod_1 x \cdot_0 cod_1 y)$ 
  by (simp add: local.c1-hom local.dsr0.d-iso)
also have  $\dots = dom_0 (cod_1 x \cdot_0 dom_0 (cod_1 y))$ 
  by simp
also have  $\dots = dom_0 (cod_1 x \cdot_0 dom_0 y)$ 
  by (simp add: local.comm-d0c1)
also have  $\dots = dom_0 (cod_1 x \cdot_0 y)$ 
  by simp
finally show ?thesis.
qed

```

```

lemma cod0dom1-exp:  $cod_0 (x \cdot_0 dom_1 y) = cod_0 (x \cdot_0 y)$ 
  by (metis local.comm-c0d1 local.d1c0 local.ds0dual.d0-local local.stgsdual.c1-strong-hom)

```

**end**

The following laws are diamond laws. It remains to define diamonds for them.

```

context two-semiring
begin

```

```

lemma fdia0fdia1-prop:  $dom_0 (y \cdot_0 dom_1 (x \cdot_1 z)) = dom_0 (y \cdot_0 (x \cdot_1 z))$ 
  by simp

```

```

lemma bdia0fdia1-prop [simp]:  $cod_0 (dom_1 (x \cdot_1 z) \cdot_0 y) = cod_0 ((x \cdot_1 z) \cdot_0 y)$ 
  by (metis local.comm-c0d1 local.d1c0 local.ds0dual.d0-local)

```

```

lemma fdia0bdia1-prop:  $dom_0 (y \cdot_0 cod_1 (x \cdot_1 z)) = dom_0 (y \cdot_0 (x \cdot_1 z))$ 
  by (metis local.c1d0 local.comm-d0c1 local.d0-local)

```

```

lemma bdia0bdia1-prop:  $cod_0 (cod_1 (x \cdot_1 z) \cdot_0 y) = cod_0 ((x \cdot_1 z) \cdot_0 y)$ 
  by simp

```

```

lemma fdia0fdia1-prop2:  $dom_0 (y \cdot_0 dom_1 (x \cdot_1 z)) \leq dom_0 (y \cdot_0 (dom_0 x \cdot_1 dom_0 z))$ 

```

**proof** –

```

  have  $dom_0 (y \cdot_0 dom_1 (x \cdot_1 z)) = dom_0 (y \cdot_0 dom_0 (x \cdot_1 z))$ 

```

```

  by simp

```

```

  also have  $\dots \leq dom_0 (y \cdot_0 (dom_0 x \cdot_1 dom_0 z))$ 

```

```

  using d0.mult-isol local.dsr0.d-iso local.tgsdual.c0-hom by presburger

```

```

  finally show ?thesis.

```

**qed**

```

lemma fdia00-prop2:  $dom_0 (y \cdot_0 dom_0 (x \cdot_1 z)) \leq dom_0 (y \cdot_0 (dom_0 x \cdot_1 dom_0 z))$ 

```

```

  using local.fdia0fdia1-prop2 by auto

```

**lemma** *bdia0dom1-prop2*:  $\text{cod}_0 (\text{dom}_1 (x \cdot_1 z) \cdot_0 y) \leq \text{cod}_0 ((\text{cod}_0 x \cdot_1 \text{cod}_0 z) \cdot_0 y)$   
**using** *local.c0-hom local.csr0.rdual.fd-def local.csr0.rdual.fd-iso1 local.tgsdual.d0-comp10*  
**by** *auto*

**lemma** *bdia0dom0-prop2*:  $\text{cod}_0 (\text{dom}_0 (x \cdot_1 z) \cdot_0 y) \leq \text{cod}_0 ((\text{dom}_0 x \cdot_1 \text{dom}_0 z) \cdot_0 y)$   
**by** (*simp add: local.csr0.rdual.dom-iso local.dd0.d0.mult-isol local.tgsdual.c0-hom*)

**lemma** *fdia0bdia1-prop-2*:  $\text{dom}_0 (y \cdot_0 \text{cod}_1 (z \cdot_1 x)) \leq \text{dom}_0 (y \cdot_0 (\text{dom}_0 x \cdot_1 \text{dom}_0 z))$   
**by** (*metis fdia00-prop2 local.c1d0 local.csr1.rdual.dsg1 local.csr1.rdual.dsg4 local.dom01-exp local.msr0dual.cod0-local*)

**lemma** *fdia0bdia0-prop2*:  $\text{dom}_0 (y \cdot_0 \text{cod}_0 (z \cdot_1 x)) \leq \text{dom}_0 (y \cdot_0 (\text{cod}_0 z \cdot_1 \text{cod}_0 x))$   
**by** (*simp add: local.c0-hom local.dd0.d0.mult-isol local.dsr0.dom-iso*)

**lemma** *bdia0bdia1-prop2*:  $\text{cod}_0 (\text{cod}_1 (z \cdot_1 x) \cdot_0 y) \leq \text{cod}_0 ((\text{cod}_0 x \cdot_1 \text{cod}_0 z) \cdot_0 y)$   
**using** *bdia0dom1-prop2 local.csr0.rdual.dsg4 local.tgsdual.d0-comp10* **by** *fastforce*

**lemma** *bdia0bdia0-prop2*:  $\text{cod}_0 (\text{cod}_0 (x \cdot_1 z) \cdot_0 y) \leq \text{cod}_0 ((\text{cod}_0 x \cdot_1 \text{cod}_0 z) \cdot_0 y)$   
**using** *bdia0dom1-prop2* **by** *force*

**lemma** *fdia1fdia0-prop3* [*simp*]:  $\text{dom}_1 (x \cdot_1 \text{dom}_0 (y \cdot_0 z)) \leq \text{dom}_1 (x \cdot_1 \text{dom}_0 (\text{dom}_1 y \cdot_0 z))$   
**by** (*metis d1.mult-isol local.comm-d0d1 local.d1-hom local.d1d0 local.dsr0.d-iso local.dsr1.d-iso local.tgsdual.cod01-loc-var*)

**lemma** *fdia1bdia0-prop3* [*simp*]:  $\text{dom}_1 (x \cdot_1 \text{cod}_0 (z \cdot_0 y)) \leq \text{dom}_1 (x \cdot_1 \text{cod}_0 (z \cdot_0 \text{dom}_1 y))$   
**by** (*simp add: d1.mult-isol local.dsr1.d-iso local.tgsdual.dom0cod1-exp*)

**lemma** *bdia1fdia0-prop3*:  $\text{cod}_1 (\text{dom}_0 (y \cdot_0 z) \cdot_1 x) \leq \text{cod}_1 (\text{dom}_0 (\text{cod}_1 y \cdot_0 z) \cdot_1 x)$   
**by** (*simp add: local.csr1.rdual.dom-iso local.dd1.d1.mult-isol local.tgsdual.cod0dom1-exp*)

**lemma** *bdia1bdia0-prop3*:  $\text{cod}_1 (\text{cod}_0 (z \cdot_0 y) \cdot_1 x) \leq \text{cod}_1 (\text{cod}_0 (z \cdot_0 \text{cod}_1 y) \cdot_1 x)$   
**by** (*metis local.c1-hom local.c1c0 local.comm-c0c1 local.csr0.rdual.dom-iso local.csr1.rdual.dom-iso local.dd1.d1.mult-isol local.tgsdual.dom01-loc-var*)

**end**

**context** *strong-two-semiring*  
**begin**

**lemma** *fdia1fdia0-prop3* [*simp*]:  $\text{dom}_1 (x \cdot_1 \text{dom}_0 (\text{dom}_1 y \cdot_0 z)) = \text{dom}_1 (x \cdot_1 \text{dom}_0 (y \cdot_0 z))$   
**by** (*metis local.comm-d0d1 local.d1-strong-hom local.d1d0 local.dom01-loc-var*)

**lemma** *fdia1bdia0-prop3* [*simp*]:  $dom_1 (x \cdot_1 cod_0 (z \cdot_0 dom_1 y)) = dom_1 (x \cdot_1 cod_0 (z \cdot_0 y))$   
**by** (*simp add: local.cod0dom1-exp*)

**lemma** *bdia1fdia0-prop3*:  $cod_1 (dom_0 (cod_1 y \cdot_0 z) \cdot_1 x) = cod_1 (dom_0 (y \cdot_0 z) \cdot_1 x)$   
**by** (*simp add: local.stgsdual.cod0dom1-exp*)

**lemma** *bdia1bdia0-prop3*:  $cod_1 (cod_0 (z \cdot_0 cod_1 y) \cdot_1 x) = cod_1 (cod_0 (z \cdot_0 y) \cdot_1 x)$   
**by** (*metis local.c1-strong-hom local.c1c0 local.cod01-loc-var local.comm-c0c1*)

**lemma** *fdia0fdia1-prop4*:  $dom_0 z \cdot_0 dom_1 (x \cdot_1 y) \leq dom_1 ((dom_0 z \cdot_0 x) \cdot_1 (dom_0 z \cdot_0 y))$

**proof**–

**have**  $dom_0 z \cdot_0 dom_1 (x \cdot_1 y) = dom_1 (dom_0 z) \cdot_0 dom_1 (x \cdot_1 y)$

**by** *simp*

**also have**  $\dots = dom_1 (dom_0 z \cdot_0 (x \cdot_1 y))$

**by** *simp*

**also have**  $\dots \leq dom_1 ((dom_0 z \cdot_0 x) \cdot_1 (dom_0 z \cdot_0 y))$

**using** *local.d0-comp1 local.dsr1.d-iso* **by** *presburger*

**finally show** *?thesis*.

**qed**

**lemma** *fdia0bdia1-prop4*:  $dom_0 z \cdot_0 cod_1 (y \cdot_1 x) \leq cod_1 ((dom_0 z \cdot_0 y) \cdot_1 (dom_0 z \cdot_0 x))$   
**by** (*metis local.c1d0 local.csr1.rdual.dom-iso local.d0-comp1 local.stgsdual.d1-strong-hom*)

**lemma** *fdia1fdia1-prop4*:  $dom_1 (x \cdot_1 y) \cdot_0 dom_0 z \leq dom_1 ((x \cdot_0 dom_0 z) \cdot_1 (y \cdot_0 dom_0 z))$

**by** (*metis local.d0-comp1-idem local.d1-strong-hom local.d1d0 local.dsr1.d-iso local.tgsdual.interchange*)

**lemma** *bdia1bdia1-prop4*:  $cod_1 (y \cdot_1 x) \cdot_0 dom_0 z \leq cod_1 ((y \cdot_0 dom_0 z) \cdot_1 (x \cdot_0 dom_0 z))$

**by** (*metis local.c1d0 local.csr1.rdual.dom-iso local.stgsdual.d1-strong-hom local.tgsdual.d1-comp1*)

**end**

### 3.4 Globular 2-Kleene algebras

**class** *two-kleene-algebra* = *two-semiring* + *kleene-algebra0* + *kleene-algebra1*

**class** *strong-two-kleene-algebra* = *strong-two-semiring* + *kleene-algebra0* + *kleene-algebra1*

**lemma** (**in** *strong-two-kleene-algebra*)  $star1 x \cdot_0 star1 y \leq star0 (x \cdot_1 y)$

```

oops

lemma (in strong-two-kleene-algebra)  $star1\ x \cdot_0\ star1\ y \leq star1\ (x \cdot_1\ y)$ 

oops

context two-kleene-algebra
begin

lemma interchange-var1:  $(x \cdot_1\ x) \cdot_0\ (y \cdot_1\ y) \cdot_0\ (z \cdot_1\ z) \leq (x \cdot_0\ y \cdot_0\ z) \cdot_1\ (x \cdot_0\ y \cdot_0\ z)$ 
  by (meson local.dd0.d0.mult-isol local.dual-order.trans local.tgsdual.interchange)

lemma interchange-var2:  $(x \cdot_1\ y) \cdot_0\ (x \cdot_1\ y) \cdot_0\ (x \cdot_1\ y) \leq (x \cdot_0\ x \cdot_0\ x) \cdot_1\ (y \cdot_0\ y \cdot_0\ y)$ 
  by (meson local.dd0.d0.mult-isol local.dual-order.trans local.tgsdual.interchange)

lemma star0-comp1:  $star0\ (x \cdot_1\ y) \leq star0\ x \cdot_1\ star0\ y$ 
proof –
  have  $a: 1_0 \leq star0\ x \cdot_1\ star0\ y$ 
    by (metis local.d1.mult-isol-var local.id0-comp1-eq local.k0.star-ref)
  have  $(x \cdot_1\ y) \cdot_0\ (star0\ x \cdot_1\ star0\ y) \leq (x \cdot_0\ star0\ x) \cdot_1\ (y \cdot_0\ star0\ y)$ 
    by (simp add: local.interchange)
  also have  $\dots \leq star0\ x \cdot_1\ star0\ y$ 
    by (simp add: local.dd1.d1.mult-isol-var)
  finally have  $(x \cdot_1\ y) \cdot_0\ (star0\ x \cdot_1\ star0\ y) \leq star0\ x \cdot_1\ star0\ y$ .
  hence  $1_0 + (x \cdot_1\ y) \cdot_0\ (star0\ x \cdot_1\ star0\ y) \leq star0\ x \cdot_1\ star0\ y$ 
    by (simp add: a)
  thus ?thesis
    using local.star0-inductl by force
qed

end

context strong-two-kleene-algebra
begin

lemma  $star1\ (x \cdot_1\ y) \leq star1\ x \cdot_0\ star1\ y$ 

oops

lemma  $star1\ x \cdot_0\ star1\ y \leq star1\ (x \cdot_0\ y)$ 

oops

lemma  $star1\ (x \cdot_0\ y) \leq star1\ x \cdot_0\ star1\ y$ 

oops

```

```

lemma  $star0\ x \cdot_1\ star0\ y \leq star0\ (x \cdot_0\ y)$ 

  oops

lemma  $star0\ (x \cdot_0\ y) \leq star0\ x \cdot_1\ star0\ y$ 

  oops

lemma  $star0\ x \cdot_1\ star0\ y \leq star0\ (x \cdot_1\ y)$ 

  oops

lemma (in strong-two-kleene-algebra)  $dom_0\ x \cdot_0\ star1\ y \leq star1\ (dom_0\ x \cdot_0\ y)$ 

  oops

end

class two-quantale-lmcs = modal-semiring0 + modal-semiring1 +
  assumes interchange:  $(w \cdot_1\ x) \cdot_0\ (y \cdot_1\ z) \leq (w \cdot_0\ y) \cdot_1\ (x \cdot_0\ z)$ 
  and d1-hom:  $dom_1\ (x \cdot_0\ y) = dom_1\ x \cdot_0\ dom_1\ y$ 
  and c1-hom:  $cod_1\ (x \cdot_0\ y) = cod_1\ x \cdot_0\ cod_1\ y$ 
  and d1d0 [simp]:  $dom_1\ (dom_0\ x) = dom_0\ x$ 
  and c1d0 [simp]:  $cod_1\ (dom_0\ x) = dom_0\ x$ 
  and d1c0 [simp]:  $dom_1\ (cod_0\ x) = cod_0\ x$ 
  and c1c0 [simp]:  $cod_1\ (cod_0\ x) = cod_0\ x$ 
  and d0d1 [simp]:  $dom_0\ (dom_1\ x) = dom_0\ x$ 
  and c0d1 [simp]:  $cod_0\ (dom_1\ x) = dom_0\ x$ 
  and d0c1 [simp]:  $dom_0\ (cod_1\ x) = cod_0\ x$ 
  and c0c1 [simp]:  $cod_0\ (cod_1\ x) = cod_0\ x$ 

begin

lemma  $dom_0\ (x \cdot_1\ y) \leq dom_0\ x \cdot_1\ dom_0\ y$ 

  oops

lemma  $cod_0\ (x \cdot_1\ y) \leq cod_0\ x \cdot_1\ cod_0\ y$ 

  oops

end

end

```

## 4 2-Quantales

```

theory Two-Quantale
  imports Quantales-Converse.Modal-Quantale Two-Kleene-Algebra

```

**begin**

**class** *quantale0* = *complete-lattice* + *monoid-mult0* +  
  **assumes** *Sup-distl0*:  $x \cdot_0 \sqcup Y = (\sqcup y \in Y. x \cdot_0 y)$   
  **assumes** *Sup-distr0*:  $\sqcup X \cdot_0 y = (\sqcup x \in X. x \cdot_0 y)$

**sublocale** *quantale0*  $\subseteq$  *q0q*: *unital-quantale 10* ( $\cdot_0$ ) - - - - -  
  **apply** *unfold-locales* **using** *local.Sup-distr0* *local.Sup-distl0* **by** *auto*

**definition** (**in** *quantale0*) *qstar0* = *q0q.qstar*

**lemma** (**in** *quantale0*) *qstar0-unfold*:  $qstar0\ x = (\sqcup i. mm0.power\ x\ i)$   
  **by** (*simp add: local.q0q.qstar-def local.qstar0-def*)

**sublocale** *quantale0*  $\subseteq$  *dq0s0*: *diodid0* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_0$ )  $1_0$   
  **by** *unfold-locales* (*simp-all add: local.q0q.sup-distl*)

**sublocale** *quantale0*  $\subseteq$  *dq0ka0*: *kleene-algebra0* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_0$ )  $1_0$  *qstar0*  
  **by** *unfold-locales* (*simp-all add: local.qstar0-def local.q0q.uwqlka.star-inductl local.q0q.uqlka.star-inductr'*)

**class** *domain-quantale0* = *quantale0* + *dom0-op* +  
  **assumes** *dom0-absorb*:  $x \leq dom_0\ x \cdot_0\ x$   
  **and** *dom0-local*:  $dom_0\ (x \cdot_0\ dom_0\ y) = dom_0\ (x \cdot_0\ y)$   
  **and** *dom0-add*:  $dom_0\ (x \sqcup y) = dom_0\ x \sqcup dom_0\ y$   
  **and** *dom0-subid*:  $dom_0\ x \leq 1_0$   
  **and** *dom0-zero*:  $dom_0\ \perp = \perp$

**sublocale** *domain-quantale0*  $\subseteq$  *dq0dq*: *domain-quantale dom0 10* ( $\cdot_0$ ) - - - - -  
  **by** *unfold-locales* (*simp-all add: local.dom0-absorb local.dom0-local local.dom0-add local.dom0-subid dom0-zero*)

**sublocale** *domain-quantale0*  $\subseteq$  *dq0ds0*: *domain-semiring0* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_0$ )  $1_0$   
*dom0*  
  **by** *unfold-locales* (*simp-all add: local.dom0-local local.dom0-add local.dom0-subid dom0-zero*)

**class** *codomain-quantale0* = *quantale0* + *cod0-op* +  
  **assumes** *cod0-absorb*:  $x \leq x \cdot_0\ cod_0\ x$   
  **and** *cod0-local*:  $cod_0\ (cod_0\ x \cdot_0\ y) = cod_0\ (x \cdot_0\ y)$   
  **and** *cod0-add*:  $cod_0\ (x \sqcup y) = cod_0\ x \sqcup cod_0\ y$   
  **and** *cod0-subid*:  $cod_0\ x \leq 1_0$   
  **and** *cod0-zero*:  $cod_0\ \perp = \perp$

**sublocale** *codomain-quantale0*  $\subseteq$  *cdq0cdq*: *codomain-quantale 10* ( $\cdot_0$ ) - - - - -  
*cod0*  
  **by** (*unfold-locales, simp-all add: local.cod0-absorb local.cod0-local local.cod0-add local.cod0-subid cod0-zero*)



**sublocale** *codomain-quantale0*  $\subseteq$  *cdq0dcs0*: *codomain-semiring0* *cod0* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  
 $\perp$  ( $\cdot_0$ )  $1_0$   
**by** (*unfold-locales*, *simp-all* *add*: *local.cod0-absorb* *local.cod0-local* *local.cod0-add*  
*local.cod0-subid* *cod0-zero*)

**class** *modal-quantale0* = *domain-quantale0* + *codomain-quantale0* +  
**assumes** *dc-compat*:  $\text{dom}_0 (\text{cod}_0 x) = \text{cod}_0 x$   
**and** *cd-compat*:  $\text{cod}_0 (\text{dom}_0 x) = \text{dom}_0 x$

**sublocale** *modal-quantale0*  $\subseteq$  *mq0mq*: *dc-modal-quantale*  $1_0$  ( $\cdot_0$ ) - - - - - *cod0*  
 $\text{dom}_0$   
**by** (*unfold-locales*, *simp-all* *add*: *dc-compat* *cd-compat*)

**sublocale** *modal-quantale0*  $\subseteq$  *mq0mka*: *modal-kleene-algebra0* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_0$ )  
 $1_0$  *qstar0* *cod0*  $\text{dom}_0$   
**by** *unfold-locales* *simp-all*

**sublocale** *modal-quantale0*  $\subseteq$  *mq0dual*: *modal-quantale0*  $\text{dom}_0$  - - - - -  $\lambda x y.$   
 $y \cdot_0 x - \text{cod}_0$   
**by** *unfold-locales* (*simp-all* *add*: *local.cdq0cdq.coddual.Sup-distl* *local.Sup-distl0*)

**class** *quantale1* = *complete-lattice* + *monoid-mult1* +  
**assumes** *Sup-distl1*:  $x \cdot_1 \sqcup Y = (\sqcup y \in Y. x \cdot_1 y)$   
**assumes** *Sup-distr1*:  $\sqcup X \cdot_1 y = (\sqcup x \in X. x \cdot_1 y)$

**sublocale** *quantale1*  $\subseteq$  *q1q*: *unital-quantale*  $1_1$  ( $\cdot_1$ ) - - - - -  
**by** (*unfold-locales*, *auto* *simp*: *local.Sup-distl1* *local.Sup-distr1*)

**definition** (**in** *quantale1*) *qstar1* = *q1q.qstar*

**lemma** (**in** *quantale1*) *qstar1-unfold*:  $qstar1 x = (\sqcup i. \text{mm1.power } x \ i)$   
**by** (*simp* *add*: *local.q1q.qstar-def* *local.qstar1-def*)

**sublocale** *quantale1*  $\subseteq$  *dq1s1*: *dioid1* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_1$ )  $1_1$   
**by** *unfold-locales* (*simp-all* *add*: *local.q1q.wswq.distrib-left*)

**sublocale** *quantale1*  $\subseteq$  *dq0ka1*: *kleene-algebra1* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_1$ )  $1_1$  *qstar1*  
**by** *unfold-locales* (*simp-all* *add*: *local.qstar1-def* *local.q1q.uwqlka.star-inductl* *local.q1q.uqka.star-inductr'*)

**class** *domain-quantale1* = *quantale1* + *dom1-op* +  
**assumes** *dom1-absorb*:  $x \leq \text{dom}_1 x \cdot_1 x$   
**and** *dom1-local*:  $\text{dom}_1 (x \cdot_1 \text{dom}_1 y) = \text{dom}_1 (x \cdot_1 y)$   
**and** *dom1-add*:  $\text{dom}_1 (x \sqcup y) = \text{dom}_1 x \sqcup \text{dom}_1 y$   
**and** *dom1-subid*:  $\text{dom}_1 x \leq 1_1$   
**and** *dom1-zero*:  $\text{dom}_1 \perp = \perp$

**sublocale** *domain-quantale1*  $\subseteq$  *dq1dq*: *domain-quantale*  $\text{dom}_1$   $1_1$  ( $\cdot_1$ ) - - - - -

**by** (*unfold-locales, simp-all add: local.dom1-absorb local.dom1-local local.dom1-add local.dom1-subid dom1-zero*)

**sublocale** *domain-quantale1*  $\subseteq$  *dq1ds1*: *domain-semiring1* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_1$ )  $1_1$   
*dom<sub>1</sub>*

**by** (*unfold-locales, simp-all add: local.dom1-local local.dom1-add local.dom1-subid dom1-zero*)

**class** *codomain-quantale1* = *quantale1* + *cod1-op* +  
**assumes** *cod1-absorb*:  $x \leq x \cdot_1 \text{cod}_1 x$   
**and** *cod1-local*:  $\text{cod}_1 (\text{cod}_1 x \cdot_1 y) = \text{cod}_1 (x \cdot_1 y)$   
**and** *cod1-add*:  $\text{cod}_1 (x \sqcup y) = \text{cod}_1 x \sqcup \text{cod}_1 y$   
**and** *cod1-subid*:  $\text{cod}_1 x \leq 1_1$   
**and** *cod1-zero*:  $\text{cod}_1 \perp = \perp$

**sublocale** *codomain-quantale1*  $\subseteq$  *cdq1cdq*: *codomain-quantale*  $1_1$  ( $\cdot_1$ ) - - - - -  
*cod<sub>1</sub>*

**by** (*unfold-locales, simp-all add: local.cod1-absorb local.cod1-local local.cod1-add local.cod1-subid cod1-zero*)

**sublocale** *codomain-quantale1*  $\subseteq$  *cdq1dcs1*: *codomain-semiring1* *cod<sub>1</sub>* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  
 $\perp$  ( $\cdot_1$ )  $1_1$

**by** (*unfold-locales, simp-all add: local.cod1-absorb local.cod1-local local.cod1-add local.cod1-subid cod1-zero*)

**class** *modal-quantale1* = *domain-quantale1* + *codomain-quantale1* +  
**assumes** *dc-compat*:  $\text{dom}_1 (\text{cod}_1 x) = \text{cod}_1 x$   
**and** *cd-compat*:  $\text{cod}_1 (\text{dom}_1 x) = \text{dom}_1 x$

**sublocale** *modal-quantale1*  $\subseteq$  *mq1mq*: *dc-modal-quantale*  $1_1$  ( $\cdot_1$ ) - - - - - *cod<sub>1</sub>*  
*dom<sub>1</sub>*

**by** (*unfold-locales, simp-all add: local.dc-compat local.cd-compat*)

**sublocale** *modal-quantale1*  $\subseteq$  *mq1mka*: *modal-kleene-algebra1* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  ( $\cdot_1$ )  
 $1_1$  *qstar1* *cod<sub>1</sub>* *dom<sub>1</sub>*

**by** *unfold-locales simp-all*

**sublocale** *modal-quantale1*  $\subseteq$  *mq1dual*: *modal-quantale1* *dom<sub>1</sub>* - - - - -  $\lambda x y.$   
 $y \cdot_1 x - \text{cod}_1$

**by** *unfold-locales (simp-all add: local.Sup-distr1 local.Sup-distl1)*

**class** *two-quantale* = *modal-quantale0* + *modal-quantale1* +  
**assumes** *interchange*:  $(w \cdot_1 x) \cdot_0 (y \cdot_1 z) \leq (w \cdot_0 y) \cdot_1 (x \cdot_0 z)$   
**and** *d1-hom*:  $\text{dom}_1 (x \cdot_0 y) \leq \text{dom}_1 x \cdot_0 \text{dom}_1 y$   
**and** *c1-hom*:  $\text{cod}_1 (x \cdot_0 y) \leq \text{cod}_1 x \cdot_0 \text{cod}_1 y$   
**and** *d0-weak-hom*:  $\text{dom}_0 (x \cdot_1 y) \leq \text{dom}_0 x \cdot_1 \text{dom}_0 y$   
**and** *c0-weak-hom*:  $\text{cod}_0 (x \cdot_1 y) \leq \text{cod}_0 x \cdot_1 \text{cod}_0 y$   
**and** *d1d0 [simp]*:  $\text{dom}_1 (\text{dom}_0 x) = \text{dom}_0 x$

```

class strong-two-quantale = two-quantale +
  assumes d1-strong-hom [simp]: dom1 (x ·0 y) = dom1 x ·0 dom1 y
  and c1-strong-hom [simp]: cod1 (x ·0 y) = cod1 x ·0 cod1 y

sublocale two-quantale ⊆ tgqs: two-semiring cod0 (⊔) (≤) (<) ⊥ (·0) 10 dom0
cod1 (·1) 11 dom1
  by (unfold-locales, simp-all add: local.mq0mq.mqs.msrdual.cd-compat local.mq0mq.mqs.msrdual.dc-compat
local.dc-compat local.cd-compat local.interchange local.c0-weak-hom local.c1-hom lo-
cal.d0-weak-hom local.d1-hom)

sublocale strong-two-quantale ⊆ stgqs: strong-two-semiring cod0 (⊔) (≤) (<) ⊥
(·0) 10 dom0 cod1 (·1) 11 dom1
  by unfold-locales simp-all

sublocale two-quantale ⊆ tgqs: two-kleene-algebra (⊔) (≤) (<) ⊥ (·0) 10 qstar0
(·1) 11 qstar1 cod0 dom0 cod1 dom1 ..

sublocale strong-two-quantale ⊆ tgqs: strong-two-kleene-algebra (⊔) (≤) (<) ⊥
(·0) 10 qstar0 (·1) 11 qstar1 cod0 dom0 cod1 dom1 ..

lemma (in strong-two-quantale) id0-le-id1: 10 = 11

  oops

context two-quantale
begin

lemma qstar0-aux: mm0.power (x ·1 y) i ≤ mm0.power x i ·1 mm0.power y i
proof (induct i)
  case 0
  then show ?case by simp
next
  case (Suc i)
  fix i
  assume h: mm0.power (x ·1 y) i ≤ mm0.power x i ·1 mm0.power y i
  have mm0.power (x ·1 y) (Suc i) = (x ·1 y) ·0 mm0.power (x ·1 y) i
    by simp
  also have ... ≤ (x ·1 y) ·0 (mm0.power x i ·1 mm0.power y i)
    by (simp add: h local.q0q.psrpq.mult-isol)
  also have ... ≤ (x ·0 mm0.power x i) ·1 (y ·0 mm0.power y i)
    by (simp add: local.interchange)
  also have ... = mm0.power x (Suc i) ·1 mm0.power y (Suc i)
    by simp
  finally show mm0.power (x ·1 y) (Suc i) ≤ mm0.power x (Suc i) ·1 mm0.power
y (Suc i).
qed

lemma qstar0-oplax: qstar0 (x ·1 y) ≤ qstar0 x ·1 qstar0 y
  by (simp add: local.tgqs.star0-comp1)

```

**lemma** *qstar1-distl0*:  $x \cdot_0 (qstar1\ y) = (\bigsqcup i. x \cdot_0 mm1.power\ y\ i)$   
**by** (*simp add: image-image local.SUP-distl0 local.qstar1-unfold*)

**lemma** *qstar1-distr0*:  $(qstar1\ x) \cdot_0 y = (\bigsqcup i. mm1.power\ x\ i \cdot_0 y)$   
**by** (*simp add: image-image local.SUP-distr0 local.qstar1-unfold*)

**lemma** *qstar0-distl1*:  $x \cdot_1 (qstar0\ y) = (\bigsqcup i. x \cdot_1 mm0.power\ y\ i)$   
**by** (*simp add: image-image local.SUP-distl1 local.qstar0-unfold*)

**lemma** *qstar0-distr1*:  $(qstar0\ x) \cdot_1 y = (\bigsqcup i. mm0.power\ x\ i \cdot_1 y)$   
**by** (*smt (verit, best) image-image local.SUP-cong local.SUP-distr1 local.qstar0-unfold*)

**lemma** *star1-laxl-aux-var*:  $dom_0\ x \cdot_0 mm1.power\ y\ i \leq mm1.power\ (dom_0\ x \cdot_0 y)\ i$   
*i*

**proof** (*induct i*)

**case** *0*  
**have**  $dom_0\ x \cdot_0 1_1 = dom_1\ (dom_0\ x) \cdot_0 1_1$   
**by** *simp*  
**also have**  $\dots \leq 1_1 \cdot_0 1_1$   
**using** *local.dom1-subid local.q0q.nsrnq.mult-isor* **by** *blast*  
**finally have**  $dom_0\ x \cdot_0 1_1 \leq 1_1$   
**by** (*simp add: local.dq0dq.dqmsr.dom-subid-aux2*)  
**thus**  $dom_0\ x \cdot_0 mm1.power\ y\ 0 \leq mm1.power\ (dom_0\ x \cdot_0 y)\ 0$   
**by** *simp*

**next**

**case** (*Suc i*)  
**fix** *i*  
**assume** *h*:  $dom_0\ x \cdot_0 mm1.power\ y\ i \leq mm1.power\ (dom_0\ x \cdot_0 y)\ i$   
**have**  $dom_0\ x \cdot_0 mm1.power\ y\ (Suc\ i) = dom_0\ x \cdot_0 (y \cdot_1 mm1.power\ y\ i)$   
**by** *simp*  
**also have**  $\dots = (dom_0\ x \cdot_1 dom_0\ x) \cdot_0 (y \cdot_1 mm1.power\ y\ i)$   
**by** *simp*  
**also have**  $\dots \leq (dom_0\ x \cdot_0 y) \cdot_1 (dom_0\ x \cdot_0 mm1.power\ y\ i)$   
**using** *local.interchange* **by** *blast*  
**also have**  $\dots \leq (dom_0\ x \cdot_0 y) \cdot_1 mm1.power\ (dom_0\ x \cdot_0 y)\ i$   
**by** (*simp add: h local.q1q.psrpq.mult-isol*)  
**finally show**  $dom_0\ x \cdot_0 mm1.power\ y\ (Suc\ i) \leq mm1.power\ (dom_0\ x \cdot_0 y)\ (Suc\ i)$   
*i*  
**by** *simp*

**qed**

**lemma** *star1-laxl-var*:  $dom_0\ x \cdot_0 qstar1\ y \leq qstar1\ (dom_0\ x \cdot_0 y)$

**proof** –

**have**  $dom_0\ x \cdot_0 qstar1\ y = (\bigsqcup i. dom_0\ x \cdot_0 mm1.power\ y\ i)$   
**using** *local.qstar1-distl0* **by** *auto*  
**also have**  $\dots \leq (\bigsqcup i. mm1.power\ (dom_0\ x \cdot_0 y)\ i)$   
**by** (*simp add: local.SUP-mono' local.star1-laxl-aux-var*)  
**finally show** *?thesis*

by (simp add: local.qstar1-unfold)  
qed

**lemma** star1-laxr-aux-var:  $mm1.power\ x\ i \cdot_0\ cod_0\ y \leq mm1.power\ (x \cdot_0\ cod_0\ y)\ i$

**proof** (induct i)

case 0 show ?case

by (simp add: local.cdq0cdq.coddual.dqmsr.dom-subid-aux2)

next

case (Suc i)

fix i

assume h:  $mm1.power\ x\ i \cdot_0\ cod_0\ y \leq mm1.power\ (x \cdot_0\ cod_0\ y)\ i$

have  $mm1.power\ x\ (Suc\ i) \cdot_0\ cod_0\ y = (x \cdot_1\ mm1.power\ x\ i) \cdot_0\ (cod_0\ y \cdot_1\ cod_0\ y)$

by simp

also have  $\dots \leq (x \cdot_0\ cod_0\ y) \cdot_1\ (mm1.power\ x\ i \cdot_0\ cod_0\ y)$

by (simp add: local.tgqs.tgsdual.d0-comp1)

finally show  $mm1.power\ x\ (Suc\ i) \cdot_0\ cod_0\ y \leq mm1.power\ (x \cdot_0\ cod_0\ y)\ (Suc\ i)$

by (simp add: h local.q0q.h-w2 local.q1q.psrpq.mult-isol)

qed

**lemma** qstar1-laxr-var:  $qstar1\ x \cdot_0\ cod_0\ y \leq qstar1\ (x \cdot_0\ cod_0\ y)$

**proof**–

have  $qstar1\ x \cdot_0\ cod_0\ y = (\bigsqcup i. mm1.power\ x\ i \cdot_0\ cod_0\ y)$

using local.qstar1-distr0 by auto

also have  $\dots \leq (\bigsqcup i. mm1.power\ (x \cdot_0\ cod_0\ y)\ i)$

by (simp add: local.SUP-mono' local.star1-laxr-aux-var)

finally show ?thesis

by (simp add: local.qstar1-unfold)

qed

**lemma** qstar1-power:  $qstar1\ x \cdot_0\ qstar1\ y = (\bigsqcup i\ j. mm1.power\ x\ i \cdot_0\ mm1.power\ y\ j)$

**proof**–

have  $qstar1\ x \cdot_0\ qstar1\ y = qstar1\ x \cdot_0\ (\bigsqcup j. mm1.power\ y\ j)$

using bot-nat-0.extremum local.qstar1-unfold by presburger

also have  $\dots = (\bigsqcup j. qstar1\ x \cdot_0\ mm1.power\ y\ j)$

using calculation local.qstar1-distl0 by auto

also have  $\dots = (\bigsqcup j. (\bigsqcup i. mm1.power\ x\ i) \cdot_0\ mm1.power\ y\ j)$

unfolding qstar1-def q1q.qstar-def by (simp add: full-SetCompr-eq)

also have  $\dots = (\bigsqcup i\ j. mm1.power\ x\ i \cdot_0\ mm1.power\ y\ j)$

by (smt (verit, ccfv-SIG) Sup.SUP-cong calculation local.qstar1-distl0 local.qstar1-distr0)

finally show ?thesis.

qed

end

**context** strong-two-quantale

**begin**

**lemma** *star1-laxl-aux*:  $\text{dom}_1 x \cdot_0 \text{mm1.power } y \ i \leq \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ i$   
**proof** (*induct i*)  
  **case** 0  
  **have**  $\text{dom}_1 x \cdot_0 1_1 \leq 1_1 \cdot_0 1_1$   
  **using** *local.dom1-subid local.q0q.nsrnq.mult-isor* **by** *blast*  
  **thus**  $\text{dom}_1 x \cdot_0 \text{mm1.power } y \ 0 \leq \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ 0$   
  **by** *simp*  
**next**  
  **case** (*Suc i*)  
  **fix** *i*  
  **assume** *h*:  $\text{dom}_1 x \cdot_0 \text{mm1.power } y \ i \leq \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ i$   
  **have**  $\text{dom}_1 x \cdot_0 \text{mm1.power } y \ (\text{Suc } i) = \text{dom}_1 x \cdot_0 (y \cdot_1 \text{mm1.power } y \ i)$   
  **by** *simp*  
  **also have**  $\dots = (\text{dom}_1 x \cdot_1 \text{dom}_1 x) \cdot_0 (y \cdot_1 \text{mm1.power } y \ i)$   
  **by** *simp*  
  **also have**  $\dots \leq (\text{dom}_1 x \cdot_0 y) \cdot_1 (\text{dom}_1 x \cdot_0 \text{mm1.power } y \ i)$   
  **using** *local.interchange* **by** *blast*  
  **also have**  $\dots \leq (\text{dom}_1 x \cdot_0 y) \cdot_1 \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ i$   
  **by** (*simp add: h local.q1q.psrpq.mult-isol*)  
  **finally show**  $\text{dom}_1 x \cdot_0 \text{mm1.power } y \ (\text{Suc } i) \leq \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ (\text{Suc } i)$   
  **by** *simp*  
**qed**

**lemma** *star1-laxl*:  $\text{dom}_1 x \cdot_0 \text{qstar1 } y \leq \text{qstar1 } (\text{dom}_1 x \cdot_0 y)$   
**proof**–  
  **have**  $\text{dom}_1 x \cdot_0 \text{qstar1 } y = (\bigsqcup i. \text{dom}_1 x \cdot_0 \text{mm1.power } y \ i)$   
  **using** *local.qstar1-distl0* **by** *auto*  
  **also have**  $\dots \leq (\bigsqcup i. \text{mm1.power } (\text{dom}_1 x \cdot_0 y) \ i)$   
  **by** (*simp add: local.SUP-mono' local.star1-laxl-aux*)  
  **finally show** *?thesis*  
  **by** (*simp add: local.qstar1-unfold*)  
**qed**

**lemma** *star1-laxr-aux*:  $\text{mm1.power } x \ i \cdot_0 \text{cod}_1 y \leq \text{mm1.power } (x \cdot_0 \text{cod}_1 y) \ i$   
**apply** (*induct i*)  
  **apply** (*metis local.cod1-subid local.mm1.power.power-0 local.q0q.psrpq.mult-isol local.stgqs.stgsdual.id1-comp0-eq*)  
  **by** (*smt (verit, ccfv-SIG) local.dual-order.trans local.q1q.psrpq.mult-isol local.tgqs.tgsdual.d1-comp1 power.power.power-Suc*)

**lemma** *qstar1-laxr*:  $\text{qstar1 } x \cdot_0 \text{cod}_1 y \leq \text{qstar1 } (x \cdot_0 \text{cod}_1 y)$   
**proof**–  
  **have**  $\text{qstar1 } x \cdot_0 \text{cod}_1 y = (\bigsqcup i. \text{mm1.power } x \ i \cdot_0 \text{cod}_1 y)$   
  **using** *local.qstar1-distr0* **by** *auto*  
  **also have**  $\dots \leq (\bigsqcup i. \text{mm1.power } (x \cdot_0 \text{cod}_1 y) \ i)$   
  **by** (*simp add: local.SUP-mono' local.star1-laxr-aux*)  
  **finally show** *?thesis*

by (*simp add: local.qstar1-unfold*)  
**qed**

**lemma** *qstar1-aux*:  $mm1.power\ x\ i \cdot_0\ mm1.power\ y\ i \leq mm1.power\ (x \cdot_0\ y)\ i$   
**proof** (*induct i*)  
 case 0  
 then show ?case  
 by *simp*  
**next**  
 case (*Suc i*)  
 fix *i*  
 assume *h*:  $mm1.power\ x\ i \cdot_0\ mm1.power\ y\ i \leq mm1.power\ (x \cdot_0\ y)\ i$   
 have  $mm1.power\ x\ (Suc\ i) \cdot_0\ mm1.power\ y\ (Suc\ i) = (x \cdot_1\ mm1.power\ x\ i) \cdot_0\ (y \cdot_1\ mm1.power\ y\ i)$   
 by *simp*  
 also have  $\dots \leq (x \cdot_0\ y) \cdot_1\ (mm1.power\ x\ i \cdot_0\ mm1.power\ y\ i)$   
 using *local.interchange* **by force**  
 also have  $\dots \leq (x \cdot_0\ y) \cdot_1\ mm1.power\ (x \cdot_0\ y)\ i$   
 by (*simp add: h local.q1q.psrpq.mult-isol*)  
 also have  $\dots = mm1.power\ (x \cdot_0\ y)\ (Suc\ i)$   
 by *simp*  
 finally show  $mm1.power\ x\ (Suc\ i) \cdot_0\ mm1.power\ y\ (Suc\ i) \leq mm1.power\ (x \cdot_0\ y)\ (Suc\ i)$ .  
**qed**

**lemma** *qstar1*  $x \cdot_0\ qstar1\ y \leq qstar0\ (x \cdot_1\ y)$

**oops**

**lemma** *qstar1*  $x \cdot_0\ qstar1\ y \leq qstar1\ (x \cdot_1\ y)$

**oops**

**lemma** *qstar1*  $(x \cdot_1\ y) \leq qstar1\ x \cdot_0\ qstar1\ y$

**oops**

**lemma** *qstar1*  $x \cdot_0\ qstar1\ y \leq qstar1\ (x \cdot_0\ y)$

**oops**

**lemma** *qstar1*  $(x \cdot_0\ y) \leq qstar1\ x \cdot_0\ qstar1\ y$

**oops**

**lemma** *qstar0*  $x \cdot_1\ qstar0\ y \leq qstar0\ (x \cdot_0\ y)$

**oops**

```

end

lemma (in strong-two-kleene-algebra) qstar0 x ·1 qstar0 y ≤ qstar0 (x ·1 y)

oops

lemma (in strong-two-kleene-algebra) qstar0 (x ·1 y) ≤ qstar0 x ·1 qstar0 y

oops

class two-quantale-lmcs = modal-quantale0 + modal-quantale1 +
  assumes interchange: (w ·1 x) ·0 (y ·1 z) ≤ (w ·0 y) ·1 (x ·0 z)
  and d1-hom: dom1 (x ·0 y) = dom1 x ·0 dom1 y
  and c1-hom: cod1 (x ·0 y) = cod1 x ·0 cod1 y
  and d1d0 [simp]: dom1 (dom0 x) = dom0 x
  and c1d0 [simp]: cod1 (dom0 x) = dom0 x
  and d1c0 [simp]: dom1 (cod0 x) = cod0 x
  and c1c0 [simp]: cod1 (cod0 x) = cod0 x
  and d0d1 [simp]: dom0 (dom1 x) = dom0 x
  and c0d1 [simp]: cod0 (dom1 x) = dom0 x
  and d0c1 [simp]: dom0 (cod1 x) = cod0 x
  and c0c1 [simp]: cod0 (cod1 x) = cod0 x

begin

lemma dom0 (x ·1 y) ≤ dom0 x ·1 dom0 y

oops

lemma cod0 (x ·1 y) ≤ cod0 x ·1 cod0 y

oops

end

end

```

## 5 Lifting 2-Catoids to powerset 2-quantales

```

theory Two-Catoid-Lifting
  imports Two-Catoid Two-Quantale Catoids.Catoid-Lifting

begin

instantiation set :: (local-two-catoid) two-quantale

begin

definition dom0-set :: 'a set ⇒ 'a set where

```



$dom_0 X = Src_0 X$

**definition**  $cod_0\text{-set} :: 'a \text{ set} \Rightarrow 'a \text{ set}$  **where**  
 $cod_0 X = Tgt_0 X$

**definition**  $comp0\text{-set} :: 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set}$  **where**  
 $X \cdot_0 Y = X *_0 Y$

**definition**  $id0\text{-set} :: 'a \text{ set}$   
**where**  $1_0 = s0fix$

**definition**  $dom_1\text{-set} :: 'a \text{ set} \Rightarrow 'a \text{ set}$  **where**  
 $dom_1 X = Src_1 X$

**definition**  $cod_1\text{-set} :: 'a \text{ set} \Rightarrow 'a \text{ set}$  **where**  
 $cod_1 X = Tgt_1 X$

**definition**  $comp1\text{-set} :: 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set}$  **where**  
 $X \cdot_1 Y = X *_1 Y$

**definition**  $id1\text{-set} :: 'a \text{ set}$  **where**  
 $1_1 = t1fix$

**instance**

**apply** *intro-classes*

**unfolding**  $comp0\text{-set-def}$   $dom_0\text{-set-def}$   $cod_0\text{-set-def}$   $id0\text{-set-def}$   $comp1\text{-set-def}$   $dom_1\text{-set-def}$   
 $cod_1\text{-set-def}$   $id1\text{-set-def}$

**apply** (*simp add: msg0.conv-assoc*)

**using**  $stmm0.stopp.stopp.conv-uns$  **apply** *blast*

**apply** (*metis stmm0.stopp.stopp.conv-unt stmm0.stopp.stopp.st-fix-set*)

**apply** (*smt (verit, ccfv-SIG) Collect-cong image-def*

$stmm0.stopp.conv-distr$ )

**apply** (*smt (z3) Collect-cong image-def multimagma.conv-distr*)

**apply** *simp+*

**apply** (*simp add: image-Un*)

**using**  $stmm0.stopp.stopp.Tgt-subid$   $stmm0.stopp.stopp.st-fix-set$  **apply** *blast*

**apply** *force*

**apply** *simp+*

**apply** (*simp add: image-Un*)

**using**  $stmm0.stopp.stopp.Src-subid$  **apply** *blast*

**apply** *force*

**apply** *simp+*

**apply** (*simp add: msg1.conv-assoc*)

**apply** (*metis stmm1.stopp.stopp.conv-uns stmm1.stopp.stopp.st-fix-set*)

**using**  $stmm1.stopp.stopp.conv-unt$  **apply** *blast*

**apply** (*smt (verit, best) Collect-cong image-def multimagma.conv-distl*)

**apply** (*smt (verit) Collect-cong image-def multimagma.conv-distr*)

**apply** *simp+*

**apply** (*simp add: image-Un*)

```

using stmm1.stopp.stopp.Tgt-subid apply blast
  apply simp+
  apply (simp add: image-Un)
  apply force
  apply simp+
  apply (simp add: interchange-lifting)
  apply (simp add: Src1-hom)
  apply (simp add: Tgt1-hom)
using Src0-hom apply blast
  apply (simp add: Tgt0-hom)
by simp

end

end

```

## 6 2-Catoids with (too) strong homomorphisms

```

theory Two-Catoid-Collapse
  imports Two-Catoid

```

```
begin
```

Here we present variants of 2-categories where the axioms are too strong. There is an Eckmann-Hilton style collapse of the two structures.

### 6.1 2-st-Multimagmas with strong homomorphism laws

```

class two-st-multimagma-collapse = st-multimagma0 + st-multimagma1 +
  assumes comm-s0s1:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$ 
  and comm-t0t1:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$ 
  and comm-s0t1:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$ 
  and comm-tr0s1:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$ 
  assumes interchange:  $(w \odot_1 x) *_0 (y \odot_1 z) \subseteq (w \odot_0 y) *_1 (x \odot_0 z)$ 
  and t0-hom:  $Tgt_0 (x \odot_1 y) = \tau_0 x \odot_1 \tau_0 y$ 
  and t1-hom:  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$ 
  and s0-hom:  $Src_0 (x \odot_1 y) = \sigma_0 x \odot_1 \sigma_0 y$ 
  and s1-hom:  $Src_1 (x \odot_0 y) = \sigma_1 x \odot_0 \sigma_1 y$ 
  and s1-s0 [simp]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$ 
  and t1-s0 [simp]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$ 
  and s1-t0 [simp]:  $\sigma_1 (\tau_0 x) = \tau_0 x$ 
  and t1-t0 [simp]:  $\tau_1 (\tau_0 x) = \tau_0 x$ 

```

```
begin
```

The source and target structure collapses.

```

lemma s0s1:  $\sigma_0 x = \sigma_1 x$ 
proof -

```

```

have Src0 ( $\sigma_1 x \odot_1 \sigma_0 (\sigma_1 x)$ ) =  $\sigma_0 (\sigma_1 x) \odot_1 \sigma_0 (\sigma_0 (\sigma_1 x))$ 
  by (simp add: local.s0-hom)
also have ... =  $\sigma_1 (\sigma_0 (\sigma_1 x)) \odot_1 \sigma_0 (\sigma_1 x)$ 
  by simp
also have ... =  $\{\sigma_0 (\sigma_1 x)\}$ 
  using local.src1-absorb by presburger
also have ...  $\neq \{\}$ 
  by (metis insert-not-empty)
finally have Src0 ( $\sigma_1 x \odot_1 \sigma_0 (\sigma_1 x)$ )  $\neq \{\}$ .
hence  $\sigma_1 x \odot_1 \sigma_0 (\sigma_1 x) \neq \{\}$ 
  by (metis image-empty)
hence  $\tau_1 (\sigma_1 x) = \sigma_1 (\sigma_0 (\sigma_1 x))$ 
  using local.Dst1 by presburger
hence  $\sigma_1 x = \sigma_1 (\sigma_0 (\sigma_1 x))$ 
  by simp
also have ... =  $\sigma_1 (\sigma_1 (\sigma_0 x))$ 
  by (simp add: local.comm-s0s1)
also have ... =  $\sigma_1 (\sigma_0 x)$ 
  by simp
also have ... =  $\sigma_0 x$ 
  by simp
finally show ?thesis..
qed

lemma t0t1:  $\tau_0 x = \tau_1 x$ 
  using local.comm-s0t1 local.commtr0s1 local.s0s1 stmm0.ts-compat by force

lemma s0t0:  $\sigma_0 x = \tau_0 x$ 
proof-
  have  $\sigma_0 x = \tau_1 (\sigma_0 x)$ 
    by simp
  also have ... =  $\sigma_0 (\tau_1 x)$ 
    by (simp add: local.comm-s0t1)
  also have ... =  $\sigma_0 (\tau_0 x)$ 
    by (simp add: t0t1)
  also have ... =  $\tau_0 x$ 
    by simp
  finally show ?thesis.
qed

lemma  $\sigma_0 x = x$ 

oops

lemma stt1:  $\sigma_1 x = \tau_1 x$ 
  using local.s0s1 s0t0 t0t1 by force

lemma  $x \in y \odot_0 z \implies x' \in y \odot_0 z \implies x = x'$ 

```

**oops**

**lemma**  $x \in y \odot_1 z \implies x' \in y \odot_1 z \implies x = x'$

**oops**

The two compositions are still different—but see below for 2-catoids.

**end**

## 6.2 2-Catoids with (too) strong homomorphisms

**class** *two-catoid-collapse* = *two-st-multimagma-collapse* + *catoid0* + *catoid1*

**begin**

The two compositions are still different, and neither of them commutes.

**lemma**  $x \odot_0 y = x \odot_1 y$

**oops**

**lemma**  $x \odot_0 y = y \odot_0 x$

**oops**

**lemma**  $x \odot_1 y = y \odot_1 x$

**oops**

**end**

## 6.3 Single-set 2-categories with (too) strong homomorphisms

**class** *two-category-collapse* = *two-catoid-collapse* + *single-set-category0* + *single-set-category1*

**begin**

**lemma** *comp-collapse*:  $x \odot_0 y = x \odot_1 y$

**by** (*smt* (*verit*, *del-insts*) *local.interchange local.mm0.conv-atom local.mm1.conv-atom local.pm1.functionality-lem-var local.s0s1 local.src0-absorb local.src1-absorb local.stmsg1.sts-msg.st-local local.t0t1 local.tgt0-absorb local.tgt1-absorb ssmsg0.st-local subset-singleton-iff*)

**lemma** *comp0-comm*:  $x \odot_0 y = y \odot_0 x$

**by** (*smt* (*verit*, *best*) *bot-set-def comp-collapse doubleton-eq-iff local.interchange local.mm0.conv-atom local.mm1.conv-atom local.pm0.functionality-lem-var local.s0t0 local.src0-absorb local.tgt0-absorb singleton-insert-inj-eq' ssmsg0.st-local*)

**lemma** *comp1-comm*:  $x \odot_1 y = y \odot_1 x$

**using** *comp0-comm comp-collapse* **by** *auto*

**lemma**  $\sigma_0 x = x$

**oops**

**lemma**  $\sigma_0 x = \sigma_0 y$

**oops**

**lemma**  $x \odot_0 y \neq \{\}$

**oops**

**lemma**  $x \odot_1 y \neq \{\}$

**oops**

**end**

## 6.4 2-lr-Multimagmas with strong interchange law

```
class two-lr-multimagma-bad = st-multimagma0 + st-multimagma1 +  
  assumes comm-s0s1:  $\sigma_0 (\sigma_1 x) = \sigma_1 (\sigma_0 x)$   
  and comm-t0t1:  $\tau_0 (\tau_1 x) = \tau_1 (\tau_0 x)$   
  and comm-s0t1:  $\sigma_0 (\tau_1 x) = \tau_1 (\sigma_0 x)$   
  and comm-t0s1:  $\tau_0 (\sigma_1 x) = \sigma_1 (\tau_0 x)$   
  assumes interchange:  $(w \odot_1 x) *_0 (y \odot_1 z) = (w \odot_0 y) *_1 (x \odot_0 z)$   
  and t0-hom:  $Tgt_0 (x \odot_1 y) = \tau_0 x \odot_1 \tau_0 y$   
  and t1-hom:  $Tgt_1 (x \odot_0 y) = \tau_1 x \odot_0 \tau_1 y$   
  and s0-hom:  $Src_0 (x \odot_1 y) \subseteq \sigma_0 x \odot_1 \sigma_0 y$   
  and s1-hom:  $Src_1 (x \odot_0 y) \subseteq \sigma_1 x \odot_0 \sigma_1 y$   
  and s1-s0 [simp]:  $\sigma_1 (\sigma_0 x) = \sigma_0 x$   
  and t1-s0 [simp]:  $\tau_1 (\sigma_0 x) = \sigma_0 x$   
  and s1-t0 [simp]:  $\sigma_1 (\tau_0 x) = \tau_0 x$   
  and t1-t0 [simp]:  $\tau_1 (\tau_0 x) = \tau_0 x$ 
```

**begin**

The source and target structure collapses.

**lemma** *s0s1*:  $\sigma_0 x = \sigma_1 x$

**by** (*metis image-empty local.comm-s0s1 local.s1-s0 local.stmm0.stopp.st-fix local.stmm0.stopp.ts-compat local.t0-hom stmm1.s-absorb-var3*)

**lemma** *t0t1*:  $\tau_0 x = \tau_1 x$

**using** *local.comm-s0t1 local.comm-t0s1 local.s0s1 stmm0.ts-compat* **by** *auto*

**lemma** *s0t0*:  $\sigma_0 x = \tau_0 x$

**proof** –

**have**  $\sigma_0 x = \tau_1 (\sigma_0 x)$

by *simp*  
 also have  $\dots = \sigma_0 (\tau_1 x)$   
 by (*simp add: local.comm-s0t1*)  
 also have  $\dots = \sigma_0 (\tau_0 x)$   
 by (*simp add: local.t0t1*)  
 also have  $\dots = \tau_0 x$   
 by *simp*  
 finally show *?thesis*.  
 qed

**lemma** *s1t1*:  $\sigma_1 x = \tau_1 x$   
 using *local.comm-t0s1 local.t0t1* by *force*

**lemma** *comp-collapse*:  $x \odot_0 y = x \odot_1 y$   
 by (*metis (no-types, opaque-lifting) local.interchange local.s0s1 local.src0-absorb local.src1-absorb local.stmm0.stopp.Dst local.stmm1.stopp.Dst local.t0t1 local.tgt0-absorb local.tgt1-absorb multimagma.conv-atom*)

**lemma** *comp0-comm*:  $x \odot_0 y = y \odot_0 x$   
 by (*metis local.comp-collapse local.interchange local.mm0.conv-atom local.s0s1 local.s1t1 local.src0-absorb local.tgt1-absorb stmm1.t-comm*)

**lemma** *comp1-comm*:  $x \odot_1 y = y \odot_1 x$   
 using *local.comp0-comm local.comp-collapse* by *auto*

**lemma** *comp0-assoc*:  $\{x\} *_0 (y \odot_0 z) = (x \odot_0 y) *_0 \{z\}$   
 by (*smt (z3) local.comp-collapse local.interchange local.s1t1 local.src1-absorb local.stmm0.stopp.Dst local.stmm0.stopp.s-assoc local.stmm1.stopp.conv-atom local.t0t1 local.t1-t0 local.tgt1-absorb*)

**lemma** *comp1-assoc*:  $\{x\} *_1 (y \odot_1 z) = (x \odot_1 y) *_1 \{z\}$   
 by (*metis (full-types) local.comp0-assoc local.comp1-comm local.comp-collapse local.interchange local.tgt1-absorb*)

**lemma**  $\sigma_0 x = x$

oops

**lemma**  $\sigma_0 x = \sigma_0 y$

oops

**lemma**  $x \odot_0 y \neq \{\}$

oops

**lemma**  $x \in y \odot_0 z \implies x' \in y \odot_0 z \implies x = x'$

oops

```

lemma  $x \odot_1 y \neq \{\}$ 

  oops

lemma  $x \in y \odot_1 z \implies x' \in y \odot_1 z \implies x = x'$ 

  oops

end

end

```

## 7 $\omega$ -Catoids

```

theory Omega-Catoid
  imports Two-Catoid

```

```

begin

```

### 7.1 Indexed catoids.

We add an index to the operations of catoids.

```

class imultimagma =
  fixes incomp :: 'a  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  'a set ( $-\odot -$  [70,70,70]70)

definition (in imultimagma) iconv :: 'a set  $\Rightarrow$  nat  $\Rightarrow$  'a set  $\Rightarrow$  'a set ( $-\star-$  [70,70,70]70)
where
   $X \star_i Y = (\bigcup x \in X. \bigcup y \in Y. x \odot_i y)$ 

class multisemigroup = imultimagma +
  assumes assoc:  $(\bigcup v \in y \odot_i z. x \odot_i v) = (\bigcup v \in x \odot_i y. v \odot_i z)$ 

begin

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

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

lemma iconv-prop:  $X \star_i Y \equiv$  ims.conv i  $X Y$ 
  by (simp add: local.iconv-def multimagma.conv-def)

end

class st-imultimagma = imultimagma +
  fixes src :: nat  $\Rightarrow$  'a  $\Rightarrow$  'a
  and tgt :: nat  $\Rightarrow$  'a  $\Rightarrow$  'a

```

```

assumes Dst:  $x \odot_i y \neq \{\}$   $\implies$   $\text{tgt } i \ x = \text{src } i \ y$ 
and src-absorb [simp]:  $(\text{src } i \ x) \odot_i x = \{x\}$ 
and tgt-absorb [simp]:  $x \odot_i (\text{tgt } i \ x) = \{x\}$ 

begin

lemma inst:  $(\text{src } 1 \ x) \odot_1 x = \{x\}$ 
  by simp

sublocale stimm: st-multimagma  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales (simp-all add: local.Dst)

sublocale stimm0: st-multimagma0  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales (simp-all add: local.Dst)

sublocale stimm1: st-multimagma1  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales (simp-all add: local.Dst)

abbreviation srcfix  $\equiv$  stimm.srcfix

abbreviation tgtfix  $\equiv$  stimm.tgtfix

abbreviation Srci  $\equiv$  stimm.Src

abbreviation Tgti  $\equiv$  stimm.Tgt

end

class icatoid = st-imagma + multisemigroup

sublocale icatoid  $\subseteq$  icat: catoid  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales

class local-icatoid = icatoid +
  assumes src-local:  $\text{Srci } i \ (x \odot_i \text{src } i \ y) \subseteq \text{Srci } i \ (x \odot_i y)$ 
  and tgt-local:  $\text{Tgti } i \ (\text{tgt } i \ x \odot_i y) \subseteq \text{Tgti } i \ (x \odot_i y)$ 

sublocale local-icatoid  $\subseteq$  licat: local-catoid  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales(simp-all add: local.src-local local.tgt-local)

class functional-imagma = imagma +
  assumes functionality:  $x \in y \odot_i z \implies x' \in y \odot_i z \implies x = x'$ 

sublocale functional-imagma  $\subseteq$  pmi: functional-magma  $\lambda x y. x \odot_i y$ 
  by unfold-locales(simp add: local.functionality)

class functional-isemigroup = functional-imagma + multisemigroup

sublocale functional-isemigroup  $\subseteq$  psgi: functional-semigroup  $\lambda x y. x \odot_i y..$ 

```



```

class functional-icatoid = functional-isemigroup + icatoid

sublocale functional-icatoid  $\subseteq$  psgi: functional-catoid  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales

class icategory = functional-icatoid + local-icatoid

sublocale icategory  $\subseteq$  icatt: single-set-category  $\lambda x y. x \odot_i y$  src i tgt i
  by unfold-locales

```

## 7.2 $\omega$ -Catoids

Next we define  $\omega$ -catoids and  $\omega$ -categories.

```

class omega-st-multimagma = st-multimagma +
  assumes comm-sisj:  $i \neq j \implies \text{src } i (\text{src } j x) = \text{src } j (\text{src } i x)$ 
  and comm-sitj:  $i \neq j \implies \text{src } i (\text{tgt } j x) = \text{tgt } j (\text{src } i x)$ 
  and comm-titj:  $i \neq j \implies \text{tgt } i (\text{tgt } j x) = \text{tgt } j (\text{tgt } i x)$ 
  and si-hom:  $i \neq j \implies \text{Src } i (x \odot_j y) \subseteq \text{src } i x \odot_j \text{src } i y$ 
  and ti-hom:  $i \neq j \implies \text{Tgt } i (x \odot_j y) \subseteq \text{tgt } i x \odot_j \text{tgt } i y$ 
  and omega-interchange:  $i < j \implies (w \odot_j x) \star_i (y \odot_j z) \subseteq (w \odot_i y) \star_j (x \odot_i z)$ 
  and sjsi [simp]:  $i < j \implies \text{src } j (\text{src } i x) = \text{src } i x$ 
  and sjti [simp]:  $i < j \implies \text{src } j (\text{tgt } i x) = \text{tgt } i x$ 
  and tjsi [simp]:  $i < j \implies \text{tgt } j (\text{src } i x) = \text{src } i x$ 
  and tjti [simp]:  $i < j \implies \text{tgt } j (\text{tgt } i x) = \text{tgt } i x$ 

```

```

class omega-catoid = omega-st-multimagma + icatoid

```

```

context omega-st-multimagma
begin

```

```

lemma omega-interchange-var:  $(w \odot_{(i+k+1)} x) \star_i (y \odot_{(i+k+1)} z) \subseteq (w \odot_i y) \star_{(i+k+1)} (x \odot_i z)$ 
  using local.omega-interchange by auto

```

```

lemma all-sisj:  $\text{src } i (\text{src } j x) = \text{src } j (\text{src } i x)$ 
  by (metis local.comm-sisj)

```

```

lemma all-titj:  $\text{tgt } i (\text{tgt } j x) = \text{tgt } j (\text{tgt } i x)$ 
  by (metis local.comm-titj)

```

```

lemma sjsi-var [simp]:  $\text{src } (i+k) (\text{src } i x) = \text{src } i x$ 
  by (metis le-add1 local.sjsi local.stimm.stopp.tt-idem nless-le)

```

```

lemma sjti-var [simp]:  $\text{src } (i+k) (\text{tgt } i x) = \text{tgt } i x$ 
  by (metis local.stimm.stopp.ts-compatible sjsi-var)

```

```

lemma tjsi-var [simp]:  $\text{tgt } (i+k) (\text{src } i x) = \text{src } i x$ 

```

**by** (*simp add: stimm.st-fix*)

**lemma** *tjti-var* [*simp*]:  $\text{tgt } (i + k) (\text{tgt } i \ x) = \text{tgt } i \ x$   
**by** (*simp add: stimm.st-fix*)

The following sublocale statement should help us to translate statements for 2-catoids to  $\omega$ -catoids. But it does not seem to work. Hence we duplicate the work done for 2-catoids, and later also for semirings and quantales.

**sublocale** *otmm: two-st-multimagma*

$\lambda x \ y. x \odot_i y$

*src*  $i$

*tgt*  $i$

$\lambda x \ y. x \odot_{(i + k + 1)} y$

*src*  $(i + k + 1)$

*tgt*  $(i + k + 1)$

**apply** *unfold-locales*

**apply** (*simp-all add: comm-sisj comm-sitj comm-titj si-hom ti-hom*)

**using** *less-add-Suc1 local.all-sisj local.sjsi* **apply** *simp*

**apply** (*metis lessI less-add-Suc1 local.comm-sitj local.sjti not-add-less1*)

**apply** (*metis less-add-Suc1 local.comm-titj local.tjti*)

**using** *local.iconv-def local.omega-interchange local.stimm.conv-def* **by** *simp*

**end**

**class** *omega-st-multimagma-strong* = *omega-st-multimagma* +

**assumes** *sj-hom-strong*:  $i < j \implies \text{Src } i \ j \ (x \odot_i y) = \text{src } j \ x \odot_i \ \text{src } j \ y$

**and** *tj-hom-strong*:  $i < j \implies \text{Tgt } i \ j \ (x \odot_i y) = \text{tgt } j \ x \odot_i \ \text{tgt } j \ y$

**begin**

**lemma** *sj-hom-strong-var*:  $\text{Src } i \ (i + k + 1) \ (x \odot_i y) = \text{src } (i + k + 1) \ x \odot_i \ \text{src } (i + k + 1) \ y$

**by** (*simp add: local.sj-hom-strong*)

**lemma** *tj-hom-strong-var*:  $\text{Tgt } i \ (i + k + 1) \ (x \odot_i y) = \text{tgt } (i + k + 1) \ x \odot_i \ \text{tgt } (i + k + 1) \ y$

**by** (*simp add: local.tj-hom-strong*)

**end**

**sublocale** *omega-st-multimagma*  $\subseteq$  *olropp: omega-st-multimagma*  $\lambda x \ i \ y. y \odot_i x$   
*tgt src*

**apply** *unfold-locales*

**apply** (*simp-all add: local.Dst*)

**using** *local.comm-titj* **apply** *simp*

**using** *local.comm-sitj* **apply** *simp*

**using** *local.all-sisj* **apply** *simp*

**apply** (*simp add: local.ti-hom*)

**apply** (*simp add: local.si-hom*)

**unfolding** *imultimagma.iconv-def*  
**using** *local.iconv-def local.omega-interchange local.stimm.conv-def local.stimm.stopp.conv-def*  
**by** *simp*

**context** *omega-st-multimagma*  
**begin**

**lemma** *sisj*:  $i \leq j \implies \text{src } i (\text{src } j x) = \text{src } i x$   
**using** *antisym-conv2 local.all-sisj local.sjsi* **by** *fastforce*

**lemma** *sisj-var* [*simp*]:  $\text{src } i (\text{src } (i + k) x) = \text{src } i x$   
**by** (*simp add: sisj*)

**lemma** *sitj*:  $i < j \implies \text{src } i (\text{tgt } j x) = \text{src } i x$   
**by** (*simp add: local.comm-sitj*)

**lemma** *sitj-var* [*simp*]:  $\text{src } i (\text{tgt } (i + k + 1) x) = \text{src } i x$   
**using** *local.otmm.twolropp.t0s1* **by** *auto*

**lemma** *tisj*:  $i < j \implies \text{tgt } i (\text{src } j x) = \text{tgt } i x$   
**by** (*simp add: local.olropp.comm-sitj*)

**lemma** *tisj-var* [*simp*]:  $\text{tgt } i (\text{src } (i + k + 1) x) = \text{tgt } i x$   
**using** *local.otmm.twolropp.s0t1* **by** *auto*

**lemma** *titi*:  $i \leq j \implies \text{tgt } i (\text{tgt } j x) = \text{tgt } i x$   
**using** *antisym-conv2 local.olropp.all-sisj local.tjti* **by** *fastforce*

**lemma** *titi-var* [*simp*]:  $\text{tgt } i (\text{tgt } (i + k) x) = \text{tgt } i x$   
**by** (*simp add: titi*)

**end**

**context** *omega-catoid*  
**begin**

**lemma** *src-icat1*:  
**assumes**  $i \leq j$   
**and**  $DD j x y$   
**shows**  $\text{Src } i (x \odot_j y) = \{\text{src } i x\}$   
**by** (*smt (verit, ccfv-SIG) assms icat.src-comp-cond image-is-empty local.comm-sitj local.si-hom local.sisj local.stimm.stopp.Dst local.tgt-absorb subset-singletonD*)

**lemma** *src-icat2*:  
**assumes**  $i < j$   
**and**  $DD j x y$   
**shows**  $\text{Src } i (x \odot_j y) = \{\text{src } i y\}$   
**by** (*metis assms less-or-eq-imp-le local.all-sisj local.sitj local.sjsi local.stimm.stopp.Dst src-icat1*)

**lemma** *tgt-icat1*:  
**assumes**  $i < j$   
**and**  $DD\ j\ x\ y$   
**shows**  $Tgt\ i\ (x \odot_j y) = \{tgt\ i\ x\}$   
**by** (*smt* (*verit*) *assms image-is-empty local.olropp.all-sisj local.stimm.stopp.Dst local.ti-hom local.tisj local.tjti not-less-iff-gr-or-eq stimm.t-idem subset-singletonD*)

**lemma** *tgt-icat2*:  
**assumes**  $i \leq j$   
**and**  $DD\ j\ x\ y$   
**shows**  $Tgt\ i\ (x \odot_j y) = \{tgt\ i\ y\}$   
**by** (*smt* (*verit*, *best*) *assms(1) assms(2) icat.tgt-comp-cond local.all-titj local.stimm.stopp.Dst local.tisj local.tjti nat-less-le tgt-icat1*)

**end**

We lift the axioms to the powerset level.

**context** *omega-st-multimagma*  
**begin**

**lemma** *comm-SiSj*:  $Srci\ i\ (Srci\ j\ X) = Srci\ j\ (Srci\ i\ X)$   
**by** (*metis* (*mono-tags*, *lifting*) *image-cong image-image local.comm-sisj*)

**lemma** *comm-TiTj*:  $Tgti\ i\ (Tgti\ j\ X) = Tgti\ j\ (Tgti\ i\ X)$   
**by** (*simp* *add: image-image local.olropp.all-sisj*)

**lemma** *comm-SiTj*:  $i \neq j \implies Srci\ i\ (Tgti\ j\ x) = Tgti\ j\ (Srci\ i\ x)$   
**by** (*metis* (*mono-tags*, *lifting*) *image-cong image-image local.comm-sitj*)

**lemma** *comm-TiSj*:  $i \neq j \implies Tgti\ i\ (Srci\ j\ x) = Srci\ j\ (Tgti\ i\ x)$   
**using** *local.comm-SiTj* **by** *simp*

**lemma** *interchange-lift*:  
**assumes**  $i < j$   
**shows**  $(W \star_j X) \star_i (Y \star_j Z) \subseteq (W \star_i Y) \star_j (X \star_i Z)$   
**proof** –  
  **{fix**  $a$   
  **assume**  $a \in (W \star_j X) \star_i (Y \star_j Z)$   
  **hence**  $\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. a \in (w \odot_j x) \star_i (y \odot_j z)$   
  **unfolding** *iconv-def* **by** *force*  
  **hence**  $\exists w \in W. \exists x \in X. \exists y \in Y. \exists z \in Z. a \in (w \odot_i y) \star_j (x \odot_i z)$   
  **using** *assms local.omega-interchange* **by** *fastforce*  
  **hence**  $a \in (W \star_i Y) \star_j (X \star_i Z)$   
  **unfolding** *iconv-def* **by** *force*  
  **thus** *?thesis..*  
**qed**

**lemma** *Srcj-hom*:

**assumes**  $i \neq j$   
**shows**  $\text{Srci } j (X \star_i Y) \subseteq \text{Srci } j X \star_i \text{Srci } j Y$   
**unfolding** *iconv-def* **by** (*clarsimp*, *metis assms image-subset-iff local.si-hom*)

**lemma** *Tgtj-hom*:  
**assumes**  $i \neq j$   
**shows**  $\text{Tgti } j (X \star_i Y) \subseteq \text{Tgti } j X \star_i \text{Tgti } j Y$   
**unfolding** *iconv-def* **by** (*clarsimp*, *metis assms image-subset-iff local.ti-hom*)

**lemma** *SjSi*:  $i \leq j \implies \text{Srci } j (\text{Srci } i X) = \text{Srci } i X$   
**by** (*smt (verit, best) antisym-conv2 imageE image-cong local.sjsi local.stimm.stopp.ST-compat local.stimm.stopp.ts-compat stimt.ts-compat*)

**lemma** *SjSi-var* [*simp*]:  $\text{Srci } (i + k) (\text{Srci } i X) = \text{Srci } i X$   
**by** (*simp add: image-image*)

**lemma** *SjTi*:  $i \leq j \implies \text{Srci } j (\text{Tgti } i X) = \text{Tgti } i X$   
**by** (*metis SjSi-var less-eqE local.stimm.stopp.TS-compat*)

**lemma** *SjTi-var* [*simp*]:  $\text{Srci } (i + k) (\text{Tgti } i X) = \text{Tgti } i X$   
**by** (*simp add: SjTi*)

**lemma** *TjSi*:  $i \leq j \implies \text{Tgti } j (\text{Srci } i X) = \text{Srci } i X$   
**by** (*metis local.SjSi local.stimm.stopp.ST-compat*)

**lemma** *TjSi-var* [*simp*]:  $\text{Tgti } (i + k) (\text{Srci } i X) = \text{Srci } i X$   
**using** *TjSi le-add1* **by** *presburger*

**lemma** *TjTi*:  $i \leq j \implies \text{Tgti } j (\text{Tgti } i X) = \text{Tgti } i X$   
**by** (*metis local.SjTi local.stimm.stopp.ST-compat*)

**lemma** *TjTi-var* [*simp*]:  $\text{Tgti } (i + k) (\text{Tgti } i X) = \text{Tgti } i X$   
**by** (*simp add: TjTi*)

**lemma** *srcfixij*:  $i \leq j \implies \text{srcfix } i \subseteq \text{srcfix } i \star_j \text{srcfix } i$   
**by** (*smt (verit, ccfv-SIG) UN-iff antisym-conv2 local.iconv-def local.tgt-absorb local.tjt mem-Collect-eq singletonI stimt.ts-compat subsetI*)

**lemma** *srcfixij-var*:  $\text{srcfix } i \subseteq \text{srcfix } i \star_{(j+k)} \text{srcfix } i$   
**by** (*smt (verit, ccfv-SIG) UN-iff local.comm-sitj local.iconv-def local.stimm.stopp.ts-compat local.tgt-absorb mem-Collect-eq singletonI subsetI*)

**lemma** *srcfixij-var2*:  $i \leq j \implies \text{srcfix } i \subseteq \text{srcfix } j$   
**by** (*metis local.SjSi local.stimm.stopp.Tgt-subid local.stimm.stopp.tfix-im*)

**lemma** *srcfixij-var3*:  $\text{srcfix } i \subseteq \text{srcfix } (i + k)$   
**using** *le-add1 srcfixij-var2* **by** *blast*

**end**

**context** *omega-st-multimagma-strong*

**begin**

**lemma** *Srcj-hom-strong*:

**assumes**  $i < j$

**shows**  $\text{Srci } j (X \star_i Y) = \text{Srci } j X \star_i \text{Srci } j Y$

**proof** –

**{fix**  $a$

**have**  $(a \in \text{Srci } j (X \star_i Y)) = (\exists b \in X \star_i Y. a = \text{src } j b)$

**by** *force*

**also have**  $\dots = (\exists b. \exists c \in X. \exists d \in Y. a = \text{src } j b \wedge b \in c \odot_i d)$

**using** *local.iconv-def* **by** *auto*

**also have**  $\dots = (\exists c \in X. \exists d \in Y. a \in \text{Srci } j (c \odot_i d))$

**by** *force*

**also have**  $\dots = (\exists c \in X. \exists d \in Y. a \in \text{src } j c \odot_i \text{src } j d)$

**using** *assms local.sj-hom-strong* **by** *auto*

**also have**  $\dots = (\exists c \in \text{Srci } j X. \exists d \in \text{Srci } j Y. a \in c \odot_i d)$

**by** *force*

**also have**  $\dots = (a \in \text{Srci } j X \star_i \text{Srci } j Y)$

**by** (*simp add: local.iconv-def*)

**finally have**  $(a \in \text{Srci } j (X \star_i Y)) = (a \in \text{Srci } j X \star_i \text{Srci } j Y).$

**thus** *?thesis*

**by** *force*

**qed**

**lemma** *Srcj-hom-strong-var*:  $\text{Srci } (i + k + 1) (X \star_i Y) = \text{Srci } (i + k + 1) X \star_i \text{Srci } (i + k + 1) Y$

**by** (*simp add: Srcj-hom-strong*)

**lemma** *Tgtj-hom-strong*:

**assumes**  $i < j$

**shows**  $\text{Tgti } j (X \star_i Y) = \text{Tgti } j X \star_i \text{Tgti } j Y$

**proof** –

**{fix**  $a$

**have**  $(a \in \text{Tgti } j (X \star_i Y)) = (\exists c \in X. \exists d \in Y. a \in \text{Tgti } j (c \odot_i d))$

**using** *local.iconv-def* **by** *force*

**also have**  $\dots = (\exists c \in X. \exists d \in Y. a \in \text{tgt } j c \odot_i \text{tgt } j d)$

**using** *assms local.tj-hom-strong* **by** *force*

**also have**  $\dots = (a \in \text{Tgti } j X \star_i \text{Tgti } j Y)$

**by** (*simp add: local.iconv-def*)

**finally have**  $(a \in \text{Tgti } j (X \star_i Y)) = (a \in \text{Tgti } j X \star_i \text{Tgti } j Y).$

**thus** *?thesis*

**by** *force*

**qed**

**lemma** *Tgtj-hom-strong-var*:  $\text{Tgti } (i + k + 1) (X \star_i Y) = \text{Tgti } (i + k + 1) X \star_i \text{Tgti } (i + k + 1) Y$

**using** *Tgtj-hom-strong* **by** *auto*

**lemma** *srcfixij-var2*:  $i < j \implies \text{srcfix } j \star_i \text{srcfix } j \subseteq \text{srcfix } j$   
**by** (*metis local.Srcj-hom-strong local.stimm.stopp.Tgt-subid local.stimm.stopp.tfix-im*)

**lemma** *srcfixij-var22*:  $\text{srcfix } (i + k + 1) \star_i \text{srcfix } (i + k + 1) \subseteq \text{srcfix } (i + k + 1)$   
**using** *Suc-eq-plus1 less-add-Suc1 local.srcfixij-var2* **by** *presburger*

**lemma** *srcfixij-eq*:  $i < j \implies \text{srcfix } j \star_i \text{srcfix } j = \text{srcfix } j$   
**unfolding** *iconv-def*  
**apply** *safe*  
**apply** (*metis imageE local.sj-hom-strong local.stimm.stopp.tt-idem*)  
**by** (*smt (verit, ccfv-threshold) UN-iff local.sjsi local.stimm.stopp.ts-compat local.tgt-absorb mem-Collect-eq singletonI*)

**lemma** *srcfixij-eq-var* [*simp*]:  $\text{srcfix } (i + k + 1) \star_i \text{srcfix } (i + k + 1) = \text{srcfix } (i + k + 1)$   
**using** *Suc-eq-plus1 less-add-Suc1 srcfixij-eq* **by** *presburger*

**end**

### 7.3 $\omega$ -Catoids and single-set $\omega$ -categories

**class** *omega-catoid-strong* = *omega-st-multimagma-strong* + *omega-catoid*

**class** *local-omega-catoid* = *omega-st-multimagma* + *local-icatoid*

**class** *functional-omega-catoid* = *omega-st-multimagma* + *functional-icatoid*

**class** *local-omega-catoid-strong* = *omega-st-multimagma-strong* + *local-icatoid*

**class** *omega-category* = *omega-st-multimagma* + *icategory*

**begin**

**lemma** *sj-hom-strong*:  
**assumes**  $i < j$   
**shows**  $\text{Srci } j (x \odot_i y) = \text{src } j x \odot_i \text{src } j y$   
**by** (*smt (verit, best) assms image-is-empty less-or-eq-imp-le licat.src-local-eq local.pmi.functionality-lem-var local.si-hom local.sisj local.stimm.stopp.Dst local.tgt-absorb local.tisj nat-neq-iff subset-singletonD*)

**lemma** *sj-hom-strong-var*:  $\text{Srci } (i + k + 1) (x \odot_i y) = \text{src } (i + k + 1) x \odot_i \text{src } (i + k + 1) y$   
**using** *local.sj-hom-strong* **by** *force*

**lemma** *sj-hom-strong-delta*:  $i < j \implies DD \ i \ x \ y = DD \ i \ (\text{src } j \ x) \ (\text{src } j \ y)$   
**using** *local.sisj local.tisj stimm.locality* **by** *simp*

**lemma** *tj-hom-strong*:  $i < j \implies Tgti\ j\ (x \odot_i\ y) = tgt\ j\ x \odot_i\ tgt\ j\ y$   
**by** (*smt* (*verit*, *best*) *empty-is-image licat.st-local local.olropp.si-hom local.pmi.functionality-lem-var local.sitj local.titi order.strict-iff-order subset-singleton-iff*)

**lemma** *tj-hom-strong-var*:  $Tgti\ (i + k + 1)\ (x \odot_i\ y) = tgt\ (i + k + 1)\ x \odot_i\ tgt\ (i + k + 1)\ y$   
**by** (*simp add: local.tj-hom-strong*)

**lemma** *tj-hom-strong-delta*:  $i < j \implies DD\ i\ x\ y = DD\ i\ (tgt\ j\ x)\ (tgt\ j\ y)$   
**using** *less-or-eq-imp-le licat.st-local local.sitj local.titi* **by** *simp*

**lemma** *convi-sgl*:  $a \in x \odot_i\ y \implies \{a\} = x \odot_i\ y$   
**by** (*simp add: local.pmi.fun-in-sgl*)

Next we derive some simple globular properties.

**lemma** *strong-interchange-STj*:  
**assumes**  $i < j$   
**and**  $a \in (w \odot_i\ x) \star_j\ (y \odot_i\ z)$   
**shows**  $Tgti\ j\ (w \odot_i\ x) = Srci\ j\ (y \odot_i\ z)$   
**by** (*smt* (*verit*) *assms(2) image-empty image-insert local.iconv-prop local.pmi.fun-in-sgl local.pmi.pcomp-def-var local.stimm.stopp.Dst multimagma.conv-exp2*)

**lemma** *strong-interchange-ssi*:  
**assumes**  $i < j$   
**and**  $a \in (w \odot_i\ x) \star_j\ (y \odot_i\ z)$   
**shows**  $src\ i\ w = src\ i\ y$   
**by** (*smt* (*verit*, *ccfv-threshold*) *assms icat.src-comp-aux icat.tgt-comp-aux less-or-eq-imp-le local.iconv-prop local.sisj local.sitj multimagma.conv-exp2*)

**end**

## 7.4 Reduced axiomatisations

**class** *omega-st-multimagma-red* = *st-multimagma* +  
**assumes** *interchange*:  $i < j \implies (w \odot_j\ x) \star_i\ (y \odot_j\ z) \subseteq (w \odot_i\ y) \star_j\ (x \odot_i\ z)$   
**assumes** *srcj-hom*:  $i < j \implies Srci\ j\ (x \odot_i\ y) = src\ j\ x \odot_i\ srcj\ y$   
**and** *tgtj-hom*:  $i < j \implies Tgti\ j\ (x \odot_i\ y) = tgt\ j\ x \odot_i\ tgt\ j\ y$   
**and** *srci-weak-hom*:  $i < j \implies Srci\ i\ (x \odot_j\ y) \subseteq src\ i\ x \odot_j\ src\ i\ y$   
**and** *tgti-weak-hom*:  $i < j \implies Tgti\ i\ (x \odot_j\ y) \subseteq src\ i\ x \odot_j\ src\ i\ y$

**begin**

**lemma** *sitjsi* [*simp*]:  $src\ i\ (tgt\ j\ (src\ i\ x)) = src\ i\ x$   
**by** (*smt* (*z3*) *empty-iff image-empty image-insert insert-subset less-add-Suc1 local.srcj-hom local.stimm.stopp.t-idem stim.s-absorb-var2 subset-empty*)

**lemma** *tisjsi* [*simp*]:  $tgt\ i\ (src\ j\ (src\ i\ x)) = src\ i\ x$   
**by** (*smt* (*verit*) *local.srcj-hom local.stimm.stopp.s-absorb-var3 local.stimm.stopp.ts-compat not-less-iff-gr-or-eq sitjsi*)



**lemma** *sjsi*:  
**assumes**  $i \leq j$   
**shows**  $\text{src } j (\text{src } i x) = \text{src } i x$   
**by** (*smt* (*verit*) *antisym-conv2* *assms* *insertE* *insert-absorb* *insert-subset* *local.Dst*  
*local.iconv-def* *local.interchange* *local.src-absorb* *local.stimm.conv-def* *local.stimm.stopp.conv-atom*  
*local.tgt-absorb* *local.tgtj-hom* *stimm.s-absorb-var2* *stimm.t-export* *tisjsi*)

**lemma** *sjti*:  $i \leq j \implies \text{src } j (\text{tgt } i x) = \text{tgt } i x$   
**by** (*metis* *local.sjsi* *local.stimm.stopp.ts-compat*)

**lemma** *tjsi*:  $i \leq j \implies \text{tgt } j (\text{src } i x) = \text{src } i x$   
**by** (*metis* *antisym-conv2* *local.sjsi* *stimm.ts-compat*)

**lemma** *tjti*:  $i \leq j \implies \text{tgt } j (\text{tgt } i x) = \text{tgt } i x$   
**by** (*simp* *add*: *local.sjti* *stimm.st-fix*)

**lemma** *comm-sisj*:  $i \neq j \implies \text{src } i (\text{src } j x) = \text{src } j (\text{src } i x)$   
**by** (*smt* (*verit*, *ccfv-threshold*) *less-or-eq-imp-le* *local.sjsi* *local.srcj-hom* *not-less-iff-gr-or-eq*  
*stimm.s-ortho-iff* *tisjsi*)

**lemma** *comm-sitj*:  $i \neq j \implies \text{src } i (\text{tgt } j x) = \text{tgt } j (\text{src } i x)$   
**by** (*smt* (*verit*, *best*) *local.srcj-hom* *local.stimm.stopp.ts-compat* *local.tjsi* *nat-less-le*  
*nle-le* *stimm.s-absorb-var*)

**lemma** *comm-titj*:  $i \neq j \implies \text{tgt } i (\text{tgt } j x) = \text{tgt } j (\text{tgt } i x)$   
**by** (*smt* (*verit*, *ccfv-SIG*) *local.comm-sisj* *local.comm-sitj* *local.stimm.stopp.ts-compat*  
*tisjsi*)

**end**

**class** *omega-catoid-red* = *icatoid* +  
**assumes** *interchange*:  $i < j \implies (w \odot_j x) \star_i (y \odot_j z) \subseteq (w \odot_i y) \star_j (x \odot_i z)$   
**and** *sj-hom*:  $i < j \implies \text{Src } i j (x \odot_i y) \subseteq \text{src } j x \odot_i \text{src } j y$   
**and** *tj-hom*:  $i < j \implies \text{Tgt } i j (x \odot_i y) \subseteq \text{tgt } j x \odot_i \text{tgt } j y$

**begin**

**lemma** *sitjsi*:  
**assumes**  $i < j$   
**shows**  $\text{src } i (\text{tgt } j (\text{src } i x)) = \text{src } i x$   
**by** (*metis* (*no-types*, *lifting*) *SUP-empty* *assms* *local.iconv-prop* *local.interchange*  
*local.src-absorb* *local.stimm.stopp.conv-atom* *local.stimm.stopp.conv-def* *local.tgt-absorb*  
*order-less-imp-le* *stimm.s-absorb-var3* *subset-empty*)

**lemma** *tisjsi*:  $i < j \implies \text{tgt } i (\text{src } j (\text{src } i x)) = \text{src } i x$   
**by** (*smt* (*verit*, *ccfv-SIG*) *image-is-empty* *local.sitjsi* *local.sj-hom* *local.stimm.stopp.Dst*  
*local.stimm.stopp.st-ortho-iff* *stimm.s-absorb-var* *subset-empty*)

**lemma** *sjsi*:  
**assumes**  $i < j$   
**shows**  $\text{src } j (\text{src } i x) = \text{src } i x$   
**by** (*smt (verit, ccfv-SIG) assms empty-iff insert-iff insert-subset less-or-eq-imp-le local.iconv-prop local.interchange local.sitjsi local.src-absorb local.stimm.stopp.conv-atom local.stimm.stopp.s-absorb-var2 local.tgt-absorb local.tisjsi stim.s-ortho-iff*)

**lemma** *sjti*:  $i < j \implies \text{src } j (\text{tgt } i x) = \text{tgt } i x$   
**by** (*metis local.sjsi local.stimm.stopp.ts-compat*)

**lemma** *tjsi*:  $i < j \implies \text{tgt } j (\text{src } i x) = \text{src } i x$   
**by** (*simp add: local.sjsi stim.st-fix*)

**lemma** *tjti*:  $i < j \implies \text{tgt } j (\text{tgt } i x) = \text{tgt } i x$   
**by** (*simp add: local.sjti stim.st-fix*)

**lemma** *comm-sisj*:  $i < j \implies \text{src } i (\text{src } j x) = \text{src } j (\text{src } i x)$   
**by** (*metis (no-types, opaque-lifting) empty-iff image-insert insert-subset local.sj-hom local.sjsi local.src-absorb local.stimm.stopp.Dst stim.ts-compat*)

**lemma** *comm-sitj*:  $i < j \implies \text{src } i (\text{tgt } j x) = \text{tgt } j (\text{src } i x)$   
**by** (*metis empty-iff image-insert insert-subset local.src-absorb local.tj-hom local.tjsi stim.s-absorb-var2*)

**lemma** *comm-tisj*:  $i < j \implies \text{tgt } i (\text{src } j x) = \text{src } j (\text{tgt } i x)$   
**by** (*metis empty-iff image-insert insert-subset local.sj-hom local.sjti local.stimm.stopp.s-absorb-var3 local.tgt-absorb*)

**lemma** *comm-titj*:  $i < j \implies \text{tgt } i (\text{tgt } j x) = \text{tgt } j (\text{tgt } i x)$   
**by** (*smt (verit) bot.extremum-uniqueI local.sjti local.stimm.stopp.s-absorb-var local.stimm.stopp.s-export local.tgt-absorb local.tj-hom stim.st-fix*)

**lemma** *si-hom*:  $i < j \implies \text{Src } i (x \odot_j y) \subseteq \text{src } i x \odot_j \text{src } i y$   
**by** (*smt (verit, del-insts) icat.tgt-comp-aux image-subset-iff local.comm-sisj local.comm-sitj local.icat.ts-msg.src-twisted-aux local.src-absorb local.stimm.stopp.Dst local.tjsi singletonI*)

**lemma** *ti-hom*:  $i < j \implies \text{Tgt } i (x \odot_j y) \subseteq \text{tgt } i x \odot_j \text{tgt } i y$   
**by** (*smt (verit, ccfv-SIG) comm-tisj icat.tgt-comp-aux image-subset-iff local.comm-titj local.stimm.stopp.Dst local.tgt-absorb local.tjti singletonI*)

**end**

**class** *omega-catoid-red-strong* = *icatoid* +  
**assumes** *interchange*:  $i < j \implies (w \odot_j x) \star_i (y \odot_j z) \subseteq (w \odot_i y) \star_j (x \odot_i z)$   
**and** *sj-hom-strong*:  $i \leq j \implies \text{Src } j (x \odot_i y) = \text{src } j x \odot_i \text{src } j y$   
**and** *tj-hom-strong*:  $i \leq j \implies \text{Tgt } j (x \odot_i y) = \text{tgt } j x \odot_i \text{tgt } j y$

**begin**

**lemma** *sitjsi*:  $i < j \implies \text{src } i (\text{tgt } j (\text{src } i x)) = \text{src } i x$   
 by (*metis UN-empty Union-empty dual-order.strict-iff-not local.src-absorb local.stimm.stopp.s-ortho-id local.tj-hom-strong stimmm.Tgt-Sup-pres stimmm.s-absorb-var stimmm.s-ortho-id stimmm.t-export stimmm.tt-idem*)

**lemma** *tisjsi*:  $i < j \implies \text{tgt } i (\text{src } j (\text{src } i x)) = \text{src } i x$   
 by (*metis image-empty less-or-eq-imp-le local.sj-hom-strong local.stimm.stopp.Dst local.stimm.stopp.s-absorb-var2 stimmm.ts-compatible*)

**lemma** *sjsi*:  
 assumes  $i < j$   
 shows  $\text{src } j (\text{src } i x) = \text{src } i x$   
**proof** –  
 have  $\{\text{src } i x\} = \text{src } i x \odot_i \text{src } i x$   
 by *simp*  
 also have  $\dots = (\text{src } j (\text{src } i x) \odot_j \text{src } i x) \star_i (\text{src } i x \odot_j \text{tgt } j (\text{src } i x))$   
 by (*simp add: local.iconv-prop*)  
 also have  $\dots \subseteq (\text{src } j (\text{src } i x) \odot_i \text{src } i x) \star_j (\text{src } i x \odot_i \text{tgt } j (\text{src } i x))$   
 by (*meson assms local.interchange*)  
 also have  $\dots = (\text{src } j (\text{src } i x) \odot_i \text{tgt } i (\text{src } j (\text{src } i x))) \star_j (\text{src } i (\text{tgt } j (\text{src } i x)))$   
 $\odot_i \text{tgt } j (\text{src } i x)$   
 by (*simp add: assms local.sitjsi local.tisjsi*)  
 also have  $\dots = \text{src } j (\text{src } i x) \odot_j \text{tgt } j (\text{src } i x)$   
 using *local.iconv-prop* by *simp*  
 finally have  $\{\text{src } i x\} \subseteq \text{src } j (\text{src } i x) \odot_j \text{tgt } j (\text{src } i x)$ .  
 thus *?thesis*  
 using *local.stimm.stopp.Dst* by *force*  
**qed**

**lemma** *sjti*:  $i < j \implies \text{src } j (\text{tgt } i x) = \text{tgt } i x$   
 by (*metis local.sjsi local.stimm.stopp.ts-compatible*)

**lemma** *tjsi*:  $i < j \implies \text{tgt } j (\text{src } i x) = \text{src } i x$   
 using *local.sjsi stimmm.st-fix* by *blast*

**lemma** *tjti*:  $i < j \implies \text{tgt } j (\text{tgt } i x) = \text{tgt } i x$   
 by (*simp add: local.sjti stimmm.st-fix*)

**lemma** *comm-sisj*:  $i < j \implies \text{src } i (\text{src } j x) = \text{src } j (\text{src } i x)$   
 by (*smt (verit) less-or-eq-imp-le local.sj-hom-strong local.sjsi local.src-absorb stimmm.s-absorb-var3*)

**lemma** *comm-sitj*:  $i < j \implies \text{src } i (\text{tgt } j x) = \text{tgt } j (\text{src } i x)$   
 by (*metis local.tj-hom-strong local.tjsi order.strict-iff-not stimmm.s-absorb-var2*)

**lemma** *comm-tisj*:  $i < j \implies \text{tgt } i (\text{src } j x) = \text{src } j (\text{tgt } i x)$   
 by (*metis dual-order.strict-implies-order local.sj-hom-strong local.sjti local.stimm.stopp.s-absorb-var*)

```

lemma comm-titj:  $i < j \implies \text{tgt } i (\text{tgt } j x) = \text{tgt } j (\text{tgt } i x)$ 
  by (metis image-empty less-or-eq-imp-le local.comm-sitj local.sj-hom-strong local.stimm.stopp.Dst stim.s-ortho-iff)

lemma s0-weak-hom:  $i < j \implies \text{Srci } i (x \odot_j y) \subseteq \text{src } i x \odot_j \text{src } i y$ 
  by (smt (verit, best) image-subsetI insertI1 local.comm-sisj local.comm-sitj local.icat.ts-msg.tgt-comp-aux local.sjsi local.src-absorb local.stimm.stopp.Dst local.tjsi)

lemma t0-weak-hom:  $i < j \implies \text{Tgti } i (x \odot_j y) \subseteq \text{tgt } i x \odot_j \text{tgt } i y$ 
  by (smt (verit, ccfv-SIG) icat.tgt-comp-aux image-subset-iff local.comm-tisj local.comm-titj local.sjsi local.stimm.stopp.Dst local.stimm.stopp.ts-compat local.tgt-absorb singletonI stim.ts-compat)

end

end

```

## 8 $\omega$ -Kleene algebras

```

theory Omega-Kleene-Algebra
  imports Quantales-Converse.Modal-Kleene-Algebra-Var

```

```

begin

```

Here we develop  $\omega$ -semigroups and  $\omega$ -Kleene algebras.

### 8.1 Copies for i-structures

```

class icomp-op =
  fixes icomp :: 'a  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  'a (- .. - [70,70,70]70)

class iid-op =
  fixes un :: nat  $\Rightarrow$  'a

class istar-op =
  fixes star :: nat  $\Rightarrow$  'a  $\Rightarrow$  'a

class idom-op =
  fixes do :: nat  $\Rightarrow$  'a  $\Rightarrow$  'a

class icod-op =
  fixes cd :: nat  $\Rightarrow$  'a  $\Rightarrow$  'a

class imonoid-mult = icomp-op + iid-op +
  assumes assoc:  $x \cdot_i (y \cdot_i z) = (x \cdot_i y) \cdot_i z$ 
  and comp-unl:  $un \ i \cdot_i x = x$ 
  and comp-unr:  $x \cdot_i un \ i = x$ 

class idiod = imonoid-mult + join-semilattice-zero +

```

```

assumes distl:  $x \cdot_i (y + z) = x \cdot_i y + x \cdot_i z$ 
and distr:  $(x + y) \cdot_i z = x \cdot_i z + y \cdot_i z$ 
and annil:  $0 \cdot_i x = 0$ 
and annir:  $x \cdot_i 0 = 0$ 

class ikleene-algebra = idiod + istar-op +
  assumes star-unfoldl:  $un\ i + x \cdot_i\ star\ i\ x \leq star\ i\ x$ 
  and star-unfoldr:  $un\ i + star\ i\ x \cdot_i\ x \leq star\ i\ x$ 
  and star-inductl:  $z + x \cdot_i\ y \leq y \implies star\ i\ x \cdot_i\ z \leq y$ 
  and star-inductr:  $z + y \cdot_i\ x \leq y \implies z \cdot_i\ star\ i\ x \leq y$ 

class idomain-semiring = idiod + idom-op +
  assumes do-absorb:  $x \leq do\ i\ x \cdot_i\ x$ 
  and do-local:  $do\ i\ (x \cdot_i\ do\ i\ y) = do\ i\ (x \cdot_i\ y)$ 
  and do-add:  $do\ i\ (x + y) = do\ i\ x + do\ i\ y$ 
  and do-subid:  $do\ i\ x \leq un\ i$ 
  and do-zero:  $do\ i\ 0 = 0$ 

class icodomain-semiring = idiod + icod-op +
  assumes cd-absorb:  $x \leq x \cdot_i\ cd\ i\ x$ 
  and cd-local:  $cd\ i\ (cd\ i\ x \cdot_i\ y) = cd\ i\ (x \cdot_i\ y)$ 
  and cd-add:  $cd\ i\ (x + y) = cd\ i\ x + cd\ i\ y$ 
  and cd-subid:  $cd\ i\ x \leq un\ i$ 
  and cd-zero:  $cd\ i\ 0 = 0$ 

class imodal-semiring = idomain-semiring + icodomain-semiring +
  assumes dc-compat:  $do\ i\ (cd\ i\ x) = cd\ i\ x$ 
  and cd-compat:  $cd\ i\ (do\ i\ x) = do\ i\ x$ 

class imodal-ikleene-algebra = imodal-semiring + ikleene-algebra

sublocale imonoid-mult  $\subseteq$  mm: monoid-mult  $un\ i\ \lambda x\ y.\ x \cdot_i\ y$ 
  by (unfold-locales, simp-all add: local.assoc local.comp-unl local.comp-unr)

sublocale idiod  $\subseteq$  di: diod-one-zero -  $\lambda x\ y.\ x \cdot_i\ y$  un i - - -
  by (unfold-locales, simp-all add: local.distl local.distr annil annir)

sublocale idiod  $\subseteq$  ddi: idiod - - -  $\lambda x\ i\ y.\ icomp\ y\ i\ x$  -
  by unfold-locales (simp-all add: local.mm.mult-assoc local.distl)

sublocale ikleene-algebra  $\subseteq$  ki: ikleene-algebra -  $\lambda x\ y.\ x \cdot_i\ y$  un i - - - star i
  apply unfold-locales
  using local.star-unfoldl apply blast
  by (simp-all add: local.star-inductl local.star-inductr local.star-unfoldl)

sublocale ikleene-algebra  $\subseteq$  dki: ikleene-algebra - - - -  $\lambda x\ i\ y.\ y \cdot_i\ x$  - -
  by (unfold-locales, simp-all add: local.star-inductr local.star-inductl)

sublocale idomain-semiring  $\subseteq$  dsri: domain-semiring -  $\lambda x\ y.\ x \cdot_i\ y$  un i - do i - -

```

**by** *unfold-locales* (*simp-all* add: *local.do-absorb local.join.sup-absorb2 local.do-local local.do-zero local.do-add local.do-subid*)

**sublocale** *icodomain-semiring*  $\subseteq$  *csri: range-semiring* -  $\lambda x y. x \cdot_i y$  un *i* - *cd i* - -  
**by** *unfold-locales* (*simp-all* add: *local.cd-absorb local.join.sup-absorb2 local.cd-local local.cd-subid local.cd-zero local.cd-add*)

**sublocale** *icodomain-semiring*  $\subseteq$  *ds0dual: idomain-semiring* - - - -  $\lambda x i y. y \cdot_i x$  - *cd*  
**by** *unfold-locales simp-all*

**sublocale** *imodal-semiring*  $\subseteq$  *msri: dr-modal-semiring* -  $\lambda x y. x \cdot_i y$  un *i* - *do i* - - *cd i*  
**by** (*unfold-locales, simp-all* add: *local.dc-compat local.cd-compat*)

**sublocale** *imodal-semiring*  $\subseteq$  *msridual: imodal-semiring do* - - - -  $\lambda x i y. y \cdot_i x$  - *cd*  
**by** *unfold-locales simp-all*

**sublocale** *imodal-kleene-algebra*  $\subseteq$  *mkai: dr-modal-kleene-algebra* -  $\lambda x y. x \cdot_i y$  un *i* - - - *star i do i cd i..*

**sublocale** *imodal-kleene-algebra*  $\subseteq$  *mkaidual: imodal-kleene-algebra* - - - -  $\lambda x i y. y \cdot_i x$  - - *do cd ..*

## 8.2 Globular $\omega$ -semirings

**class** *omega-semiring* = *imodal-semiring* +  
**assumes** *interchange*:  $i < j \implies (w \cdot_j x) \cdot_i (y \cdot_j z) \leq (w \cdot_i y) \cdot_j (x \cdot_i z)$   
**and** *di-hom*:  $i \neq j \implies do\ i\ (x \cdot_j y) \leq do\ i\ x \cdot_j do\ i\ y$   
**and** *ci-hom*:  $i \neq j \implies cd\ i\ (x \cdot_j y) \leq cd\ i\ x \cdot_j cd\ i\ y$   
**and** *djdi*:  $i < j \implies do\ j\ (do\ i\ x) = do\ i\ x$

**class** *strong-omega-semiring* = *omega-semiring* +  
**assumes** *dj-strong-hom*:  $i < j \implies do\ j\ (x \cdot_i y) = do\ j\ x \cdot_i do\ j\ y$   
**and** *cj-strong-hom*:  $i < j \implies cd\ j\ (x \cdot_i y) = cd\ j\ x \cdot_i cd\ j\ y$

**sublocale** *omega-semiring*  $\subseteq$  *tgsdual: omega-semiring do* - - - -  $\lambda x i y. y \cdot_i x$  - *cd*  
**apply** *unfold-locales*  
**apply** (*simp-all* add: *local.interchange local.ci-hom local.di-hom*)  
**by** (*metis local.dc-compat local.djdi msri.d-cod-fix*)

**sublocale** *strong-omega-semiring*  $\subseteq$  *stgsdual: strong-omega-semiring do* - - - -  $\lambda x i y. y \cdot_i x$  - *cd*  
**by** *unfold-locales* (*simp-all* add: *local.cj-strong-hom local.dj-strong-hom*)

**context** *omega-semiring*  
**begin**

**lemma** *interchange-var*:  $(w \cdot (i + k + 1) x) \cdot i (y \cdot (i + k + 1) z) \leq (w \cdot i y) \cdot (i + k + 1) (x \cdot_i z)$   
**by** (*simp add: local.interchange*)

**lemma** *djdi-var* [*simp*]:  $do (i + k + 1) (do i x) = do i x$   
**by** (*simp add: local.djdi*)

**lemma** *cjdi*:  $i \leq j \implies cd j (do i x) = do i x$   
**by** (*metis local.cd-compat local.djdi order-class.nless-le*)

**lemma** *cjdi-var* [*simp*]:  $cd (i + k) (do i x) = do i x$   
**by** (*simp add: cjdi*)

**lemma** *djci*:  $i \leq j \implies do j (cd i x) = cd i x$   
**by** (*metis cjdi local.dc-compat*)

**lemma** *djci-var* [*simp*]:  $do (i + k) (cd i x) = cd i x$   
**by** (*simp add: djci*)

**lemma** *cjci*:  $i \leq j \implies cd j (cd i x) = cd i x$   
**using** *djci msri.d-cod-fix* **by** *force*

**lemma** *cjci-var* [*simp*]:  $cd (i + k) (cd i x) = cd i x$   
**by** (*simp add: cjci*)

**lemma** *unj-compi-var*:  $i \leq j \implies un j \leq un j \cdot_i un i$   
**by** (*metis cjdi local.cd-subid local.di.mult-isol local.dsri.dom-one local.mm.mult-1-right*)

**lemma** *un-iso*:  $i \leq j \implies un i \leq un j$   
**by** (*metis cjdi-var le-Suc-ex local.cd-subid local.dsri.dom-one*)

**lemma** *uni-compj-eq* :  $i < j \implies un i \cdot_j un i = un i$   
**by** (*metis local.djdi local.dsri.dom-one local.dsri.dsg1*)

**lemma** *uni-compj-eq-var* [*simp*]:  $un i \cdot_{(i + k)} un i = un i$   
**by** (*metis cjdi le-add1 local.csri.rdual.dns1'' local.dsri.dom-one*)

**lemma** *dj-uni*:  $i < j \implies do j (un i) = un i$   
**by** (*metis local.djdi local.dsri.dom-one*)

**lemma** *dj-uni-var* [*simp*]:  $do (i + k) (un i) = un i$   
**by** (*metis djci-var local.csri.rdual.dom-one*)

**lemma** *di-unj*:  $i < j \implies do i (un j) = un i$   
**by** (*metis dj-uni local.do-subid local.dsri.dom-iso local.dsri.dom-one local.dual-order.antisym*)

**lemma** *di-unj-var* [*simp*]:  $do i (un (i + k)) = un i$   
**by** (*metis di-unj le-add1 local.dsri.dom-one order-neq-le-trans*)

**lemma** *ci-unj*:  $i < j \implies cd\ i\ (un\ j) = un\ i$   
**by** (*metis less-or-eq-imp-le local.cd-subid local.csri.rdual.dom-iso local.csri.rdual.dom-one local.dual-order.antisym un-iso*)

**lemma** *ci-unj-var* [*simp*]:  $cd\ i\ (un\ (i + k)) = un\ i$   
**by** (*metis ci-unj le-add1 local.csri.rdual.dom-one order-neq-le-trans*)

**lemma** *cj-uni*:  $i < j \implies cd\ j\ (un\ i) = un\ i$   
**using** *dj-uni local.msri.msrdual.d-cod-fix* **by** *force*

**lemma** *cj-uni-var* [*simp*]:  $cd\ (i + k)\ (un\ i) = un\ i$   
**by** (*simp add: local.msri.msrdual.d-cod-fix*)

**lemma** *comm-didj*:  $i \leq j \implies do\ i\ (do\ j\ x) = do\ j\ (do\ i\ x)$

**proof** –

**have** *h*:  $i < j \implies do\ i\ (do\ j\ x) = do\ j\ (do\ i\ x)$   
**by** (*smt (verit) local.di-hom local.djdi local.dsri.dom-llp local.dsri.dom-subid-aux2 local.dual-order.eq-iff*)  
**have**  $i = j \implies do\ i\ (do\ j\ x) = do\ j\ (do\ i\ x)$   
**by** *simp*  
**thus** *?thesis*  
**by** (*smt (verit) local.di-hom local.djdi local.dsri.dom-llp local.dsri.dsg1 local.dual-order.antisym nat-neq-iff*)  
**qed**

**lemma** *comm-didj-var*:  $do\ i\ (do\ (i + k)\ x) = do\ (i + k)\ (do\ i\ x)$   
**by** (*meson comm-didj le-add1*)

**lemma** *comm-dicj*:  $i < j \implies do\ i\ (cd\ j\ x) = cd\ j\ (do\ i\ x)$   
**by** (*smt (verit, ccfv-threshold) cjdi less-or-eq-imp-le local.csri.rdual.d-preserves-equation local.csri.rdual.dns1'' local.dc-compat local.di-hom local.dsri.d-preserves-equation local.dsri.dom-llp local.dsri.dsg1 local.tgsdual.di-hom nat-neq-iff*)

**lemma** *comm-dicj-var*:  $do\ i\ (cd\ (i + k + 1)\ x) = cd\ (i + k + 1)\ (do\ i\ x)$   
**using** *comm-dicj* **by** *auto*

**lemma** *comm-cicj*:  $i \leq j \implies cd\ i\ (cd\ j\ x) = cd\ j\ (cd\ i\ x)$   
**by** (*smt (verit) cjci local.csri.rdual.dns1'' local.csri.rdual.dom-llp local.dual-order.antisym local.tgsdual.di-hom*)

**lemma** *comm-cicj-var* [*simp*]:  $cd\ i\ (cd\ (i + k)\ x) = cd\ (i + k)\ (cd\ i\ x)$   
**by** (*meson comm-cicj le-add1*)

**lemma** *comm-cidj*:  $i < j \implies cd\ i\ (do\ j\ x) = do\ j\ (cd\ i\ x)$   
**by** (*smt (verit) comm-cicj djci less-or-eq-imp-le local.cd-compat local.csri.rdual.dns1'' local.csri.rdual.dom-llp local.di-hom local.dsri.dom-subid-aux2'' local.dsri.dsg1 local.dual-order.antisym local.tgsdual.di-hom nat-neq-iff*)

We prove further lemmas that are not related to the globular structure.



**lemma** *di-compi-idem*:  $i \leq j \implies do\ i\ x \cdot_j\ do\ i\ x = do\ i\ x$   
**by** (*metis* *cjdi local.csri.rdual.dns1''*)

**lemma** *di-compi-idem-var* [*simp*]:  $do\ i\ x \cdot_{(i+k)}\ do\ i\ x = do\ i\ x$   
**by** (*simp* *add: di-compi-idem*)

**lemma** *codi-compj-idem*:  $i \leq j \implies cd\ i\ x \cdot_j\ cd\ i\ x = cd\ i\ x$   
**by** (*metis* *djci local.dsri.dsg1*)

**lemma** *codi-compj-idem-var* [*simp*]:  $cd\ i\ x \cdot_{(i+k)}\ cd\ i\ x = cd\ i\ x$   
**by** (*simp* *add: codi-compj-idem*)

**lemma** *domij-loc*:  $i \leq j \implies do\ i\ (x \cdot_j\ do\ j\ y) = do\ i\ (x \cdot_j\ y)$   
**by** (*smt* (*verit, ccfv-SIG*) *comm-didj local.djdi local.do-local order-class.dual-order.order-iff-strict*)

**lemma** *domij-loc-var* [*simp*]:  $do\ i\ (x \cdot_{(i+k)}\ do\ (i+k)\ y) = do\ i\ (x \cdot_{(i+k)}\ y)$   
**by** (*simp* *add: domij-loc*)

**lemma** *codij-locl*:  $i \leq j \implies cd\ i\ (cd\ j\ x \cdot_j\ y) = cd\ i\ (x \cdot_j\ y)$   
**by** (*smt* (*verit, ccfv-SIG*) *cjci comm-cicj local.ds0dual.do-local*)

**lemma** *codij-locl-var* [*simp*]:  $cd\ i\ (cd\ (i+k)\ x \cdot_{(i+k)}\ y) = cd\ i\ (x \cdot_{(i+k)}\ y)$   
**by** (*simp* *add: codij-locl*)

**lemma** *domij-exp*:  $i < j \implies do\ i\ (cd\ j\ x \cdot_j\ y) = do\ i\ (x \cdot_j\ y)$   
**by** (*metis* *cjdi comm-dicj local.cd-local preorder-class.dual-order.strict-implies-order*)

**lemma** *domij-exp-var* [*simp*]:  $do\ i\ (cd\ (i+k+1)\ x \cdot_{(i+k+1)}\ y) = do\ i\ (x \cdot_{(i+k+1)}\ y)$   
**using** *domij-exp* **by** *force*

**lemma** *codij-exp*:  $i < j \implies cd\ i\ (x \cdot_j\ do\ j\ y) = cd\ i\ (x \cdot_j\ y)$   
**by** (*metis* *comm-cidj djci less-or-eq-imp-le local.do-local*)

**lemma** *codij-exp-var* [*simp*]:  $cd\ i\ (x \cdot_{(i+k+1)}\ do\ (i+k+1)\ y) = cd\ i\ (x \cdot_{(i+k+1)}\ y)$   
**using** *codij-exp* **by** *force*

**lemma** *domij-loc-var2*:  $i \leq j \implies do\ i\ (x \cdot_i\ do\ j\ y) = do\ i\ (x \cdot_i\ y)$   
**by** (*metis* *domij-loc local.do-local local.dsri.dom-el-idem local.dsri.dsg1*)

**lemma** *domij-loc-var3*:  $do\ i\ (x \cdot_i\ do\ (i+k)\ y) = do\ i\ (x \cdot_i\ y)$   
**by** (*simp* *add: domij-loc-var2*)

**lemma** *codij-loc-var*:  $i \leq j \implies cd\ i\ (cd\ j\ x \cdot_i\ y) = cd\ i\ (x \cdot_i\ y)$   
**by** (*metis* *codij-locl local.csri.rdual.dom-el-idem local.csri.rdual.dsg1 local.ds0dual.do-local*)

**lemma** *codij-loc-var2*:  $cd\ i\ (cd\ (i+k)\ x \cdot_i\ y) = cd\ i\ (x \cdot_i\ y)$

by (simp add: codij-loc-var)

**lemma** di-compj:  $i < j \implies do\ i\ x \cdot_i (y \cdot_j z) \leq (do\ i\ x \cdot_i y) \cdot_j (do\ i\ x \cdot_i z)$   
by (metis di-compi-idem local.tgsdual.interchange preorder-class.less-le-not-le)

**lemma** dj-compj:  $i < j \implies do\ j\ x \cdot_i (y \cdot_j z) \leq (do\ j\ x \cdot_i y) \cdot_j (do\ j\ x \cdot_i z)$   
by (metis local.dsri.dom-el-idem local.tgsdual.interchange)

**lemma** dij-export:  $i \leq j \implies do\ i (do\ j\ x \cdot_j y) \leq do\ i\ x \cdot_j do\ i\ y$   
by (smt (verit, cefv-SIG) comm-didj domij-loc local.ddi.di.mult-isol-var local.dsri.dom-iso local.dsri.dom-subid-aux2 local.dsri.dsg1 local.dsri.dsg4)

**lemma** dij-export-var [simp]:  $do\ i (do\ (i + k)\ x \cdot_{(i + k)} y) \leq do\ i\ x \cdot_{(i + k)} do\ i\ y$   
by (simp add: dij-export)

**lemma** codij-export:  $i \leq j \implies cd\ i (x \cdot_j cd\ j\ y) \leq cd\ i\ x \cdot_j cd\ i\ y$   
by (smt (verit, cefv-SIG) cjci comm-cicj local.ci-hom local.csri.rdual.dsg3 local.dual-order.eq-iff)

**lemma** codij-export-var [simp]:  $cd\ i (x \cdot_{(i + k)} cd\ (i + k)\ y) \leq cd\ i\ x \cdot_{(i + k)} cd\ i\ y$   
by (simp add: codij-export)

**lemma** dji-export:  $i \leq j \implies do\ j (do\ i\ x \cdot_j y) = do\ i\ x \cdot_j do\ j\ y$   
by (smt (verit, del-insts) comm-didj domij-loc local.dsri.dsg1 local.dsri.dsg3)

**lemma** dji-export-var:  $do\ (i + k) (do\ i\ x \cdot_{(i + k)} y) = do\ i\ x \cdot_{(i + k)} do\ (i + k)\ y$   
by (simp add: dji-export)

**lemma** codji-export:  $i \leq j \implies cd\ j (x \cdot_j cd\ i\ y) = cd\ j\ x \cdot_j cd\ i\ y$   
by (metis cjci local.csri.rdual.dsg3)

**lemma** codji-export-var:  $cd\ (i + k) (x \cdot_{(i + k)} cd\ i\ y) = cd\ (i + k)\ x \cdot_{(i + k)} cd\ i\ y$   
by (simp add: codji-export)

**lemma** di-compji:  $i \leq j \implies do\ i\ x \cdot_j do\ i\ y = do\ i\ x \cdot_i do\ i\ y$   
**apply** (rule order.antisym)  
**apply** (metis cjdi local.csri.rdual.dom-subid-aux2 local.csri.rdual.dom-subid-aux2'' local.ddi.di.mult-isol-var local.dsri.dom-iso local.dsri.domain-invol local.dsri.dsg1)  
by (metis local.ddi.di.mult-isol-var local.djdi local.dsri.dom-iso local.dsri.dom-subid-aux2 local.dsri.dom-subid-aux2'' local.dsri.dsg1 order-le-imp-less-or-eq)

**lemma** di-compji-var:  $do\ i\ x \cdot_{(i + k)} do\ i\ y = do\ i\ x \cdot_i do\ i\ y$   
by (meson di-compji le-add1)

**lemma** dom-exchange-strong:  $i \leq j \implies (do\ i\ w \cdot_j do\ i\ x) \cdot_i (do\ i\ y \cdot_j do\ i\ z) = (do\ i\ w \cdot_i do\ i\ y) \cdot_j (do\ i\ x \cdot_i do\ i\ z)$

**by** (*metis di-compji local.ddi.assoc local.dsri.dsg3 local.dsri.dsg4*)

**lemma** *codidomj-exp*:  $i < j \implies cd\ i\ (x \cdot_i\ y) \leq cd\ i\ (x \cdot_i\ cd\ j\ y)$

**by** (*smt (verit) local.cjci local.codij-loc-var local.comm-cicj local.csri.rdual.dom-iso order-less-le local.tgsdual.di-hom*)

**lemma** *codidomj-exp-var*:  $cd\ i\ (x \cdot_i\ y) \leq cd\ i\ (x \cdot_i\ cd\ (i + k + 1)\ y)$

**using** *codidomj-exp by force*

The following laws are diamond laws. It remains to define diamonds for them.

**lemma** *fdiaifdiaj-prop*:  $i \leq j \implies do\ i\ (y \cdot_i\ do\ j\ (x \cdot_j\ z)) = do\ i\ (y \cdot_i\ (x \cdot_j\ z))$

**by** (*simp add: domij-loc-var2*)

**lemma** *bdiaifdiaj-prop*:  $i < j \implies cd\ i\ (do\ j\ (x \cdot_j\ z) \cdot_i\ y) = cd\ i\ ((x \cdot_j\ z) \cdot_i\ y)$

**by** (*metis codij-exp local.cd-local local.dsri.dom-el-idem local.dsri.dsg1*)

**lemma** *fdiaibdiaj-prop*:  $i < j \implies do\ i\ (y \cdot_i\ cd\ j\ (x \cdot_j\ z)) = do\ i\ (y \cdot_i\ (x \cdot_j\ z))$

**by** (*metis domij-exp local.csri.rdual.dom-el-idem local.csri.rdual.dsg1 local.msridual.cd-local*)

**lemma** *bdiaibdiaj-prop*:  $i \leq j \implies cd\ i\ (cd\ j\ (x \cdot_j\ z) \cdot_i\ y) = cd\ i\ ((x \cdot_j\ z) \cdot_i\ y)$

**by** (*simp add: codij-loc-var*)

**lemma** *fdiaifdiaj-prop2*:  $i < j \implies do\ i\ (y \cdot_i\ do\ j\ (x \cdot_j\ z)) \leq do\ i\ (y \cdot_i\ (do\ i\ x \cdot_j\ do\ i\ z))$

**by** (*metis di-compji domij-loc local.di-hom local.dsri.ddual.bd-def local.dsri.dom-mult-closed local.dsri.dsg1 local.dsri.fd-iso1 nat-neq-iff preorder-class.order.strict-implies-order*)

**lemma** *fdiaii-prop2*:  $i < j \implies do\ i\ (y \cdot_i\ do\ i\ (x \cdot_j\ z)) \leq do\ i\ (y \cdot_i\ (do\ i\ x \cdot_j\ do\ i\ z))$

**using** *local.di.mult-isol local.di-hom local.dsri.dom-iso nat-neq-iff by presburger*

**lemma** *bdiaidomj-prop2*:  $i < j \implies cd\ i\ (do\ j\ (x \cdot_j\ z) \cdot_i\ y) \leq cd\ i\ ((cd\ i\ x \cdot_j\ cd\ i\ z) \cdot_i\ y)$

**by** (*metis bdiaifdiaj-prop csri.bd-def less-not-refl local.csri.rdual.dom-iso local.csri.rdual.domain-invol local.csri.rdual.fd-iso1 local.tgsdual.di-hom*)

**lemma** *bdiaidomi-prop2*:  $i < j \implies cd\ i\ (do\ i\ (x \cdot_j\ z) \cdot_i\ y) \leq cd\ i\ ((do\ i\ x \cdot_j\ do\ i\ z) \cdot_i\ y)$

**by** (*simp add: local.csri.rdual.dom-iso local.ddi.di.mult-isol-var local.di-hom*)

**lemma** *fdiaibdiaj-prop-2*:  $i < j \implies do\ i\ (y \cdot_i\ cd\ j\ (z \cdot_j\ x)) \leq do\ i\ (y \cdot_i\ (do\ i\ x \cdot_j\ do\ i\ z))$

**by** (*smt (verit) fdiaibdiaj-prop fdiaii-prop2 local.djdi local.dsri.dsg4 local.msridual.cd-local msri.d-cod-fix*)

**lemma** *fdiaibdiai-prop2*:  $i < j \implies do\ i\ (y \cdot_i\ cd\ i\ (z \cdot_j\ x)) \leq do\ i\ (y \cdot_i\ (cd\ i\ z \cdot_j\ cd\ i\ x))$

**by** (*simp add: local.di.mult-isol local.dsri.dom-iso local.tgsdual.di-hom*)

**lemma** *bdiaibdiaj-prop2*:  $i < j \implies cd\ i\ (cd\ j\ (z \cdot_j x) \cdot_i y) \leq cd\ i\ ((cd\ i\ x \cdot_j cd\ i\ z) \cdot_i y)$

**by** (*smt (z3) bdiaibdiaj-prop bdiaidomj-prop2 bdiaifdiaj-prop codji-export djci less-or-eq-imp-le local.dsri.dsg4 msri.d-cod-fix*)

**lemma** *bdiaibdiai-prop2*:  $i < j \implies cd\ i\ (cd\ i\ (x \cdot_j z) \cdot_i y) \leq cd\ i\ ((cd\ i\ x \cdot_j cd\ i\ z) \cdot_i y)$

**using** *bdiaidomj-prop2 bdiaifdiaj-prop* **by** *force*

**lemma** *fdiajfdiai-prop3*:  $i < j \implies do\ j\ (x \cdot_j do\ i\ (y \cdot_i z)) \leq do\ j\ (x \cdot_j do\ i\ (do\ j\ y \cdot_i z))$

**by** (*smt (verit) comm-didj domij-loc-var2 local.di.mult-isol local.di-hom local.djdi local.dsri.dom-iso nat-neq-iff preorder-class.order.strict-implies-order*)

**lemma** *bdiajbdiai-prop3*:  $i < j \implies cd\ j\ (cd\ i\ (z \cdot_i y) \cdot_j x) \leq cd\ j\ (cd\ i\ (z \cdot_i cd\ j\ y) \cdot_j x)$

**by** (*simp add: codidomj-exp local.csri.rdual.dom-iso local.di.mult-isol*)

**end**

The following proofs need the domain codomain duality, which has been formalised using a sublocale statement above. It is only available outside of a context.

**lemma** (*in omega-semiring*) *domicodj-exp*:  $i < j \implies do\ i\ (x \cdot_i y) \leq do\ i\ (cd\ j\ x \cdot_i y)$

**by** (*smt (verit, cfv-SIG) local.cjdi local.comm-dicj local.dsri.dom-iso local.order-eq-iff local.tgsdual.domij-loc-var local.djdi local.msridual.cd-local local.tgsdual.di-hom msri.d-cod-fix nat-neq-iff*)

**lemma** (*in omega-semiring*) *domicodj-exp-var*:  $do\ i\ (x \cdot_i y) \leq do\ i\ (cd\ (i + k + 1)\ x \cdot_i y)$

**using** *local.domicodj-exp* **by** *force*

**lemma** (*in omega-semiring*) *fdiajbdiai-prop3*:  $i < j \implies do\ j\ (x \cdot_j cd\ i\ (z \cdot_i y)) \leq do\ j\ (x \cdot_j cd\ i\ (z \cdot_i do\ j\ y))$

**by** (*simp add: local.di.mult-isol local.dsri.dom-iso local.tgsdual.domicodj-exp*)

**lemma** (*in omega-semiring*) *bdiajfdiai-prop3*:  $i < j \implies cd\ j\ (do\ i\ (y \cdot_i z) \cdot_j x) \leq cd\ j\ (do\ i\ (cd\ j\ y \cdot_i z) \cdot_j x)$

**by** (*simp add: local.tgsdual.fdiajbdiai-prop3*)

**context** *strong-omega-semiring*

**begin**

**lemma** *idj-compj*:  $i \leq j \implies un\ j \cdot_i un\ j \leq un\ j$

**by** (*metis local.cd-subid local.cj-strong-hom local.csri.rdual.dns1'' local.csri.rdual.dom-one order-le-imp-less-or-eq*)

**lemma** *idj-compi-eq*:  $i < j \implies un\ j = un\ j \cdot_i un\ j$   
**by** (*simp add: idj-compj local.order-eq-iff local.unj-compi-var*)

**lemma** *domicodj-exp*:  $i < j \implies do\ i\ (x \cdot_i y) = do\ i\ (cd\ j\ x \cdot_i y)$   
**by** (*smt (verit, del-insts) local.csri.rdual.domain-invol local.csri.rdual.dsg1 local.domij-exp local.stgsdual.dj-strong-hom*)

**lemma** *domicodj-exp-var* [*simp*]:  $do\ i\ (cd\ (i + k + 1)\ x \cdot_i y) = do\ i\ (x \cdot_i y)$   
**by** (*metis Suc-eq-plus1 less-add-Suc1 local.domicodj-exp*)

**lemma** *codidomj-exp*:  $i < j \implies cd\ i\ (x \cdot_i do\ j\ y) = cd\ i\ (x \cdot_i y)$   
**by** (*smt (verit, ccfv-SIG) local.comm-cidj local.dc-compat local.dj-strong-hom local.ds0dual.do-local local.tgsdual.djdi*)

**lemma** *codidomj-exp-var* [*simp*]:  $cd\ i\ (x \cdot_i do\ (i + k + 1)\ y) = cd\ i\ (x \cdot_i y)$   
**using** *local.codidomj-exp* **by** *force*

**lemma** *fdiajfdiai-prop3*:  $i < j \implies do\ j\ (x \cdot_j do\ i\ (do\ j\ y \cdot_i z)) = do\ j\ (x \cdot_j do\ i\ (y \cdot_i z))$   
**by** (*smt (verit, best) local.comm-didj local.dj-strong-hom local.tgsdual.cjci local.tgsdual.codij-loc-var preorder-class.less-le-not-le*)

**lemma** *fdiajbdiai-prop3*:  $i < j \implies do\ j\ (x \cdot_j cd\ i\ (z \cdot_i do\ j\ y)) = do\ j\ (x \cdot_j cd\ i\ (z \cdot_i y))$   
**by** (*simp add: local.codidomj-exp*)

**lemma** *bdiajfdiai-prop3*:  $i < j \implies cd\ j\ (do\ i\ (cd\ j\ y \cdot_i z) \cdot_j x) = cd\ j\ (do\ i\ (y \cdot_i z) \cdot_j x)$   
**by** (*metis local.domicodj-exp*)

**lemma** *bdiajbdiai-prop3*:  $i < j \implies cd\ j\ (cd\ i\ (z \cdot_i cd\ j\ y) \cdot_j x) = cd\ j\ (cd\ i\ (z \cdot_i y) \cdot_j x)$   
**by** (*smt (verit, ccfv-threshold) less-or-eq-imp-le local.cj-strong-hom local.cjci local.comm-cicj*)

**lemma** *fdiaifdiaj-prop4*:  $i < j \implies do\ i\ z \cdot_i do\ j\ (x \cdot_j y) \leq do\ j\ ((do\ i\ z \cdot_i x) \cdot_j (do\ i\ z \cdot_i y))$   
**by** (*smt (verit, ccfv-threshold) local.di-compj local.djdi local.dsri.dom-iso local.stgsdual.cj-strong-hom*)

**lemma** *fdia0bdia1-prop4*:  $i < j \implies do\ i\ z \cdot_i cd\ j\ (y \cdot_j x) \leq cd\ j\ ((do\ i\ z \cdot_i y) \cdot_j (do\ i\ z \cdot_i x))$   
**by** (*smt (verit, ccfv-threshold) local.csri.rdual.dom-iso local.di-compj local.djdi local.stgsdual.dj-strong-hom msri.d-cod-fix*)

**lemma** *fdiajfdiaj-prop4*:  $i < j \implies do\ j\ (x \cdot_j y) \cdot_i do\ i\ z \leq do\ j\ ((x \cdot_i do\ i\ z) \cdot_j (y \cdot_i do\ i\ z))$   
**by** (*smt (verit, ccfv-threshold) local.djdi local.dsri.dom-iso local.dsri.dsg1 local.stgsdual.cj-strong-hom local.tgsdual.interchange*)

**lemma** *bdiabdiaj-prop4*:  $i < j \implies cd\ j\ (y \cdot_j x) \cdot_i\ do\ i\ z \leq cd\ j\ ((y \cdot_i\ do\ i\ z) \cdot_j\ (x \cdot_i\ do\ i\ z))$   
**by** (*smt* (*verit*) *local.cd-compat* *local.csri.rdual.dom-iso* *local.stgsdual.dj-strong-hom* *local.tgsdual.di-compj* *local.tgsdual.djdi*)

**end**

### 8.3 Globular $\omega$ -Kleene algebras

**class** *omega-kleene-algebra* = *omega-semiring* + *ikleene-algebra*

**class** *strong-omega-kleene-algebra* = *strong-omega-semiring* + *ikleene-algebra*

**context** *omega-kleene-algebra*

**begin**

**lemma** *interchange-var1*:  $i < j \implies (x \cdot_j x) \cdot_i ((y \cdot_j y) \cdot_i (z \cdot_j z)) \leq (x \cdot_i (y \cdot_i z)) \cdot_j (x \cdot_i (y \cdot_i z))$   
**by** (*meson* *local.di.mult-isol* *local.interchange* *local.order-trans*)

**lemma** *interchange-var2*:  $i < j \implies (x \cdot_j y) \cdot_i ((x \cdot_j y) \cdot_i (x \cdot_j y)) \leq (x \cdot_i (x \cdot_i x)) \cdot_j (y \cdot_i (y \cdot_i y))$   
**by** (*meson* *local.di.mult-isol* *local.interchange* *local.order-trans*)

**lemma** *star-compj*:

**assumes**  $i < j$

**shows**  $star\ i\ (x \cdot_j y) \leq star\ i\ x \cdot_j star\ i\ y$

**proof** –

**have**  $a: un\ i \leq star\ i\ x \cdot_j star\ i\ y$

**by** (*metis* *assms* *local.di.mult-isol-var* *local.ki.star-ref* *local.uni-compj-eq*)

**have**  $(x \cdot_j y) \cdot_i (star\ i\ x \cdot_j star\ i\ y) \leq (x \cdot_i star\ i\ x) \cdot_j (y \cdot_i star\ i\ y)$

**by** (*simp* *add*: *assms* *local.interchange*)

**also have**  $\dots \leq star\ i\ x \cdot_j star\ i\ y$

**by** (*simp* *add*: *local.di.mult-isol-var*)

**finally have**  $(x \cdot_j y) \cdot_i (star\ i\ x \cdot_j star\ i\ y) \leq star\ i\ x \cdot_j star\ i\ y.$

**hence**  $un\ i + (x \cdot_j y) \cdot_i (star\ i\ x \cdot_j star\ i\ y) \leq star\ i\ x \cdot_j star\ i\ y$

**by** (*simp* *add*: *a*)

**thus** *?thesis*

**using** *local.star-inductl* **by** *force*

**qed**

**lemma** *star-compj-var*:  $star\ i\ (x \cdot_{(i+k+1)} y) \leq star\ i\ x \cdot_{(i+k+1)} star\ i\ y$   
**using** *star-compj* **by** *force*

**end**

**end**

## 9 $\omega$ -Quantales

**theory** *Omega-Quantale*

**imports** *Quantales-Converse.Modal-Quantale Omega-Kleene-Algebra*

**begin**

**class** *iquantale* = *complete-lattice* + *imonoid-mult* +  
**assumes** *Sup-distl*:  $x \cdot_i \sqcup Y = (\sqcup y \in Y. x \cdot_i y)$   
**assumes** *Sup-distr*:  $\sqcup X \cdot_i y = (\sqcup x \in X. x \cdot_i y)$

**sublocale** *iquantale*  $\subseteq$  *qiq*: *unital-quantale un i*  $\lambda x y. x \cdot_i y$  - - - - -  
**apply** *unfold-locales* **using** *local.Sup-distr local.Sup-distl* **by** *auto*

**definition** (**in** *iquantale*) *istar* = *qiq.qstar*

**lemma** (**in** *iquantale*) *istar-unfold*: *istar i x* =  $(\sqcup k. \text{mm.power } i x k)$   
**unfolding** *local.qiq.qstar-def local.istar-def* **by** *simp*

**sublocale** *iquantale*  $\subseteq$  *dqisi*: *idiod* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  *icomp un*  
**by** *unfold-locales (simp-all add: local.qiq.sup-distl)*

**sublocale** *iquantale*  $\subseteq$  *dqikai*: *ikleene-algebra* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  *icomp un istar*  
**by** *unfold-locales (simp-all add: local.istar-def local.qiq.uwqlka.star-inductl local.qiq.uqka.star-inductr')*

**class** *idomain-quantale* = *iquantale* + *idom-op* +  
**assumes** *do-absorb*:  $x \leq \text{do } i x \cdot_i x$   
**and** *do-local* [*simp*]:  $\text{do } i (x \cdot_i \text{do } i y) = \text{do } i (x \cdot_i y)$   
**and** *do-add*:  $\text{do } i (x \sqcup y) = \text{do } i x \sqcup \text{do } i y$   
**and** *do-subid*:  $\text{do } i x \leq \text{un } i$   
**and** *do-zero* [*simp*]:  $\text{do } i \perp = \perp$

**sublocale** *idomain-quantale*  $\subseteq$  *dqidq*: *domain-quantale do i un i*  $\lambda x y. x \cdot_i y$  - - -  
- - - - -  
**by** (*unfold-locales, simp-all add: local.do-absorb local.do-add local.do-subid*)

**sublocale** *idomain-quantale*  $\subseteq$  *dqidsi*: *idomain-semiring* ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  *icomp un do*  
**by** (*unfold-locales, simp-all add: local.do-add local.do-subid*)

**class** *icodomain-quantale* = *iquantale* + *icod-op* +  
**assumes** *cd-absorb*:  $x \leq x \cdot_i \text{cd } i x$   
**and** *cd-local* [*simp*]:  $\text{cd } i (\text{cd } i x \cdot_i y) = \text{cd } i (x \cdot_i y)$   
**and** *cd-add*:  $\text{cd } i (x \sqcup y) = \text{cd } i x \sqcup \text{cd } i y$   
**and** *cd-subid*:  $\text{cd } i x \leq \text{un } i$   
**and** *cd-zero* [*simp*]:  $\text{cd } i \perp = \perp$

**sublocale** *icodomain-quantale*  $\subseteq$  *cdqidq*: *codomain-quantale un i*  $\lambda x y. x \cdot_i y$  - - -

```

- - - - - cd i
  by (unfold-locales, simp-all add: local.cd-absorb local.cd-add local.cd-subid)

sublocale icodomain-quantale  $\subseteq$  cdqidcsi: icodomain-semiring cd ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$ 
icomp un
  by (unfold-locales, simp-all add: local.cd-absorb local.cd-add local.cd-subid)

class imodal-quantale = idomain-quantale + icodomain-quantale +
  assumes dc-compat [simp]: do i (cd i x) = cd i x
  and cd-compat [simp]: cd i (do i x) = do i x

sublocale imodal-quantale  $\subseteq$  mqimq: dc-modal-quantale un i  $\lambda x y. x \cdot_i y$  - - - - -
- - - cd i do i
  by unfold-locales simp-all

sublocale imodal-quantale  $\subseteq$  mqimka: imodal-kleene-algebra ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  icomp
un istar cd do
  by unfold-locales simp-all

sublocale imodal-quantale  $\subseteq$  mqidual: imodal-quantale do - - - - -  $\lambda x i y. y$ 
 $\cdot_i x - cd$ 
  by unfold-locales (simp-all add: local.cdqidq.coddual.Sup-distl local.Sup-distl)

class omega-quantale = imodal-quantale +
  assumes interchange:  $i < j \implies (w \cdot_j x) \cdot_i (y \cdot_j z) \leq (w \cdot_i y) \cdot_j (x \cdot_i z)$ 
  and dj-hom:  $i \neq j \implies do j (x \cdot_i y) \leq do j x \cdot_i do j y$ 
  and cj-hom:  $i \neq j \implies cd j (x \cdot_i y) \leq cd j x \cdot_i cd j y$ 
  and djdi:  $i < j \implies do j (do i x) = do i x$ 

class strong-omega-quantale = omega-quantale +
  assumes dj-strong-hom:  $i < j \implies do j (x \cdot_i y) = do j x \cdot_i do j y$ 
  and cj-strong-hom:  $i < j \implies cd j (x \cdot_i y) = cd j x \cdot_i cd j y$ 

sublocale omega-quantale  $\subseteq$  tqqs: omega-semiring cd ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  icomp un do
  by unfold-locales (simp-all add: local.interchange local.dj-hom local.cj-hom lo-
cal.djdi)

sublocale strong-omega-quantale  $\subseteq$  stgqs: strong-omega-semiring cd ( $\sqcup$ ) ( $\leq$ ) ( $<$ )
 $\perp$  icomp un do
  by unfold-locales (simp-all add: local.dj-strong-hom local.cj-strong-hom)

sublocale omega-quantale  $\subseteq$  tqqs: omega-kleene-algebra ( $\sqcup$ ) ( $\leq$ ) ( $<$ )  $\perp$  icomp un
istar cd do ..

sublocale strong-omega-quantale  $\subseteq$  tqqs: strong-omega-kleene-algebra ( $\sqcup$ ) ( $\leq$ ) ( $<$ )
 $\perp$  icomp un istar cd do ..

context omega-quantale
begin

```



**lemma** *istar-aux*:  $i < j \implies \text{mm.power } i \ (x \cdot_j y) \ k \leq \text{mm.power } i \ x \ k \cdot_j \text{mm.power } i \ y \ k$   
**proof** (*induct k*)  
  **case** 0  
  **then show** ?*case*  
    **by** (*simp add: tgqs.uni-compj-eq*)  
**next**  
  **case** (*Suc k*)  
  **fix** *k*  
  **assume**  $i < j$   
  **assume**  $h: i < j \implies \text{mm.power } i \ (x \cdot_j y) \ k \leq \text{mm.power } i \ x \ k \cdot_j \text{mm.power } i \ y \ k$   
  **have**  $\text{mm.power } i \ (x \cdot_j y) \ (\text{Suc } k) = (x \cdot_j y) \cdot_i \text{mm.power } i \ (x \cdot_j y) \ k$   
    **by** *simp*  
  **also have**  $\dots \leq (x \cdot_j y) \cdot_i (\text{mm.power } i \ x \ k \cdot_j \text{mm.power } i \ y \ k)$   
    **by** (*simp add: Suc.premis h local.qiq.psrpq.mult-isol*)  
  **also have**  $\dots \leq (x \cdot_i \text{mm.power } i \ x \ k) \cdot_j (y \cdot_i \text{mm.power } i \ y \ k)$   
    **by** (*simp add: Suc.premis local.tgqs.tgsdual.interchange*)  
  **also have**  $\dots = \text{mm.power } i \ x \ (\text{Suc } k) \cdot_j \text{mm.power } i \ y \ (\text{Suc } k)$   
    **by** *simp*  
  **finally show**  $\text{mm.power } i \ (x \cdot_j y) \ (\text{Suc } k) \leq \text{mm.power } i \ x \ (\text{Suc } k) \cdot_j \text{mm.power } i \ y \ (\text{Suc } k)$ .  
**qed**

**lemma** *istar-oplax*:  $i < j \implies \text{istar } i \ (x \cdot_j y) \leq \text{istar } i \ x \cdot_j \text{istar } i \ y$   
  **by** (*simp add: local.tgqs.star-compj*)

**lemma** *istar-distli*:  $i < j \implies x \cdot_i (\text{istar } j \ y) = (\bigsqcup k. x \cdot_i (\text{mm.power } j \ y \ k))$   
  **by** (*simp add: image-image local.qiq.Sup-distl local.istar-unfold*)

**lemma** *istar-distri*:  $i < j \implies (\text{istar } j \ x) \cdot_i y = (\bigsqcup k. \text{mm.power } j \ x \ k \cdot_i y)$   
  **by** (*simp add: image-image local.qiq.Sup-distr local.istar-unfold*)

**lemma** *istar-distlj*:  $i < j \implies x \cdot_j (\text{istar } i \ y) = (\bigsqcup k. x \cdot_j (\text{mm.power } i \ y \ k))$   
  **by** (*simp add: image-image local.Sup-distl local.istar-unfold*)

**lemma** *istar-distrj*:  $i < j \implies (\text{istar } i \ x) \cdot_j y = (\bigsqcup k. \text{mm.power } i \ x \ k \cdot_j y)$   
  **by** (*simp add: image-image local.qiq.Sup-distr local.istar-unfold*)

**lemma** *istar-laxl-aux-var*:  $i < j \implies \text{do } i \ x \cdot_i \text{mm.power } j \ y \ k \leq \text{mm.power } j \ (\text{do } i \ x \cdot_i y) \ k$

**proof** (*induct k*)  
  **case** 0  
  **assume**  $i < j$   
  **have**  $\text{do } i \ x \cdot_i \text{un } j = \text{do } j \ (\text{do } i \ x) \cdot_i \text{un } j$   
    **by** (*simp add: 0 local.djdi*)  
  **also have**  $\dots \leq \text{un } j \cdot_i \text{un } j$   
    **by** (*simp add: local.qiq.nsrnq.mult-isor*)  
  **finally have**  $\text{do } i \ x \cdot_i \text{un } j \leq \text{un } j$

by (*simp add: local.dqidq.dqmsr.dom-subid-aux2*)  
 thus  $do\ i\ x \cdot_i\ mm.power\ j\ y\ 0 \leq mm.power\ j\ (do\ i\ x \cdot_i\ y)\ 0$   
 by *simp*  
**next**  
 case (*Suc k*)  
 fix *k*  
 assume  $i < j$   
 assume  $h: i < j \implies do\ i\ x \cdot_i\ mm.power\ j\ y\ k \leq mm.power\ j\ (do\ i\ x \cdot_i\ y)\ k$   
 have  $do\ i\ x \cdot_i\ mm.power\ j\ y\ (Suc\ k) = do\ i\ x \cdot_i\ (y \cdot_j\ mm.power\ j\ y\ k)$   
 by *simp*  
 also have  $\dots = (do\ i\ x \cdot_j\ do\ i\ x) \cdot_i\ (y \cdot_j\ mm.power\ j\ y\ k)$   
 using *Suc.premis less-imp-add-positive* by *fastforce*  
 also have  $\dots \leq (do\ i\ x \cdot_i\ y) \cdot_j\ (do\ i\ x \cdot_i\ mm.power\ j\ y\ k)$   
 by (*simp add: Suc.premis local.interchange*)  
 also have  $\dots \leq (do\ i\ x \cdot_i\ y) \cdot_j\ mm.power\ j\ (do\ i\ x \cdot_i\ y)\ k$   
 by (*simp add: Suc.premis h local.qiq.psrpq.mult-isol*)  
 finally show  $do\ i\ x \cdot_i\ mm.power\ j\ y\ (Suc\ k) \leq mm.power\ j\ (do\ i\ x \cdot_i\ y)\ (Suc\ k)$   
 by *simp*  
**qed**

**lemma** *istar-laxl-var*:

assumes  $i < j$   
 shows  $do\ i\ x \cdot_i\ istar\ j\ y \leq istar\ j\ (do\ i\ x \cdot_i\ y)$   
**proof** –  
 have  $do\ i\ x \cdot_i\ istar\ j\ y = (\bigsqcup k. do\ i\ x \cdot_i\ mm.power\ j\ y\ k)$   
 by (*simp add: image-image local.Sup-distl local.istar-unfold*)  
 also have  $\dots \leq (\bigsqcup k. mm.power\ j\ (do\ i\ x \cdot_i\ y)\ k)$   
 by (*simp add: asms local.SUP-mono' local.istar-laxl-aux-var*)  
 finally show *?thesis*  
 using *local.istar-unfold* by *auto*  
**qed**

**lemma** *istar-laxl-var2*:  $do\ i\ x \cdot_i\ istar\ (i + k + 1)\ y \leq istar\ (i + k + 1)\ (do\ i\ x \cdot_i\ y)$   
 using *istar-laxl-var* by *force*

**lemma** *istar-laxr-aux-var*:  $i < j \implies mm.power\ j\ x\ k \cdot_i\ cd\ i\ y \leq mm.power\ j\ (x \cdot_i\ cd\ i\ y)\ k$

**proof** (*induct k*)  
 case 0 show *?case*  
 by (*simp add: local.cdqidq.coddual.dqmsr.dom-subid-aux2*)  
**next**  
 case (*Suc k*)  
 assume  $h0: i < j$   
 fix *k*  
 assume  $h: i < j \implies mm.power\ j\ x\ k \cdot_i\ cd\ i\ y \leq mm.power\ j\ (x \cdot_i\ cd\ i\ y)\ k$   
 have  $mm.power\ j\ x\ (Suc\ k) \cdot_i\ cd\ i\ y = (x \cdot_j\ mm.power\ j\ x\ k) \cdot_i\ (cd\ i\ y \cdot_j\ cd\ i\ y)$   
 using *h0 less-imp-add-positive* by *fastforce*  
 also have  $\dots \leq (x \cdot_i\ cd\ i\ y) \cdot_j\ (mm.power\ j\ x\ k \cdot_i\ cd\ i\ y)$

by (simp add: h0 local.tgqs.tgsdual.interchange)  
 finally show  $mm.power\ j\ x\ (Suc\ k)\ \cdot_i\ cd\ i\ y \leq mm.power\ j\ (x\ \cdot_i\ cd\ i\ y)\ (Suc\ k)$   
 by (simp add: h h0 local.qiq.h-w2 local.qiq.psrpq.mult-isol)  
 qed

lemma *istar-laxr-var*:

assumes  $i < j$   
 shows  $istar\ j\ x\ \cdot_i\ cd\ i\ y \leq istar\ j\ (x\ \cdot_i\ cd\ i\ y)$

proof –

have  $istar\ j\ x\ \cdot_i\ cd\ i\ y = (\bigsqcup k. mm.power\ j\ x\ k\ \cdot_i\ cd\ i\ y)$   
 using *assms istar-distri* by *presburger*  
 also have  $\dots \leq (\bigsqcup k. mm.power\ j\ (x\ \cdot_i\ cd\ i\ y)\ k)$   
 by (simp add: *assms local.SUP-mono' local.istar-laxr-aux-var*)  
 finally show *?thesis*  
 by (simp add: *local.istar-unfold*)

qed

lemma *istar-laxr-var2*:  $istar\ (i + k + 1)\ x\ \cdot_i\ cd\ i\ y \leq istar\ (i + k + 1)\ (x\ \cdot_i\ cd\ i\ y)$

using *istar-laxr-var* by *force*

lemma *istar-prop*:

assumes  $i < j$   
 shows  $istar\ j\ x\ \cdot_i\ istar\ j\ y = (\bigsqcup k\ l. mm.power\ j\ x\ k\ \cdot_i\ mm.power\ j\ y\ l)$

proof –

have  $istar\ j\ x\ \cdot_i\ istar\ j\ y = istar\ j\ x\ \cdot_i\ (\bigsqcup k. mm.power\ j\ y\ k)$   
 using *local.istar-unfold* by *auto*  
 also have  $\dots = (\bigsqcup l. istar\ j\ x\ \cdot_i\ mm.power\ j\ y\ l)$   
 by (simp add: *image-image local.Sup-distl*)  
 also have  $\dots = (\bigsqcup l. (\bigsqcup k. mm.power\ j\ x\ k)\ \cdot_i\ mm.power\ j\ y\ l)$   
 unfolding *istar-def qiq.qstar-def* by (simp add: *full-SetCompr-eq*)  
 also have  $\dots = (\bigsqcup l. (\bigsqcup k. mm.power\ j\ x\ k\ \cdot_i\ mm.power\ j\ y\ l))$   
 using *assms istar-distri local.istar-unfold* by *auto*  
 also have  $\dots = (\bigsqcup k\ l. mm.power\ j\ x\ k\ \cdot_i\ mm.power\ j\ y\ l)$   
 using *local.SUP-commute* by *force*  
 finally show *?thesis*.

qed

end

context *strong-omega-quantale*

begin

lemma *istar-laxl-aux*:  $i < j \implies do\ j\ x\ \cdot_i\ mm.power\ j\ y\ k \leq mm.power\ j\ (do\ j\ x\ \cdot_i\ y)\ k$

proof (induct *k*)

case 0

assume  $i < j$

have  $do\ i\ x\ \cdot_i\ un\ j \leq un\ j\ \cdot_i\ un\ j$

**using**  $0$  *local.dqidq.dqmsr.dom-subid-aux2 local.stgqs.stgsdual.idj-compi-eq* **by**  
*force*  
**thus**  $do\ j\ x \cdot_i\ mm.power\ j\ y\ 0 \leq mm.power\ j\ (do\ j\ x \cdot_i\ y)\ 0$   
**by** (*metis 0 local.do-subid local.mm.power.power-0 local.qiq.nsrnq.mult-isol local.stgqs.stgsdual.idj-compi-eq*)  
**next**  
**case** (*Suc k*)  
**assume**  $i < j$   
**fix**  $k$   
**assume**  $h: i < j \implies do\ j\ x \cdot_i\ mm.power\ j\ y\ k \leq mm.power\ j\ (do\ j\ x \cdot_i\ y)\ k$   
**have**  $do\ j\ x \cdot_i\ mm.power\ j\ y\ (Suc\ k) = do\ j\ x \cdot_i\ (y \cdot_j\ mm.power\ j\ y\ k)$   
**by** *simp*  
**also have**  $\dots = (do\ j\ x \cdot_j\ do\ j\ x) \cdot_i\ (y \cdot_j\ mm.power\ j\ y\ k)$   
**by** *simp*  
**also have**  $\dots \leq (do\ j\ x \cdot_i\ y) \cdot_j\ (do\ j\ x \cdot_i\ mm.power\ j\ y\ k)$   
**using** *Suc.premis local.interchange* **by** *blast*  
**also have**  $\dots \leq (do\ j\ x \cdot_i\ y) \cdot_j\ mm.power\ j\ (do\ j\ x \cdot_i\ y)\ k$   
**by** (*simp add: Suc.premis h local.qiq.psrpq.mult-isol*)  
**finally show**  $do\ j\ x \cdot_i\ mm.power\ j\ y\ (Suc\ k) \leq mm.power\ j\ (do\ j\ x \cdot_i\ y)\ (Suc\ k)$   
**by** *simp*  
**qed**

**lemma** *istar-laxl*:  
**assumes**  $i < j$   
**shows**  $do\ j\ x \cdot_i\ istar\ j\ y \leq istar\ j\ (do\ j\ x \cdot_i\ y)$   
**proof** –  
**have**  $do\ j\ x \cdot_i\ istar\ j\ y = (\bigsqcup k. do\ j\ x \cdot_i\ mm.power\ j\ y\ k)$   
**using** *assms local.istar-distli* **by** *force*  
**also have**  $\dots \leq (\bigsqcup k. mm.power\ j\ (do\ j\ x \cdot_i\ y)\ k)$   
**by** (*simp add: assms istar-laxl-aux local.SUP-mono'*)  
**finally show** *?thesis*  
**by** (*simp add: local.istar-unfold*)  
**qed**

**lemma** *istar-laxr-aux*:  $i < j \implies mm.power\ j\ x\ k \cdot_i\ cd\ j\ y \leq mm.power\ j\ (x \cdot_i\ cd\ j\ y)\ k$   
**proof** (*induct k*)  
**case**  $0$  **thus** *?case*  
**by** (*metis local.cd-subid local.mm.power.power-0 local.qiq.psrpq.mult-isol local.stgqs.stgsdual.idj-compi-eq*)  
**next**  
**case** (*Suc k*)  
**assume**  $i < j$   
**fix**  $k$   
**assume**  $h: i < j \implies mm.power\ j\ x\ k \cdot_i\ cd\ j\ y \leq mm.power\ j\ (x \cdot_i\ cd\ j\ y)\ k$   
**have**  $mm.power\ j\ x\ (Suc\ k) \cdot_i\ cd\ j\ y = (x \cdot_j\ mm.power\ j\ x\ k) \cdot_i\ cd\ j\ y$   
**by** *simp*  
**also have**  $\dots = (x \cdot_j\ mm.power\ j\ x\ k) \cdot_i\ (cd\ j\ y \cdot_j\ cd\ j\ y)$   
**by** *simp*

**also have**  $\dots \leq (x \cdot_i \text{cd } j \ y) \cdot_j (\text{mm.power } j \ x \ k \cdot_i \text{cd } j \ y)$   
**using** *Suc.prem*s *local.interchange* **by** *blast*  
**also have**  $\dots \leq (x \cdot_i \text{cd } j \ y) \cdot_j \text{mm.power } j \ (x \cdot_i \text{cd } j \ y) \ k$   
**by** (*simp add: Suc.prem*s *h local.qiq.psrpq.mult-isol*)  
**finally show**  $\text{mm.power } j \ x \ (\text{Suc } k) \cdot_i \text{cd } j \ y \leq \text{mm.power } j \ (x \cdot_i \text{cd } j \ y) \ (\text{Suc } k)$   
**by** *simp*  
**qed**

**lemma** *iqstar-laxr*:  
**assumes**  $i < j$   
**shows**  $\text{istar } j \ x \cdot_i \text{cd } j \ y \leq \text{istar } j \ (x \cdot_i \text{cd } j \ y)$   
**proof** –  
**have**  $\text{istar } j \ x \cdot_i \text{cd } j \ y = (\bigsqcup k. \text{mm.power } j \ x \ k \cdot_i \text{cd } j \ y)$   
**using** *assms local.istar-distri* **by** *force*  
**also have**  $\dots \leq (\bigsqcup k. \text{mm.power } j \ (x \cdot_i \text{cd } j \ y) \ k)$   
**by** (*simp add: assms istar-laxr-aux local.SUP-mono*<sup>^</sup>)  
**finally show** *?thesis*  
**by** (*simp add: local.istar-unfold*)  
**qed**

**lemma** *qstar1-aux*:  $i < j \implies \text{mm.power } j \ x \ k \cdot_i \text{mm.power } j \ y \ k \leq \text{mm.power } j \ (x \cdot_i y) \ k$   
**proof** (*induct k*)  
**case**  $0$   
**then show** *?case*  
**using** *local.stgqs.stgsdual.idj-compi-eq* **by** *force*  
**next**  
**case** (*Suc k*)  
**assume**  $i < j$   
**fix**  $k$   
**assume**  $h: i < j \implies \text{mm.power } j \ x \ k \cdot_i \text{mm.power } j \ y \ k \leq \text{mm.power } j \ (x \cdot_i y) \ k$   
**have**  $\text{mm.power } j \ x \ (\text{Suc } k) \cdot_i \text{mm.power } j \ y \ (\text{Suc } k) = (x \cdot_j \text{mm.power } j \ x \ k) \cdot_i (y \cdot_j \text{mm.power } j \ y \ k)$   
**by** *simp*  
**also have**  $\dots \leq (x \cdot_i y) \cdot_j (\text{mm.power } j \ x \ k \cdot_i \text{mm.power } j \ y \ k)$   
**using** *Suc.prem*s *local.interchange* **by** *force*  
**also have**  $\dots \leq (x \cdot_i y) \cdot_j \text{mm.power } j \ (x \cdot_i y) \ k$   
**by** (*simp add: Suc.prem*s *h local.qiq.psrpq.mult-isol*)  
**also have**  $\dots = \text{mm.power } j \ (x \cdot_i y) \ (\text{Suc } k)$   
**by** *simp*  
**finally show**  $\text{mm.power } j \ x \ (\text{Suc } k) \cdot_i \text{mm.power } j \ y \ (\text{Suc } k) \leq \text{mm.power } j \ (x \cdot_i y) \ (\text{Suc } k)$ .  
**qed**  
**end**  
**end**

## 10 Lifting $\omega$ -catoids to powerset $\omega$ -quantales

```
theory Omega-Catoid-Lifting
  imports Omega-Catoid Omega-Quantale

begin

instantiation set :: (local-omega-catoid) omega-quantale

begin

definition do-set :: nat  $\Rightarrow$  'a set  $\Rightarrow$  'a set where
  do i X = Srci i X

definition cd-set :: nat  $\Rightarrow$  'a set  $\Rightarrow$  'a set where
  cd i X = Tgti i X

definition icomp-set :: 'a set  $\Rightarrow$  nat  $\Rightarrow$  'a set  $\Rightarrow$  'a set where
  X  $\cdot_i$  Y = X  $\star_i$  Y

definition un-set :: nat  $\Rightarrow$  'a set where
  un i = srefix i

instance
  apply intro-classes
  unfolding icomp-set-def do-set-def cd-set-def un-set-def iconv-prop
    apply (simp add: ims.conv-assoc)
  using stimm.stopp.stopp.conv-uns apply blast
    apply (metis stimm.stopp.stopp.conv-unt stimm.stopp.stopp.st-fix-set)
    apply (simp add: ims.conv-distl)
    apply (simp add: multimagma.conv-distr)
    apply force+
    apply (metis iconv-prop interchange-lift)
    apply (metis iconv-prop omega-st-multimagma-class.Srcj-hom)
    apply (metis iconv-prop omega-st-multimagma-class.Tgtj-hom)
  by (simp add: olropp.TjTi)

end

end
```

## References

- [1] C. Calk, E. Goubault, P. Malbos, and G. Struth. Algebraic coherent confluence and higher globular Kleene algebras. *Log. Methods Comput. Sci.*, 18(4), 2022.

- [2] C. Calk, P. Malbos, D. Pous, and G. Struth. Higher catoids, higher quantales and their correspondences. *CoRR*, abs/2307.09253, 2023.
- [3] C. Calk and G. Struth. Modal quantales, involutive quantales, Dedekind quantales. *Archive of Formal Proofs*, July 2023. [https://isa-afp.org/entries/Quantales\\_Converse.html](https://isa-afp.org/entries/Quantales_Converse.html), Formal proof development.
- [4] G. Struth. Quantales. *Archive of Formal Proofs*, December 2018. <https://isa-afp.org/entries/Quantales.html>, Formal proof development.
- [5] G. Struth. Catoids, categories, groupoids. *Archive of Formal Proofs*, August 2023. <https://isa-afp.org/entries/Catoids.html>, Formal proof development.