

Inner Structure, Determinism and Modal Algebra of Multirelations

Walter Guttmann and Georg Struth

March 17, 2025

Abstract

Binary multirelations form a model of alternating nondeterminism useful for analysing games, interactions of computing systems with their environments or abstract interpretations of probabilistic programs. We investigate this alternating structure in a relational language based on power allegories extended with specific operations on multirelations. We develop algebras of modal operators over multirelations, related to concurrent dynamic logics, in this language.

Contents

1 Properties of Binary Relations	3
1.1 Univalence, totality, determinism, and related properties	3
1.2 Inverse image and the diagonal and graph functors	5
1.3 Relational domain, codomain and modalities	6
1.4 Residuation	8
1.5 Symmetric quotient	11
2 Properties of Power Allegories	11
2.1 Power transpose, epsilon, epsilonoff	12
2.2 Relational image functor	13
2.3 Unit and multiplication of powerset monad	14
2.4 Subset relation	17
2.5 Complementation relation	18
2.6 Kleisli lifting and Kleisli composition	19
2.7 Relational box	20
3 Basic Properties of Multirelations	22
3.1 Peleg composition, parallel composition (inner union) and units	23
3.2 Tests	27
3.3 Parallel subidentities	30
3.4 Domain	35

3.5	Vectors	41
3.6	Up-closure and Parikh composition	42
3.7	Nonterminal and terminal multirelations	44
3.8	Powers	51
3.9	Star	52
3.10	Omega	55
4	Multirelational Properties of Power Allegories	56
4.1	Peleg lifting	56
4.2	Fusion and fission	60
4.3	Galois connections for multirelations	60
4.4	Properties of alpha, fission and fusion	63
4.5	Properties of fusion, fission, nu and tau	63
4.6	Box and diamond	64
5	Inner Structure	65
5.1	Inner union, inner intersection and inner complement	65
5.2	Dual	71
5.3	Co-composition	72
5.4	Inner order	75
5.5	Up-closure, down-closure and convex-closure	75
6	Powerdomain Preorders	86
6.1	Functional properties of multirelations	95
6.2	Equivalences induced by powerdomain preorders	105
6.3	Further results for convex-closure	114
7	Fusion and Fission	119
7.1	Atoms and co-atoms	119
7.2	Inner-functional properties	120
7.3	Fusion	123
7.4	Fission	129
7.5	Co-fusion and co-fission	133
8	Modalities	138
8.1	Tests	138
8.2	Domain and antidomain	155
8.3	Left residual	159
8.4	Modal operations	168
8.5	Goldblatt's axioms without star	179
8.6	Goldblatt's axioms with star	189
8.7	Propositional Hoare logic	194

The theories formally verify results in [3, 1, 2]. See these papers for further details and related work.

The basic algebra of homogeneous binary multirelations is formalised in [4]. The present theories consider heterogeneous binary multirelations, which may have different source and target sets. While homogeneous multirelations arise as a special case where source and target sets coincide, we do not attempt to generalise the algebras of [4] to the heterogeneous case but study new concepts instead. Thus the present theories and [4] are complementary. A unification of the two approaches based on category theory is possible future work.

Algebraic structures for multirelations with Parikh composition are formalised in [5].

1 Properties of Binary Relations

theory Relational-Properties

imports Main

begin

This is a general-purpose theory for enrichments of Rel, which is still quite basic, but helpful for developing properties of multirelations.

notation relcomp (**infixl** $\langle;\rangle$ 75)
notation converse ($\langle\cdot,\cdot\rangle$ [1000] 999)

type-synonym ('a,'b) rel = ('a \times 'b) set

lemma modular-law: $R ; S \cap T \subseteq (R \cap T ; S^\sim) ; S$
by blast

lemma compl-conv: $-(R^\sim) = (-R)^\sim$
by fastforce

definition top :: ('a,'b) rel **where**
 $top = \{(a,b) \mid a \ b. \ True\}$

abbreviation neg R \equiv Id \cap $-R$

1.1 Univalence, totality, determinism, and related properties

definition univalent :: ('a,'b) rel \Rightarrow bool **where**
 $univalent R = (R^\sim ; R \subseteq Id)$

```

definition total :: ('a,'b) rel  $\Rightarrow$  bool where
  total R = (Id  $\subseteq$  R ; R $^{\sim}$ )

definition injective :: ('a,'b) rel  $\Rightarrow$  bool where
  injective R = (R ; R $^{\sim}$   $\subseteq$  Id)

definition surjective :: ('a,'b) rel  $\Rightarrow$  bool where
  surjective R = (Id  $\subseteq$  R $^{\sim}$  ; R)

definition deterministic :: ('a,'b) rel  $\Rightarrow$  bool where
  deterministic R = (univalent R  $\wedge$  total R)

definition bijective :: ('a,'b) rel  $\Rightarrow$  bool where
  bijective R = (injective R  $\wedge$  surjective R)

lemma univalent-set: univalent R = ( $\forall$  a b c. (a,b)  $\in$  R  $\wedge$  (a,c)  $\in$  R  $\longrightarrow$  b = c)
  unfolding univalent-def converse-unfold Id-def by force

  Univalent relations feature as single-valued relations in Main.

lemma univ-single-valued: univalent R = single-valued R
  unfolding univalent-set single-valued-def by auto

lemma total-set: total R = ( $\forall$  a.  $\exists$  b. (a,b)  $\in$  R)
  unfolding total-def converse-unfold Id-def by force

lemma total-var: total R = (R ; top = top)
  unfolding total-set top-def by force

lemma deterministic-set: deterministic R = ( $\forall$  a .  $\exists$ !B . (a,B)  $\in$  R)
  unfolding deterministic-def univalent-set total-set by force

lemma deterministic-var1: deterministic R = (R ; -Id = -R)
  unfolding deterministic-set Id-def by force

lemma deterministic-var2: deterministic R = ( $\forall$  S. R ; -S = -(R ; S))
  unfolding deterministic-set by (standard, force, blast)

lemma inj-univ: injective R = univalent (R $^{\sim}$ )
  by (simp add: injective-def univalent-def)

lemma injective-set: injective S = ( $\forall$  a b c. (a,c)  $\in$  S  $\wedge$  (b,c)  $\in$  S  $\longrightarrow$  a = b)
  by (meson converseD converseI inj-univ univalent-set)

lemma surj-tot: surjective R = total (R $^{\sim}$ )
  by (simp add: surjective-def total-def)

lemma surjective-set: surjective S = ( $\forall$  b.  $\exists$  a. (a,b)  $\in$  S)
  by (simp add: surj-tot total-set)

```

```

lemma surj-var: surjective  $R = (R^\sim ; \text{top} = \text{top})$ 
  by (simp add: surj-tot total-var)

lemma bij-det: bijective  $R = \text{deterministic } (R^\sim)$ 
  by (simp add: bijective-def deterministic-def inj-univ surj-tot)

lemma univ-relcomp: univalent  $R \implies \text{univalent } S \implies \text{univalent } (R ; S)$ 
  by (simp add: single-valued-relcomp univ-single-valued)

lemma tot-relcomp: total  $R \implies \text{total } S \implies \text{total } (R ; S)$ 
  by (meson relcomp.simps total-set)

lemma det-relcomp: deterministic  $R \implies \text{deterministic } S \implies \text{deterministic } (R ; S)$ 
  by (simp add: deterministic-def tot-relcomp univ-relcomp)

lemma inj-relcomp: injective  $R \implies \text{injective } S \implies \text{injective } (R ; S)$ 
  by (simp add: converse-relcomp inj-univ univ-relcomp)

lemma surj-relcomp: surjective  $R \implies \text{surjective } S \implies \text{surjective } (R ; S)$ 
  by (simp add: converse-relcomp surj-tot tot-relcomp)

lemma bij-relcomp: bijective  $R \implies \text{bijective } S \implies \text{bijective } (R ; S)$ 
  by (simp add: bijective-def inj-relcomp surj-relcomp)

lemma det-Id: deterministic Id
  by (simp add: deterministic-var1)

lemma bij-Id: bijective Id
  by (simp add: bij-det det-Id)

lemma tot-top: total top
  by (simp add: top-def total-set)

lemma tot-surj: surjective top
  by (simp add: surjective-set top-def)

lemma det-meet-distl: univalent  $R \implies R ; (S \cap T) = R ; S \cap R ; T$ 
  unfolding univalent-set relcomp-def relcompp-apply by force

lemma inj-meet-distr: injective  $T \implies (R \cap S) ; T = R ; T \cap S ; T$ 
  unfolding injective-def converse-def Id-def relcomp.simps by force

lemma univ-modular: univalent  $S \implies R ; S \cap T = (R \cap T ; S^\sim) ; S$ 
  unfolding univalent-set converse-unfold relcomp.simps by force

```

1.2 Inverse image and the diagonal and graph functors

definition Invim :: $('a, 'b) \text{ rel} \Rightarrow 'b \text{ set} \Rightarrow 'a \text{ set}$ **where**

Invim $R = \text{Image} (R^\sim)$

definition $\Delta :: 'a \text{ set} \Rightarrow ('a,'a) \text{ rel } (\langle\Delta\rangle)$ **where**
 $\Delta P = \{(p,p) \mid p. p \in P\}$

definition $\text{Grph} :: ('a \Rightarrow 'b) \Rightarrow ('a,'b) \text{ rel}$ **where**
 $\text{Grph } f = \{(x,y). y = f x\}$

lemma $\text{Image-Grph} [\text{simp}]: \text{Image} \circ \text{Grph} = \text{image}$
unfolding $\text{Image-def Grph-def image-def by auto}$

1.3 Relational domain, codomain and modalities

Domain and codomain (range) maps have been defined in Main, but they return sets instead of relations.

definition $\text{dom} :: ('a,'b) \text{ rel} \Rightarrow ('a,'a) \text{ rel}$ **where**
 $\text{dom } R = \text{Id} \cap R ; R^\sim$

definition $\text{cod} :: ('a,'b) \text{ rel} \Rightarrow ('b,'b) \text{ rel}$ **where**
 $\text{cod } R = \text{dom} (R^\sim)$

definition $\text{rel-fdia} :: ('a,'b) \text{ rel} \Rightarrow ('b,'b) \text{ rel} \Rightarrow ('a,'a) \text{ rel } (\langle(|\cdot|)\rangle [61,81] 82)$
where
 $|R\rangle Q = \text{dom} (R ; \text{dom } Q)$

definition $\text{rel-bdia} :: ('a,'b) \text{ rel} \Rightarrow ('a,'a) \text{ rel} \Rightarrow ('b,'b) \text{ rel } (\langle(\langle|\cdot|\rangle)\rangle [61,81] 82)$
where
 $\text{rel-bdia } R = \text{rel-fdia} (R^\sim)$

definition $\text{rel-fbox} :: ('a,'b) \text{ rel} \Rightarrow ('b,'b) \text{ rel} \Rightarrow ('a,'a) \text{ rel } (\langle(|\cdot|)\rangle [61,81] 82)$
where
 $|R]Q = \text{neg} (\text{dom} (R ; \text{neg} (\text{dom } Q)))$

definition $\text{rel-bbox} :: ('a,'b) \text{ rel} \Rightarrow ('a,'a) \text{ rel} \Rightarrow ('b,'b) \text{ rel } (\langle(|\cdot|)\rangle [61,81] 82)$
where
 $\text{rel-bbox } R = \text{rel-fbox} (R^\sim)$

lemma $\text{rel-bdia-def-var}: \text{rel-bdia} = \text{rel-fdia} \circ \text{converse}$
unfolding $\text{rel-fdia-def fun-eq-iff comp-def rel-bdia-def by simp}$

lemma $\text{dom-set}: \text{dom } R = \{(a,a) \mid a. \exists b. (a,b) \in R\}$
unfolding $\text{dom-def Id-def converse-unfold relcomp-unfold by force}$

lemma $\text{dom-Domain}: \text{dom} = \Delta \circ \text{Domain}$
unfolding $\text{fun-eq-iff dom-set Delta-def Domain-def by clarsimp blast}$

lemma $\text{cod-set}: \text{cod } R = \{(b,b) \mid b. \exists a. (a,b) \in R\}$
by (*smt (verit) Collect-cong cod-def converseD converseI dom-set*)

lemma *cod-Range*: $\text{cod} = \Delta \circ \text{Range}$
unfolding *fun-eq-iff cod-set Delta-def Range-def* **by** *clarsimp blast*

lemma *rel-fdia-set*: $|R\rangle Q = \{(a,a) \mid a. \exists b. (a,b) \in R \wedge (b,b) \in \text{dom } Q\}$
unfolding *rel-fdia-def dom-set relcomp-unfold* **by** *force*

lemma *rel-bdia-set*: $\langle R | P = \{(b,b) \mid b. \exists a. (a,b) \in R \wedge (a,a) \in \text{dom } P\}$
by (*smt (verit, best)*) *Collect-cong converseD converseI rel-bdia-def rel-fdia-set*)

lemma *rel-fbox-set*: $|R| Q = \{(a,a) \mid a. \forall b. (a,b) \in R \rightarrow (b,b) \in \text{dom } Q\}$
unfolding *rel-fbox-def dom-set relcomp-unfold* **by** *force*

lemma *rel-bbox-set*: $[R] P = \{(b,b) \mid b. \forall a. (a,b) \in R \rightarrow (a,a) \in \text{dom } P\}$
by (*smt (verit)*) *Collect-cong converseD converseI rel-bbox-def rel-fbox-set*)

lemma *dom-alt-def*: $\text{dom } R = \text{Id} \cap R ; \text{top}$
unfolding *dom-set top-def Id-def* **by** *force*

lemma *dom-gla*: $(\text{dom } R \subseteq \text{Id} \cap S) = (R \subseteq (\text{Id} \cap S) ; R)$
unfolding *dom-set Id-def relcomp-unfold* **by** *blast*

lemma *dom-gla-top*: $(\text{dom } R \subseteq \text{Id} \cap S) = (R \subseteq (\text{Id} \cap S) ; \text{top})$
unfolding *dom-set Id-def top-def relcomp-unfold* **by** *blast*

lemma *dom-subid*: $(\text{dom } R = R) = (R = \text{Id} \cap R)$
by (*metis (no-types, lifting)* *Id-O-R R-O-Id dom-alt-def dom-gla-top equalityD1 inf.absorb-iff2 inf.commute inf.idem le-inf-iff relcomp-mono*)

lemma *dom-cod*: $(\text{dom } R = R) = (\text{cod } R = R)$
by (*metis dom-def cod-def converse-Id converse-Int converse-converse converse-relcomp*)

lemma *dom-top*: $R ; \text{top} = \text{dom } R ; \text{top}$
unfolding *dom-set top-def* **by** *blast*

lemma *top-dom*: $\text{dom } R = \text{dom } (R ; \text{top})$
unfolding *dom-def top-def* **by** *blast*

lemma *cod-top*: $\text{cod } R = \text{Id} \cap \text{top} ; R$
by (*metis dom-def cod-def converse-Id converse-Int converse-converse converse-relcomp dom-alt-def surj-var tot-surj*)

lemma *dom-conv [simp]*: $(\text{dom } R)^\sim = \text{dom } R$
by (*simp add: dom-def converse-Int converse-relcomp*)

lemma *total-dom*: $\text{total } R = (\text{dom } R = \text{Id})$
by (*metis dom-def inf.orderE inf-le2 total-def*)

lemma *surj-cod*: $\text{surjective } R = (\text{cod } R = \text{Id})$

```

by (simp add: cod-def surj-tot total-dom)

lemma fdia-demod: (|R⟩ P ⊆ dom Q) = (R ; dom P ⊆ dom Q ; R)
  unfolding rel-fdia-set dom-set relcomp-unfold by force

lemma bbox-demod: (dom P ⊆ [R] Q) = (R ; dom P ⊆ dom Q ; R)
  unfolding rel-bbox-set dom-def by force

lemma bdia-demod: ⟨R| P ⊆ dom Q) = (dom P ; R ⊆ R ; dom Q)
proof -
  have ⟨R| P ⊆ dom Q) = (|R⁻⟩ P ⊆ dom Q)
    by (simp add: rel-bdia-def)
  also have ... = ((R⁻) ; dom P ⊆ dom Q ; (R⁻))
    by (simp add: fdia-demod)
  also have ... = (((dom P ; R)⁻) ⊆ ((R ; dom Q)⁻))
    by (simp add: converse-relcomp)
  also have ... = (dom P ; R ⊆ R ; dom Q)
    by simp
  finally show ?thesis.
qed

lemma fbox-demod: (dom P ⊆ |R] Q) = (dom P ; R ⊆ R ; dom Q)
  unfolding rel-fbox-set dom-set relcomp-unfold by force

lemma fdia-demod-top: (|R⟩ P ⊆ dom Q) = (R ; dom P ; top ⊆ dom Q ; top)
  by (metis dom-def dom-gla-top rel-fdia-def top-dom)

lemma bbox-demod-top: (dom P ⊆ [R] Q) = (R ; dom P ; top ⊆ dom Q ; top)
  unfolding rel-bbox-def rel-fbox-def dom-def top-def by force

lemma fdia-bbox-galois: (|R⟩ P ⊆ dom Q) = (dom P ⊆ [R] Q)
  by (meson bbox-demod-top fdia-demod-top)

lemma bdia-fbox-galois: ⟨R| P ⊆ dom Q) = (dom P ⊆ |R] Q)
  by (simp add: fdia-bbox-galois rel-bbox-def rel-bdia-def)

lemma fdia-bdia-conjugation: (|R⟩ P ⊆ neg (dom Q)) = (⟨R| Q ⊆ neg (dom P))
  unfolding rel-fdia-set rel-bdia-set dom-set by force

lemma bfox-bbox-conjugation: (neg (dom Q) ⊆ |R] P) = (neg (dom P) ⊆ [R] Q)
  unfolding rel-fbox-set rel-bbox-set dom-set by clarsimp blast

```

1.4 Residuation

```

definition lres :: ('a,'c) rel ⇒ ('b,'c) rel ⇒ ('a,'b) rel (infixl ‹//› 75)
  where R // S = {(a,b). ∀ c. (b,c) ∈ S → (a,c) ∈ R}

definition rres :: ('c,'a) rel ⇒ ('c,'b) rel ⇒ ('a,'b) rel (infixl ‹\› 75)
  where R \ S = {(b,a). ∀ c. (c,b) ∈ R → (c,a) ∈ S}

```

```

lemma rres-lres-conv:  $R \setminus S = (S^\sim // R^\sim)^\sim$ 
  unfolding rres-def lres-def by clarsimp fastforce

lemma lres-galois:  $(R ; S \subseteq T) = (R \subseteq T // S)$ 
  unfolding lres-def by blast

lemma rres-galois:  $(R ; S \subseteq T) = (S \subseteq R \setminus T)$ 
  by (metis converse-converse converse-mono converse-relcomp lres-galois
rres-lres-conv)

lemma lres-compl:  $R // S = -(-R ; S^\sim)$ 
  unfolding lres-def converse-unfold by force

lemma rres-compl:  $R \setminus S = -(R^\sim ; -S)$ 
  unfolding rres-def converse-unfold by force

lemma lres-simp [simp]:  $(R // R) ; R = R$ 
  by (metis Id-O-R lres-galois relcomp-mono subsetI subsetI subset-antisym)

lemma rres-simp [simp]:  $R ; (R \setminus R) = R$ 
  by (metis converse-converse converse-relcomp lres-simp rres-lres-conv)

lemma lres-curry:  $R // (T ; S) = (R // S) // T$ 
  by (metis (no-types, opaque-lifting) O-assoc dual-order.refl lres-galois
subset-antisym)

lemma rres-curry:  $(R ; S) \setminus T = S \setminus (R \setminus T)$ 
  by (simp add: converse-relcomp lres-curry rres-lres-conv)

lemma lres-Id:  $Id \subseteq R // R$ 
  unfolding lres-def Id-def by force

lemma det-lres: deterministic  $R \implies (R ; S) // S = R ; (S // S)$ 
  by (metis (no-types, lifting) O-assoc deterministic-var2 lres-compl)

lemma det-rres: deterministic  $(R^\sim) \implies S \setminus (S ; R) = (S \setminus S) ; R$ 
  by (simp add: converse-relcomp det-lres rres-lres-conv)

lemma rres-bij: bijective  $S \implies (R \setminus T) ; S = R \setminus (T ; S)$ 
  unfolding bijective-def injective-set surjective-set relcomp-unfold cod-def Id-def
rres-def by clarsimp blast

lemma lres-bij: bijective  $S \implies (R // T^\sim) ; S = R // (T ; S)^\sim$ 
  unfolding bijective-def injective-set surjective-set relcomp-unfold cod-def Id-def
lres-def converse-unfold by blast

lemma dom-rres-top:  $(\text{dom } P \subseteq R \setminus (\text{dom } Q ; \text{top})) = (\text{dom } P ; \text{top} \subseteq R \setminus (\text{dom } Q ; \text{top}))$ 

```

unfolding *dom-def top-def rres-def relcomp-unfold Id-def converse-unfold* **by**
clarsimp blast

lemma *dom-rres-top-var*: $(\text{dom } P \subseteq R \setminus (\text{dom } Q ; \text{top})) = (P ; \text{top} \subseteq R \setminus (Q ; \text{top}))$
by (*metis dom-rres-top dom-top*)

lemma *fdia-rres-top*: $(|R\rangle P \subseteq \text{dom } Q) = (\text{dom } P \subseteq R \setminus (\text{dom } Q ; \text{top}))$
by (*metis dom-alt-def dom-gla-top rel-fdia-def rres-galois*)

lemma *fdia-rres-top-var*: $(|R\rangle P \subseteq \text{dom } Q) = (\text{dom } P \subseteq R \setminus (Q ; \text{top}))$
by (*metis dom-top fdia-rres-top*)

lemma *dom-galois-var2*: $(|R\rangle (Id \cap P) \subseteq Id \cap Q) = (Id \cap P \subseteq R \setminus ((Id \cap Q) ; \text{top}))$
by (*metis dom-subid fdia-rres-top-var inf-sup-aci(4)*)

lemma *rres-top*: $R \setminus (\text{dom } Q ; \text{top}) ; \text{top} = R \setminus (\text{dom } Q ; \text{top})$
unfolding *rres-def top-def dom-def relcomp-unfold* **by** *clarsimp*

lemma *ddd-var*: $(|R\rangle P \subseteq \text{dom } Q) = (\text{dom } P \subseteq \text{dom } ((R \setminus (\text{dom } Q ; \text{top})) ; \text{top}))$
unfolding *rel-fdia-def dom-def rres-def top-def relcomp-unfold Id-def* **by** *force*

lemma *wlp-prop*: $\text{dom } ((R \setminus (\text{dom } Q ; \text{top})) ; \text{top}) = \text{neg } (\text{cod } (\text{neg } (\text{dom } Q); R))$
unfolding *rres-def Id-def cod-def dom-def top-def relcomp-unfold* **by** *fastforce*

lemma *wlp-prop-var*: $\text{dom } ((R \setminus (\text{dom } Q ; \text{top}))) = \text{neg } (\text{cod } ((\text{neg } (\text{dom } Q)); R))$
by (*metis rres-top wlp-prop*)

lemma *dom-demod*: $(|R\rangle (Id \cap P) \subseteq Id \cap Q) = (R ; (Id \cap P) \subseteq (Id \cap Q) ; R)$
proof

assume $|R\rangle (Id \cap P) \subseteq Id \cap Q$
hence $R ; (Id \cap P) \subseteq (Id \cap Q) ; R ; (Id \cap P)$
by (*metis O-assoc dom-gla dom-subid inf.absorb-iff2 inf-le1 rel-fdia-def*)
thus $R ; (Id \cap P) \subseteq (Id \cap Q) ; R$
by *auto*

next

assume $R ; (Id \cap P) \subseteq (Id \cap Q) ; R$
hence $|R\rangle (Id \cap P) \subseteq \text{dom } ((Id \cap Q) ; R)$
by (*metis (no-types, lifting) dom-subid dom-top fdia-demod-top inf.absorb-iff2 inf-le1 relcomp-distrib2 sup.order-iff*)
hence $|R\rangle (Id \cap P) \subseteq (Id \cap Q) \cap \text{dom } R$

unfolding *dom-def Id-def* **by** *blast*
thus $|R\rangle (Id \cap P) \subseteq Id \cap Q$
by *blast*

qed

lemma *fdia-bbox-galois-var*: $(|R\rangle (Id \cap P) \subseteq Id \cap Q) = (Id \cap P \subseteq Id \cap - \text{cod}$

```
((Id ∩ −Q); R))
  unfolding rel-fdia-def dom-def cod-def Id-def by blast

lemma dom-demod-var2: ( |R⟩ (Id ∩ P) ⊆ Id ∩ Q) = (Id ∩ P ⊆ R \ ((Id ∩ Q)
; R))
  by (meson dom-demod rres-galois)
```

1.5 Symmetric quotient

```
definition syq :: ('c,'a) rel ⇒ ('c,'b) rel ⇒ ('a,'b) rel (infixl ‹÷› 75)
  where R ÷ S = (R \ S) ∩ (R ∼ // S ∼)
```

```
lemma syq-set: R ÷ S = {(a,b). ∀ c. (c,a) ∈ R ↔ (c,b) ∈ S}
  unfolding syq-def relcomp-unfold lres-def rres-def by force
```

```
lemma converse-syq [simp]: (R ÷ S) ∼ = S ÷ R
  unfolding syq-def converse-def rres-def lres-def by blast
```

```
lemma syq-compl: R ÷ S = −(R ∼ ; −S) ∩ −(−(R ∼) ; S)
  by (simp add: lres-compl rres-compl syq-def)
```

```
lemma syq-compl2 [simp]: −R ÷ −S = R ÷ S
  unfolding syq-compl by blast
```

```
lemma syq-expand1: R ; (R ÷ S) = S ∩ (top ; (R ÷ S))
  unfolding syq-set top-def relcomp-unfold by force
```

```
lemma syq-expand2: (R ÷ S) ; S ∼ = R ∼ ∩ ((R ÷ S) ; top)
  unfolding syq-set top-def relcomp-unfold by force
```

```
lemma syq-comp1: (R ÷ S) ; (S ÷ T) = (R ÷ T) ∩ (top ; (S ÷ T))
  unfolding syq-set top-def relcomp-unfold by fastforce
```

```
lemma syq-comp2: (R ÷ S) ; (S ÷ T) = (R ÷ T) ∩ ((R ÷ S) ; top)
  unfolding syq-set top-def relcomp-unfold by fastforce
```

```
lemma syq-bij: bijective T ⇒ (R ÷ S) ; T = R ÷ (S ; T)
  by (simp add: bijective-def inj-meet-distr lres-bij rres-bij syq-def)
```

end

2 Properties of Power Allegories

theory Power-Allegories-Properties

imports Relational-Properties

begin

2.1 Power transpose, epsilon, epsiloff

definition *Lambda* :: ('a,'b) rel \Rightarrow ('a,'b set) rel ($\langle\Lambda\rangle$) **where**

$$\Lambda R = \{(a,B) \mid a B. B = \{b. (a,b) \in R\}\}$$

definition *epsilon* :: ('a,'a set) rel **where**

$$\epsilon = \{(a,A). a \in A\}$$

definition *epsiloff* = $\{(A,a). a \in A\}$

definition *alpha* :: ('a,'b set) rel \Rightarrow ('a,'b) rel ($\langle\alpha\rangle$) **where**

$$\alpha R = R ; \text{epsiloff}$$

alpha can be seen as a relational approximation of a multirelation. The next lemma provides a relational definition of Lambda.

lemma *Lambda-syq*: $\Lambda R = R^\sim \doteqdot \epsilon$

unfolding *Lambda-def* *syq-set epsilon-def* **by** *blast*

lemma *epsiloff-epsilon*: *epsiloff* = ϵ^\sim

unfolding *epsiloff-def* *epsilon-def converse-unfold* **by** *simp*

lemma *alpha-set*: $\alpha R = \{(a,b) \mid a b. b \in \bigcup\{B. (a,B) \in R\}\}$

unfolding *alpha-def* *epsiloff-def* **by** *force*

lemma *alpha-relcomp [simp]*: $\alpha(R ; S) = R ; \alpha S$

by (*simp add: O-assoc alpha-def*)

lemma *Lambda-epsiloff-up1*: $f = \Lambda R \implies R = \alpha f$

unfolding *Lambda-def* *alpha-set* **by** *simp*

lemma *Lambda-epsiloff-up2: deterministic f* $\implies R = \alpha f \implies f = \Lambda R$

unfolding *Lambda-def* *alpha-set deterministic-set*

apply *safe*

apply *force*

by (*clarsimp, smt (verit, best) mem-Collect-eq set-eq-iff*)

lemma *Lambda-epsiloff-up*:

assumes *deterministic f*

shows $(R = \alpha f) = (f = \Lambda R)$

by (*meson Lambda-epsiloff-up1 Lambda-epsiloff-up2 assms*)

lemma *det-lambda: deterministic (Lambda R)*

unfolding *Lambda-def* *deterministic-set* **by** *simp*

lemma *Lambda-alpha-canc: deterministic f* $\implies \Lambda(\alpha f) = f$

using *Lambda-epsiloff-up2* **by** *blast*

lemma *alpha-Lambda-canc [simp]*: $\alpha(\Lambda R) = R$

using *Lambda-epsiloff-up1* **by** *blast*

```

lemma alpha-cancel:
  assumes deterministic f
  and deterministic g
  shows  $\alpha f = \alpha g \implies f = g$ 
  by (metis Lambda-epsilonff-up2 assms)

lemma Lambda-fusion:
  assumes deterministic f
  shows  $\Lambda(f ; R) = f ; \Lambda R$ 
proof -
  have h: deterministic (f ;  $\Lambda R$ )
    by (simp add: assms det-lambda det-relcomp)
  have f ; R =  $\alpha(f ; \Lambda R)$ 
    by simp
  also have ... = f ;  $\alpha(\Lambda R)$ 
    by simp
  thus ?thesis
    by (simp add: alpha-cancel det-lambda h)
qed

lemma Lambda-fusion-var:  $\Lambda(\Lambda R ; S) = \Lambda R ; \Lambda S$ 
  by (simp add: Lambda-fusion det-lambda)

lemma Lambda-epsilonff [simp]:  $\Lambda \text{ epsilonff} = Id$ 
proof -
  have  $\Lambda \text{ epsilonff} = \Lambda(Id ; \text{epsilonff})$ 
    by simp
  also have ... = Id
    by (metis Lambda-epsilonff-up alpha-def det-Id)
  finally show ?thesis.
qed

lemma alpha-epsilonff [simp]:  $\alpha Id = \text{epsilonff}$ 
  by (simp add: alpha-def)

lemma alpha-Sup-pres:  $\alpha(\bigcup \mathcal{R}) = (\bigcup R \in \mathcal{R}. \alpha R)$ 
  unfolding alpha-def by force

lemma alpha-ord-pres:  $R \subseteq S \implies \alpha R \subseteq \alpha S$ 
  unfolding alpha-def by force

lemma alpha-inf-pres:  $\alpha \{(a,A). \exists B C. A = B \cap C \wedge (a,B) \in R \wedge (a,C) \in S\}$ 
   $= \alpha R \cap \alpha S$ 
  unfolding alpha-set by blast

```

2.2 Relational image functor

definition pow :: ('a, 'b) rel \Rightarrow ('a set, 'b set) rel ($\langle \mathcal{P} \rangle$) **where**
 $\mathcal{P} R = \Lambda(\text{epsilonff} ; R)$

```

lemma pow-set:  $\mathcal{P} R = \{(A,B). B = \text{Image } R A\}$ 
  unfolding pow-def epsilon-def Lambda-def relcomp-def Image-def by force

lemma pow-set-var:  $\mathcal{P} R = \{(A,B). B = \{b. \exists a \in A. (a,b) \in R\}\}$ 
  unfolding pow-set Image-def by simp

lemma pow-converse-set:  $\mathcal{P} (R^\sim) = \{(Q,P). P = \{a. \exists b. (a,b) \in R \wedge b \in Q\}\}$ 
  unfolding pow-set Image-def by force

lemma det-pow: deterministic ( $\mathcal{P} R$ )
  unfolding pow-set deterministic-set Image-def by simp

lemma Lambda-pow:  $\Lambda (R ; S) = \Lambda R ; \mathcal{P} S$ 
proof -
  have  $\Lambda R ; \mathcal{P} S = \Lambda R ; \Lambda (\text{epsilon} ; S)$ 
    by (simp add: pow-def)
  also have ... =  $\Lambda (\Lambda R ; \text{epsilon} ; S)$ 
    by (simp add: Lambda-fusion-var O-assoc)
  also have ... =  $\Lambda (R ; S)$ 
    by (metis alpha-Lambda-canc alpha-def)
  finally show ?thesis..
qed

lemma pow-func1 [simp]:  $\mathcal{P} Id = Id$ 
  by (simp add: pow-def)

lemma pow-func2:  $\mathcal{P} (R ; S) = \mathcal{P} R ; \mathcal{P} S$ 
  by (metis Lambda-pow pow-def O-assoc)

lemma Grph-Image [simp]:  $\text{Grph} \circ \text{Image} = \mathcal{P}$ 
  apply (simp add: fun-eq-iff)
  unfolding pow-def Grph-def Image-def Lambda-def epsilon-def by blast

lemma lambda-alpha-idem [simp]:  $\Lambda (\alpha (\Lambda (\alpha R))) = \Lambda (\alpha R)$ 
  by simp

2.3 Unit and multiplication of powerset monad

definition eta :: ('a, 'a set) rel ( $\langle \eta \rangle$ ) where
   $\eta = \Lambda Id$ 

definition mu :: ('a set set, 'a set) rel ( $\langle \mu \rangle$ ) where
   $\mu = \text{pow epsilon}$ 

lemma eta-set:  $\eta = \{(a, \{a\}) | a. \text{True}\}$ 
  unfolding eta-def Lambda-def Id-def by simp

lemma alpha-eta [simp]:  $\alpha \eta = Id$ 

```

```

by (simp add: eta-def)

lemma det-eta: deterministic  $\eta$ 
  unfolding deterministic-set eta-set by simp

lemma mu-set:  $\mu = \{(A, B) \mid B = \{b \mid \exists C. C \in A \wedge b \in C\}\}$ 
  unfolding mu-def pow-def Lambda-def epsilon-def by force

lemma det-mu: deterministic  $\mu$ 
  unfolding deterministic-set mu-set by simp

lemma Lambda-eta:
  assumes deterministic R
  shows  $\Lambda R = R ; \eta$ 
proof -
  have  $\Lambda R = \Lambda (R ; Id)$ 
    by simp
  also have ... =  $R ; \Lambda Id$ 
    using Lambda-fusion assms by blast
  also have ... =  $R ; \eta$ 
    by (simp add: eta-def)
  finally show ?thesis.
qed

lemma eta-nat-trans:
  assumes deterministic R
  shows  $\eta ; \mathcal{P} R = R ; \eta$ 
proof -
  have  $\eta ; \mathcal{P} R = \Lambda Id ; \mathcal{P} R$ 
    by (simp add: eta-def)
  also have ... =  $\Lambda (Id ; R)$ 
    using Lambda-pow by blast
  also have ... =  $R ; \eta$ 
    by (simp add: Lambda-eta assms)
  finally show ?thesis.
qed

lemma mu-nat-trans:
  assumes deterministic R
  shows  $\mathcal{P} (\mathcal{P} R) ; \mu = \mu ; \mathcal{P} R$ 
  by (metis pow-def alpha-Lambda-canc alpha-def mu-def pow-func2)

The standard axioms for the powerset monad are derivable.

lemma pow-monad1 [simp]:  $\mathcal{P} \mu ; \mu = \mu ; \mu$ 
  by (metis pow-def alpha-Lambda-canc alpha-def mu-def pow-func2)

lemma pow-monad2 [simp]:  $\mathcal{P} \eta ; \mu = Id$ 

```

```

by (metis alpha-Lambda-canc alpha-def eta-def mu-def pow-func1 pow-func2)

lemma pow-monad3 [simp]:  $\eta ; \mu = Id$ 
  by (metis Lambda-epsiloff Lambda-pow alpha-def alpha-epsiloff eta-def mu-def)

lemma Lambda-mu:
  assumes deterministic R
  shows  $\Lambda(R) ; \mu = R$ 
proof -
  have  $\Lambda R ; \mu = R ; \eta ; \mu$ 
    by (simp add: Lambda-eta assms)
  also have ... = R
    by (simp add: O-assoc)
  finally show ?thesis.
qed

lemma pow-Lambda-mu [simp]:  $\mathcal{P}(\Lambda R) ; \mu = \mathcal{P} R$ 
  by (metis alpha-Lambda-canc alpha-def mu-def pow-func2)

lemma lambda-alpha-mu:  $\Lambda(\alpha R) = \Lambda R ; \mu$ 
  by (simp add: Lambda-pow alpha-def mu-def)

lemma alpha-eta-pow [simp]:  $\alpha(\eta ; \mathcal{P} R) = R$ 
proof -
  have  $\alpha(\eta ; \mathcal{P} R) = \alpha(\Lambda Id ; \mathcal{P} R)$ 
    by (simp add: eta-def)
  also have ... =  $\alpha(\Lambda(Id ; R))$ 
    by (metis Lambda-pow)
  also have ... = R
    by simp
  finally show ?thesis.
qed

lemma eta-pow-Lambda [simp]:  $\eta ; \mathcal{P} R = \Lambda R$ 
  by (metis Id-O-R Lambda-pow eta-def)

lemma pow-prop1:  $\mathcal{P} R \subseteq S \implies R \subseteq \alpha(\eta ; S)$ 
  by (metis alpha-eta-pow alpha-ord-pres relcomp-distrib subset-Un-eq)

lemma pow-prop-2:  $R \subseteq \mathcal{P} S \implies \alpha(\eta ; R) \subseteq S$ 
  by (metis alpha-eta-pow alpha-ord-pres relcomp-distrib subset-Un-eq)

lemma pow-prop:  $R = \mathcal{P} S \implies \alpha(\eta ; R) = S$ 
  using alpha-eta-pow by blast

lemma alpha-eta-id [simp]:  $\alpha(R ; \eta) = R$ 
  by simp

lemma eta-alpha-idem [simp]:  $\alpha(\alpha R ; \eta) ; \eta = \alpha R ; \eta$ 

```

by simp

lemma lambda-eta-alpha [simp]: $\Lambda (\alpha (\alpha R ; \eta)) = \Lambda (\alpha R)$
by simp

lemma eta-lambda-idem [simp]: $\alpha (\Lambda (\alpha R)) ; \eta = \alpha R ; \eta$
by simp

lemma Grph-eta [simp]: $Grph (\lambda x. \{x\}) = \eta$
unfolded Grph-def eta-def Lambda-def Id-def by force

lemma Grph-epsiloff [simp]: $Grph (\lambda x. \{x\}) ; \text{epsiloff} = Id$
by (metis Grph-eta alpha-def alpha-eta)

lemma Image-epsiloff [simp]: $\text{Image } \text{epsiloff} \circ (\lambda x. \{x\}) = id$
unfolded Image-def epsiloff-def id-def by force

2.4 Subset relation

definition Omega :: ('a set, 'a set) rel ($\langle \Omega \rangle$) where
 $\Omega = \text{epsilon} \setminus \text{epsilon}$

lemma Omega-set: $\Omega = \{(A, B). A \subseteq B\}$
unfolded Omega-def rres-def epsilon-def by force

lemma conv-Omega: $\Omega^\sim = \text{epsiloff} // \text{epsiloff}$
by (simp add: Omega-def epsiloff-epsilon rres-lres-conv)

lemma epsilon-eta-Omega [simp]: $\eta ; \Omega = \text{epsilon}$
unfolded eta-set Omega-set epsilon-def by force

lemma epsiloff-eta-Omega [simp]: $\Omega^\sim ; \eta^\sim = \text{epsiloff}$
by (metis converse-relcomp epsiloff-epsilon epsilon-eta-Omega)

lemma epsilon-Omega [simp]: $\text{epsilon} ; \Omega = \text{epsilon}$
by (simp add: Omega-def)

lemma conv-Omega-epsiloff [simp]: $\Omega^\sim ; \text{epsiloff} = \text{epsiloff}$
by (simp add: conv-Omega)

lemma Lambda-conv [simp]: $(\Lambda R)^\sim = \text{epsilon} \div R^\sim$
by (simp add: Lambda-syq)

lemma Lambda-Omega: $\Lambda R ; \Omega = R^\sim \setminus \text{epsilon}$
proof –
have $\Lambda R ; \Omega = \Lambda R ; (\text{epsilon} \setminus \text{epsilon})$
by (simp add: Omega-def)
also have ... = $\Lambda R ; -(\text{epsiloff} ; -\text{epsilon})$
by (simp add: epsiloff-epsilon rres-compl)

```

also have ... =  $-(\Lambda R ; \text{epsilon-off} ; -\text{epsilon})$ 
  by (metis O-assoc det-lambda deterministic-var2)
also have ... =  $-(R ; -\text{epsilon})$ 
  by (metis alpha-Lambda-canc alpha-def)
also have ... =  $R^\sim \setminus \text{epsilon}$ 
  by (simp add: rres-compl)
finally show ?thesis.
qed

lemma syq-epsilon-off-prop [simp]:  $\Omega^\sim ; (\text{epsilon} \div R) = \text{epsilon-off} // R^\sim$ 
  by (metis Lambda-Omega Lambda-syq converse-converse converse-relcomp
converse-syq epsilon-off-epsilon rres-lres-conv)

lemma pow-semicomm:  $((P, Q) \in \mathcal{P} R ; \Omega) = (\Delta P ; R \subseteq R ; \Delta Q)$ 
  unfolding pow-set Image-def Omega-def rres-def epsilon-def Delta-def by blast

```

2.5 Complementation relation

```

definition Compl :: ('a set, 'a set) rel ( $\langle \mathcal{C} \rangle$ ) where
   $\mathcal{C} = \text{epsilon} \div -\text{epsilon}$ 

```

```

lemma Compl-set:  $\mathcal{C} = \{(A, -A) \mid A. \text{True}\}$ 
  unfolding Compl-def syq-set epsilon-def by force

```

```

lemma Compl-Compl [simp]:  $\mathcal{C} ; \mathcal{C} = \text{Id}$ 
  by (metis Compl-def Lambda-syq
boolean-algebra-class.boolean-algebra.double-compl converse-converse converse-syq
det-lambda deterministic-def set-eq-subset syq-compl2 total-def univalent-def)

```

```

lemma Compl-def-var:  $\mathcal{C} = \Lambda (-\text{epsilon-off})$ 
  by (metis Compl-def Lambda-syq
boolean-algebra-class.boolean-algebra.double-compl compl-conv converse-converse
epsilon-off-epsilon syq-compl2)

```

```

lemma converse-Compl [simp]:  $\mathcal{C}^\sim = \mathcal{C}$ 
  by (metis Compl-def converse-syq double-complement syq-compl2)

```

```

lemma det-Compl: deterministic  $\mathcal{C}$ 
  by (simp add: Compl-def-var det-lambda)

```

```

lemma bij-Compl: bijective  $\mathcal{C}$ 
  by (simp add: bij-det det-Compl)

```

```

lemma Compl-compl-epsilon-off [simp]:  $\mathcal{C} ; -\text{epsilon-off} = \text{epsilon-off}$ 
  by (metis Compl-Compl Compl-def-var alpha-Lambda-canc alpha-epsilon-off
alpha-relcomp)

```

```

lemma Compl-epsilon-off [simp]:  $\mathcal{C} ; \text{epsilon-off} = -\text{epsilon-off}$ 
  by (smt (z3) Compl-def-var alpha-Lambda-canc alpha-def)

```

```

lemma compl-epsilon-Compl [simp]:  $\neg \text{epsilon} ; \mathcal{C} = \text{epsilon}$ 
  by (metis Compl-compl-epsilon Compl-conv converse-Compl converse-converse
converse-relcomp epsilon-off-epsilon)

lemma epsilon-Compl [simp]:  $\text{epsilon} ; \mathcal{C} = \neg \text{epsilon}$ 
  by (metis Compl-epsilon-off Compl-conv converse-Compl converse-converse
converse-relcomp epsilon-off-epsilon)

lemma Lambda-Compl-var:  $\Lambda R ; \mathcal{C} = R^\sim \div \neg \text{epsilon}$ 
  by (simp add: Lambda-syq bij-det det-Compl syq-bij)

lemma Lambda-Compl:  $\Lambda R ; \mathcal{C} = \Lambda (\neg R)$ 
proof -
  have  $\Lambda R ; \mathcal{C} = \Lambda R ; \Lambda (\neg \text{epsilon})$ 
    by (simp add: Compl-def-var)
  also have  $\dots = \Lambda (\Lambda R ; \neg \text{epsilon})$ 
    by (simp add: Lambda-fusion-var)
  also have  $\dots = \Lambda (\neg (\Lambda R ; \text{epsilon}))$ 
    by (metis det-lambda deterministic-var2)
  also have  $\dots = \Lambda (\neg R)$ 
    by (metis alpha-Lambda-canc alpha-def)
  finally show ?thesis.
qed

```

2.6 Kleisli lifting and Kleisli composition

```

definition klift :: ('a,'b set) rel  $\Rightarrow$  ('a set,'b set) rel ( $\langle \cdot \rangle_{\mathcal{P}}$  [1000] 999) where
   $(R)_{\mathcal{P}} = \mathcal{P} (\alpha R)$ 

```

```

definition kcomp :: ('a,'b set) rel  $\Rightarrow$  ('b,'c set) rel  $\Rightarrow$  ('a,'c set) rel (infixl  $\cdot_{\mathcal{P}}$  70) where
   $R \cdot_{\mathcal{P}} S = R ; (S)_{\mathcal{P}}$ 

```

```

lemma klift-var:  $(R)_{\mathcal{P}} = \Lambda (\text{epsilon-off} ; R ; \text{epsilon-off})$ 
  by (simp add: pow-def O-assoc alpha-def klift-def)

```

```

lemma klift-set:  $(R)_{\mathcal{P}} = \{(A,B). B = \bigcup (\text{Image } R A)\}$ 
  unfolding klift-def Image-def pow-set alpha-set by force

```

```

lemma klift-set-var:  $(R)_{\mathcal{P}} = \{(A,B). B = \bigcup \{C. \exists a \in A. (a,C) \in R\}\}$ 
  unfolding klift-set Image-def by auto

```

```

lemma klift-mu:  $(R)_{\mathcal{P}} = \mathcal{P} R ; \mu$ 
proof -
  have  $(R)_{\mathcal{P}} = \mathcal{P} (R ; \text{epsilon-off})$ 
    by (simp add: alpha-def klift-def)
  also have  $\dots = \mathcal{P} R ; \mathcal{P} \text{ epsilon-off}$ 
    by (simp add: pow-func2)

```

```

also have ... =  $\mathcal{P} R ; \mu$ 
  by (simp add: mu-def)
finally show ?thesis.
qed

lemma klift-empty:  $(\{\}, A) \in (R)_\mathcal{P} \longleftrightarrow A = \{\}$ 
  using klift-set by auto

lemma klift-ext1:  $(R ; (S)_\mathcal{P})_\mathcal{P} = (R)_\mathcal{P} ; (S)_\mathcal{P}$ 
  by (metis (no-types, opaque-lifting) Lambda-epsiloff-up1 Lambda-fusion-var
    O-assoc alpha-def klift-var)

lemma klift-ext2: deterministic  $R \implies \eta ; (R)_\mathcal{P} = R$ 
  by (metis Id-O-R Lambda-alpha-canc Lambda-pow eta-def klift-def)

lemma klift-ext3 [simp]:  $(\eta)_\mathcal{P} = Id$ 
  by (simp add: klift-def)

lemma pow-klift [simp]:  $(R ; \eta)_\mathcal{P} = \mathcal{P} R$ 
  by (simp add: klift-def)

lemma mu-klift [simp]:  $(Id)_\mathcal{P} = \mu$ 
  by (simp add: klift-def mu-def)

lemma kcomp-var:  $R \cdot_\mathcal{P} S = R ; \mathcal{P} S ; \mu$ 
  by (simp add: O-assoc kcomp-def klift-mu)

lemma kcomp-assoc:  $R \cdot_\mathcal{P} (S \cdot_\mathcal{P} T) = (R \cdot_\mathcal{P} S) \cdot_\mathcal{P} T$ 
proof -
  have  $R \cdot_\mathcal{P} (S \cdot_\mathcal{P} T) = R ; (S ; (T)_\mathcal{P})_\mathcal{P}$ 
    by (simp add: kcomp-def)
  also have ... =  $R ; ((S)_\mathcal{P} ; (T)_\mathcal{P})$ 
    by (simp add: klift-ext1)
  also have ... =  $(R \cdot_\mathcal{P} S) \cdot_\mathcal{P} T$ 
    by (simp add: O-assoc kcomp-def)
  finally show ?thesis.
qed

lemma kcomp-oner:  $R \cdot_\mathcal{P} \eta = R$ 
  by (simp add: kcomp-def)

lemma kcomp-onel: deterministic  $R \implies \eta \cdot_\mathcal{P} R = R$ 
  by (simp add: kcomp-def klift-ext2)

```

2.7 Relational box

```

definition rbox :: ('a,'b) rel  $\Rightarrow$  ('b set, 'a set) rel where
  rbox  $R = \Lambda (\text{epsiloff} // R)$ 

```

```

lemma rbox-set: rbox R = {(Q,P). P = {a.  $\forall b. (a,b) \in R \rightarrow b \in Q\}}$ }
  unfolding rbox-def Lambda-def lres-def epsiloff-def
  by force

lemma rbox-exp: ((Q,P) ∈ (rbox (R::('a,'b) rel))) = (P = -{a.  $\exists b. (a,b) \in R \wedge b \in -Q\}$ )
  by (smt (z3) Collect-cong Collect-neg-eq ComplD ComplI case-prodD case-prodI
mem-Collect-eq rbox-set)

lemma rbox-subset: rbox R ;  $\Omega^\sim$  = {(Q,P). P ⊆ {a.  $\forall b. (a,b) \in R \rightarrow b \in Q\}$ }
  unfolding rbox-set Omega-set by blast

lemma rbox-semicomm: (Q,P) ∈ rbox R ;  $\Omega^\sim$  = ( $\Delta P$  ; R ⊆ R ;  $\Delta Q$ )
  unfolding rbox-subset Delta-def by blast

lemma rbox-semicomm-var: (Q,P) ∈ rbox R ;  $\Omega^\sim$  = ( $\Delta P \subseteq (R ; \Delta Q) // R$ )
  by (simp add: lres-galois rbox-semicomm)

lemma rbox-omega: rbox epsiloff =  $\Lambda (\Omega^\sim)$ 
  by (simp add: conv-Omega rbox-def)

lemma Omega-rbox:  $\Omega = (\alpha (rbox \text{epsiloff}))^\sim$ 
  by (simp add: rbox-omega)

lemma pow-rbox: ((Q,P) ∈ rbox R ;  $\Omega^\sim$ ) = ((P,Q) ∈  $\mathcal{P} R$  ;  $\Omega$ )
proof –
  have (Q,P) ∈ rbox R ;  $\Omega^\sim$  = ( $\Delta P$  ; R ⊆ R ;  $\Delta Q$ )
    by (simp add: rbox-semicomm)
  also have ... = ((P,Q) ∈  $\mathcal{P} R$  ;  $\Omega$ )
    by (simp add: pow-semicomm)
  finally show ?thesis.
qed

lemma rbox-pow-Compl: rbox R =  $\mathcal{C}$  ;  $\mathcal{P} (R^\sim)$  ;  $\mathcal{C}$ 
proof –
  have  $\mathcal{C}$  ;  $\mathcal{P} (R^\sim)$  ;  $\mathcal{C}$  =  $\Lambda (-\text{epsiloff})$  ;  $\mathcal{P} (R^\sim)$  ;  $\mathcal{C}$ 
    by (simp add: Compl-def-var)
  also have ... =  $\Lambda (-\text{epsiloff} ; R^\sim)$  ;  $\mathcal{C}$ 
    by (simp add: Lambda-pow)
  also have ... =  $\Lambda (-(-\text{epsiloff} ; R^\sim))$ 
    by (simp add: Lambda-Compl)
  also have ... =  $\Lambda (\text{epsiloff} // R)$ 
    by (simp add: lres-compl)
  also have ... = rbox R
    by (simp add: rbox-def)
  finally show ?thesis..
qed

lemma pow-rbox-Compl:  $\mathcal{P} R = \mathcal{C}$  ; rbox (R $^\sim$ ) ;  $\mathcal{C}$ 

```

```

by (metis Compl-Compl Id-O-R O-assoc R-O-Id converse-converse
rbox-pow-Compl)

lemma pow-conjugation: C ; (P (R~) ; Ω)~ = P R ; C ; Ω~
proof -
  have P R ; C ; Ω~ = Λ (-(epsiloff ; R)) ; -(epsiloff ; epsilon)
    by (simp add: Lambda-Compl pow-def conv-Omega epsiloff-epsilon lres-compl)
  also have ... = -(Λ (-(epsiloff ; R)) ; -epsiloff ; epsilon)
    by (metis (no-types, opaque-lifting) alpha-Lambda-canc alpha-def
converse-converse det-lambda det-lres deterministic-var2 epsiloff-epsilon
lres-compl)
  also have ... = -(Λ (-(epsiloff ; R)) ; C ; epsiloff ; epsilon)
    by (metis Compl-epsiloff alpha-def alpha-relcomp)
  also have ... = -(Λ (epsiloff ; R) ; epsiloff ; epsilon)
    by (simp add: Lambda-Compl)
  also have ... = -(epsiloff ; R ; epsilon)
    by (metis Lambda-epsiloff-up1 alpha-def)
  also have ... = -(C ; -epsiloff ; (epsiloff ; R~)~)
    by (metis Compl-compl-epsiloff O-assoc converse-converse converse-relcomp
epsiloff-epsilon)
  also have ... = C ; (epsiloff // (epsiloff ; R~))
    by (metis (no-types, opaque-lifting) Compl-def-var Lambda-Compl
Lambda-fusion-var pow-def alpha-Lambda-canc
boolean-algebra-class.boolean-algebra.double-compl converse-converse
pow-rbox-Compl rbox-def)
  also have ... = C; (P (R~) ; Ω)~
    by (simp add: Lambda-Omega pow-def epsiloff-epsilon rres-lres-conv)
  finally show ?thesis..
qed

lemma pow-rbox-eq: rbox R ; Ω~ = (P R ; Ω)~
  by (metis (no-types, lifting) Compl-Compl O-assoc R-O-Id converse-converse
converse-relcomp pow-conjugation rbox-pow-Compl)

end

```

3 Basic Properties of Multirelations

```

theory Multirelations-Basics

imports Power-Allegories-Properties

begin

```

This theory extends a previous AFP entry for multirelations with one single objects to proper multirelations in Rel.

3.1 Peleg composition, parallel composition (inner union) and units

```

type-synonym ('a,'b) mrel = ('a,'b set) rel

definition s-prod :: ('a,'b) mrel  $\Rightarrow$  ('b,'c) mrel  $\Rightarrow$  ('a,'c) mrel (infixl  $\leftrightarrow$  75)
where

$$R \cdot S = \{(a,A). (\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (b,f b) \in S) \wedge A = \bigcup(f ' B)))\}$$


definition s-id :: ('a,'a) mrel ( $\langle 1_\sigma \rangle$ ) where

$$1_\sigma = (\bigcup a. \{(a,\{a\})\})$$


definition p-prod :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel (infixl  $\langle \parallel \rangle$  70)
where

$$R \parallel S = \{(a,A). (\exists B C. A = B \cup C \wedge (a,B) \in R \wedge (a,C) \in S)\}$$


definition p-id :: ('a,'b) mrel ( $\langle 1_\pi \rangle$ ) where

$$1_\pi = (\bigcup a. \{(a,\{\})\})$$


definition U :: ('a,'b) mrel where

$$U = \{(a,A) \mid a \text{ A. } \text{True}\}$$


abbreviation NC  $\equiv$  U  $-$  1 $\pi$ 

named-theorems mr-simp
declare s-prod-def [mr-simp] p-prod-def [mr-simp] s-id-def [mr-simp] p-id-def [mr-simp]
[U-def [mr-simp]

lemma s-prod-idl [simp]:  $1_\sigma \cdot R = R$ 
by (auto simp: mr-simp)

lemma s-prod-idr [simp]:  $R \cdot 1_\sigma = R$ 
by (auto simp: mr-simp) (metis UN-singleton)

lemma p-prod-ild [simp]:  $1_\pi \parallel R = R$ 
by (simp add: mr-simp)

lemma c-prod-idr [simp]:  $R \parallel 1_\pi = R$ 
by (simp add: mr-simp)

lemma cl7 [simp]:  $1_\sigma \parallel 1_\sigma = 1_\sigma$ 
by (auto simp: mr-simp)

lemma p-prod-assoc:  $R \parallel S \parallel T = R \parallel (S \parallel T)$ 
apply (rule set-eqI, clarsimp simp: mr-simp)
by (metis (no-types, lifting) sup-assoc)

lemma p-prod-comm:  $R \parallel S = S \parallel R$ 
by (auto simp: mr-simp)

```

```

lemma subidem-par:  $R \subseteq R \parallel R$ 
  by (auto simp: mr-simp)

lemma meet-le-par:  $R \cap S \subseteq R \parallel S$ 
  by (auto simp: mr-simp)

lemma s-prod-distr:  $(R \cup S) \cdot T = R \cdot T \cup S \cdot T$ 
  by (auto simp: mr-simp)

lemma s-prod-sup-distr:  $(\bigcup X) \cdot S = (\bigcup R \in X. R \cdot S)$ 
  by (auto simp: mr-simp)

lemma s-prod-subdistl:  $R \cdot S \cup R \cdot T \subseteq R \cdot (S \cup T)$ 
  by (auto simp: mr-simp)

lemma s-prod-sup-subdistl:  $X \neq \{\} \implies (\bigcup S \in X. R \cdot S) \subseteq R \cdot \bigcup X$ 
  by (simp add: mr-simp) blast

lemma s-prod-isol:  $R \subseteq S \implies R \cdot T \subseteq S \cdot T$ 
  by (metis s-prod-distr sup.order-iff)

lemma s-prod-isor:  $R \subseteq S \implies T \cdot R \subseteq T \cdot S$ 
  by (metis le-supE s-prod-subdistl sup.absorb-iff1)

lemma s-prod-zerol [simp]:  $\{\} \cdot R = \{\}$ 
  by (force simp: mr-simp)

lemma s-prod-wzeror:  $R \cdot \{\} \subseteq R$ 
  by (force simp: mr-simp)

lemma p-prod-zeror [simp]:  $R \parallel \{\} = \{\}$ 
  by (simp add: mr-simp)

lemma s-prod-p-idl [simp]:  $1_\pi \cdot R = 1_\pi$ 
  by (force simp: mr-simp)

lemma p-id-st:  $R \cdot 1_\pi = \{(a,\{\}) \mid a. \exists B. (a,B) \in R\}$ 
  by (force simp: mr-simp)

lemma c6:  $R \cdot 1_\pi \subseteq 1_\pi$ 
  by (clar simp simp: mr-simp)

lemma p-prod-distl:  $R \parallel (S \cup T) = R \parallel S \cup R \parallel T$ 
  by (fastforce simp: mr-simp)

lemma p-prod-sup-distl:  $R \parallel (\bigcup X) = (\bigcup S \in X. R \parallel S)$ 
  by (fastforce simp: mr-simp)

```

lemma *p-prod-isol*: $R \subseteq S \implies R \parallel T \subseteq S \parallel T$
by (*metis p-prod-comm p-prod-distl sup.orderE sup.orderI*)

lemma *p-prod-isor*: $R \subseteq S \implies T \parallel R \subseteq T \parallel S$
by (*simp add: p-prod-comm p-prod-isol*)

lemma *s-prod-assoc1*: $(R \cdot S) \cdot T \subseteq R \cdot (S \cdot T)$
by (*clarsimp simp: mr-simp*) *metis*

lemma *seq-conc-subdistr*: $(R \parallel S) \cdot T \subseteq R \cdot T \parallel S \cdot T$
by (*clarsimp simp: mr-simp UnI1 UnI2*) *blast*

lemma *U-U* [*simp*]: $U \cdot U = U$
by (*simp add: mr-simp*) *blast*

lemma *U-par-idem* [*simp*]: $U \parallel U = U$
by (*simp add: U-def equalityI subidem-par*)

lemma *p-id-NC*: $R - 1_\pi = R \cap NC$
by (*simp add: Diff-eq U-def*)

lemma *NC-NC* [*simp*]: $NC \cdot NC = NC$
by (*rule set-eqI, clarsimp simp: mr-simp, metis SUP-bot-conv(2) UN-constant insert-not-empty*)

lemma *nc-par-idem* [*simp*]: $NC \parallel NC = NC$
by (*force simp: mr-simp*)

lemma *cl4*:
assumes $T \parallel T \subseteq T$
shows $R \cdot T \parallel S \cdot T \subseteq (R \parallel S) \cdot T$
proof *clarify*
fix *a A*
assume $(a,A) \in R \cdot T \parallel S \cdot T$
hence $\exists B C. A = B \cup C \wedge (\exists D. (a,D) \in R \wedge (\exists f. (\forall d \in D. (d,f d) \in T) \wedge B = \bigcup ((\lambda x. f x) \cdot D))) \wedge (\exists E. (a,E) \in S \wedge (\exists g. (\forall e \in E. (e,g e) \in T) \wedge C = \bigcup ((\lambda x. g x) \cdot E)))$
by (*simp add: mr-simp*)
hence $\exists D E. (a,D \cup E) \in R \parallel S \wedge (\exists f g. (\forall d \in D. (d,f d) \in T) \wedge (\forall e \in E. (e,g e) \in T) \wedge A = (\bigcup ((\lambda x. f x) \cdot D)) \cup (\bigcup ((\lambda x. g x) \cdot E)))$
by (*auto simp: mr-simp*)
hence $\exists D E. (a,D \cup E) \in R \parallel S \wedge (\exists f g. (\forall d \in D-E. (d,f d) \in T) \wedge (\forall e \in E-D. (e,g e) \in T) \wedge (\forall x \in D \cap E. (x,f x) \in T \wedge (x,g x) \in T) \wedge A = (\bigcup ((\lambda x. f x) \cdot (D-E)) \cup (\bigcup ((\lambda x. g x) \cdot (E-D)) \cup (\bigcup ((\lambda y. f y \cup g y) \cdot (D \cap E))))))$
by *auto blast*
hence $\exists D E. (a,D \cup E) \in R \parallel S \wedge (\exists f g. (\forall d \in D-E. (d,f d) \in T) \wedge (\forall e \in E-D. (e,g e) \in T) \wedge (\forall x \in D \cap E. (x,f x \cup g x) \in T) \wedge A = (\bigcup ((\lambda x. f x) \cdot (D-E)) \cup (\bigcup ((\lambda x. g x) \cdot (E-D)) \cup (\bigcup ((\lambda y. f y \cup g y) \cdot (D \cap E))))))$
apply *clarify*

```

subgoal for D E f g
  apply (rule exI[of - D])
  apply (rule exI[of - E])
  apply clarify
  apply (rule exI[of - f])
  apply (rule exI[of - g])
  using assms by (auto simp: p-prod-def, blast)
  done
  hence  $\exists D E. (a, D \cup E) \in R \parallel S \wedge (\exists h. (\forall d \in D - E. (d, h d) \in T) \wedge (\forall e \in E - D. (e, h e) \in T) \wedge (\forall x \in D \cap E. (x, h x) \in T) \wedge A = (\bigcup ((\lambda x. h x) ` (D - E))) \cup (\bigcup ((\lambda x. h x) ` (E - D)) \cup (\bigcup ((\lambda y. h y) ` (D \cap E))))))$ 
  apply clarify
  subgoal for D E f g
    apply (rule exI[of - D])
    apply (rule exI[of - E])
    apply clarify
    apply (rule exI[of -  $\lambda x.$  if  $x \in (D - E)$  then  $f x$  else (if  $x \in D \cap E$  then  $(f x \cup g x)$  else  $g x$ )])
    by auto
    done
  hence  $(\exists B. (a, B) \in R \parallel S \wedge (\exists h. (\forall b \in B. (b, h b) \in T) \wedge A = \bigcup ((\lambda x. h x) ` B)))$ 
    by clarsimp blast
  thus  $(a, A) \in (R \parallel S) \cdot T$ 
    by (simp add: mr-simp)
qed

```

```

lemma cl3:  $R \cdot (S \parallel T) \subseteq R \cdot S \parallel R \cdot T$ 
proof clarify
  fix a A
  assume  $(a, A) \in R \cdot (S \parallel T)$ 
  hence  $\exists B. (a, B) \in R \wedge (\exists f. (\forall b \in B. \exists C D. f b = C \cup D \wedge (b, C) \in S \wedge (b, D) \in T) \wedge A = \bigcup ((\lambda x. f x) ` B))$ 
    by (clarsimp simp: mr-simp)
  hence  $\exists B. (a, B) \in R \wedge (\exists f g h. (\forall b \in B. f b = g b \cup h b \wedge (b, g b) \in S \wedge (b, h b) \in T) \wedge A = \bigcup ((\lambda x. f x) ` B))$ 
    by (clarsimp simp: bchoice, metis)
  hence  $\exists B. (a, B) \in R \wedge (\exists g h. (\forall b \in B. (b, g b) \in S \wedge (b, h b) \in T) \wedge A = (\bigcup ((\lambda x. g x) ` B)) \cup (\bigcup ((\lambda x. h x) ` B)))$ 
    by blast
  hence  $\exists C D. (\exists B. (a, B) \in R \wedge (\exists g. (\forall b \in B. (b, g b) \in S) \wedge C = \bigcup ((\lambda x. g x) ` B))) \wedge (\exists B. (a, B) \in R \wedge (\exists h. (\forall b \in B. (b, h b) \in T) \wedge D = \bigcup ((\lambda x. h x) ` B))) \wedge A = C \cup D$ 
    by blast
  thus  $(a, A) \in R \cdot S \parallel R \cdot T$ 
    by (auto simp: mr-simp)
qed

```

lemma p-id-assoc1: $(1_\pi \cdot R) \cdot S = 1_\pi \cdot (R \cdot S)$

by *simp*

lemma *p-id-assoc2*: $(R \cdot 1_\pi) \cdot T = R \cdot (1_\pi \cdot T)$
by (*rule set-eqI, clarsimp simp: mr-simp*) *fastforce*

lemma *cl1 [simp]*: $R \cdot 1_\pi \cup R \cdot NC = R \cdot U$
by (*rule set-eqI, clarsimp simp: mr-simp, metis UN-constant UN-empty*)

lemma *tarski-aux*:

assumes $R - 1_\pi \neq \{\}$
and $(a, A) \in NC$
shows $(a, A) \in NC \cdot ((R - 1_\pi) \cdot NC)$
using assms apply (*clarsimp simp: mr-simp*)
by (*metis UN-constant insert-not-empty singletonD*)

lemma *tarski*:

assumes $R - 1_\pi \neq \{\}$
shows $NC \cdot ((R - 1_\pi) \cdot NC) = NC$
by *standard (simp add: U-def p-id-def s-prod-def, force, metis assms tarski-aux subrelI)*

lemma *tarski-var*:

assumes $R \cap NC \neq \{\}$
shows $NC \cdot ((R \cap NC) \cdot NC) = NC$
by (*metis assms p-id-NC tarski*)

lemma *s-le-nc*: $1_\sigma \subseteq NC$
by (*auto simp: mr-simp*)

lemma *U-nc [simp]*: $U \cdot NC = U$
by (*metis NC-NC cl1 s-prod-distr s-prod-idl s-prod-p-idl*)

lemma *x-y-split [simp]*: $(R \cap NC) \cdot S \cup R \cdot \{\} = R \cdot S$
by (*auto simp: mr-simp*)

lemma *c-nc-comp1 [simp]*: $1_\pi \cup NC = U$
using *cl1 s-prod-idl by blast*

3.2 Tests

lemma *s-id-st*: $R \cap 1_\sigma = \{(a, \{a\}) \mid a. (a, \{a\}) \in R\}$
by (*force simp: mr-simp*)

lemma *subid-aux2*:
assumes $(a, A) \in R \cap 1_\sigma$
shows $A = \{a\}$
using assms by (*auto simp: mr-simp*)

lemma *s-prod-test-aux1*:

```

assumes (a,A) ∈ R · (P ∩ 1σ)
shows ((a,A) ∈ R ∧ (∀ a ∈ A. (a,{a}) ∈ (P ∩ 1σ)))
using assms by (auto simp: mr-simp)

lemma s-prod-test-aux2:
assumes (a,A) ∈ R
and ∀ a ∈ A. (a,{a}) ∈ S
shows (a,A) ∈ R · S
using assms by (fastforce simp: mr-simp)

lemma s-prod-test: (a,A) ∈ R · (P ∩ 1σ) ←→ (a,A) ∈ R ∧ (∀ a ∈ A. (a,{a}) ∈
(P ∩ 1σ))
by (meson s-prod-test-aux1 s-prod-test-aux2)

lemma s-prod-test-var: R · (P ∩ 1σ) = {(a,A). (a,A) ∈ R ∧ (∀ a ∈ A. (a,{a}) ∈
(P ∩ 1σ))}
apply (rule antisym)
by (fastforce simp: mr-simp) +

lemma test-s-prod-aux1:
assumes (a,A) ∈ (P ∩ 1σ) · R
shows (a,{a}) ∈ (P ∩ 1σ) ∧ (a,A) ∈ R
using assms by (auto simp: mr-simp)

lemma test-s-prod-aux2:
assumes (a,A) ∈ R
and (a,{a}) ∈ P
shows (a,A) ∈ P · R
using assms s-prod-def by fastforce

lemma test-s-prod: (a,A) ∈ (P ∩ 1σ) · R ←→ (a,{a}) ∈ (P ∩ 1σ) ∧ (a,A) ∈ R
by (meson test-s-prod-aux1 test-s-prod-aux2)

lemma test-s-prod-var: (P ∩ 1σ) · R = {(a,A). (a,{a}) ∈ (P ∩ 1σ) ∧ (a,A) ∈ R}
by (simp add: set-eq-iff test-s-prod)

lemma test-assoc1: (R · (P ∩ 1σ)) · S = R · ((P ∩ 1σ) · S)
apply (rule antisym)
apply (simp add: s-prod-assoc1)
apply (clarify simp: mr-simp)
by (metis UN-singleton)

lemma test-assoc2: ((P ∩ 1σ) · R) · S = (P ∩ 1σ) · (R · S)
apply (rule antisym)
apply (simp add: s-prod-assoc1)
by (fastforce simp: mr-simp s-prod-assoc1)

lemma test-assoc3: (R · S) · (P ∩ 1σ) = R · (S · (P ∩ 1σ))
proof (rule antisym)

```

```

show ( $R \cdot S$ ) · ( $P \cap 1_\sigma$ )  $\subseteq R \cdot (S \cdot (P \cap 1_\sigma))$ 
  by (simp add: s-prod-assoc1)
show  $R \cdot (S \cdot (P \cap 1_\sigma)) \subseteq (R \cdot S) \cdot (P \cap 1_\sigma)$ 
  proof clarify
    fix  $a A$ 
    assume  $hyp1: (a, A) \in R \cdot (S \cdot (P \cap 1_\sigma))$ 
    hence  $\exists B. (a, B) \in R \wedge (\exists f. (\forall b \in B. (b, f b) \in S \cdot (P \cap 1_\sigma)) \wedge A = \bigcup((\lambda x. f x) ` B))$ 
      by (simp add: s-prod-test s-prod-def)
      hence  $\exists B. (a, B) \in R \wedge (\exists f. (\forall b \in B. (b, f b) \in S \wedge (\forall a \in f b. (a, \{a\}) \in (P \cap 1_\sigma))) \wedge A = \bigcup((\lambda x. f x) ` B))$ 
        by (simp add: s-prod-test)
        hence  $\exists B. (a, B) \in R \wedge (\exists f. (\forall b \in B. (b, f b) \in S) \wedge (\forall a \in \bigcup((\lambda x. f x) ` B). (a, \{a\}) \in (P \cap 1_\sigma))) \wedge A = \bigcup((\lambda x. f x) ` B))$ 
          by auto
          hence  $\exists B. (a, B) \in R \wedge (\exists f. (\forall b \in B. (b, f b) \in S) \wedge (\forall a \in A. (a, \{a\}) \in (P \cap 1_\sigma))) \wedge A = \bigcup((\lambda x. f x) ` B))$ 
            by auto blast
            hence  $(a, A) \in R \cdot S \wedge (\forall a \in A. (a, \{a\}) \in (P \cap 1_\sigma))$ 
              by (auto simp: mr-simp)
              thus  $(a, A) \in (R \cdot S) \cdot (P \cap 1_\sigma)$ 
                by (simp add: s-prod-test)
    qed
qed

lemma s-distl-test:  $(P \cap 1_\sigma) \cdot (S \cup T) = (P \cap 1_\sigma) \cdot S \cup (P \cap 1_\sigma) \cdot T$ 
  by (fastforce simp: mr-simp)

lemma s-distl-sup-test:  $(P \cap 1_\sigma) \cdot \bigcup X = (\bigcup S \in X. (P \cap 1_\sigma) \cdot S)$ 
  by (auto simp: mr-simp)

lemma subid-par-idem [simp]:  $(P \cap 1_\sigma) \parallel (P \cap 1_\sigma) = (P \cap 1_\sigma)$ 
  by (auto simp: mr-simp)

lemma seq-conc-subdistr:  $(P \cap 1_\sigma) \cdot (S \parallel T) = ((P \cap 1_\sigma) \cdot S) \parallel ((P \cap 1_\sigma) \cdot T)$ 
  apply (rule antisym)
  apply (simp add: cl3)
  by (fastforce simp: mr-simp)

lemma test-s-prod-is-meet [simp]:  $(P \cap 1_\sigma) \cdot (Q \cap 1_\sigma) = P \cap Q \cap 1_\sigma$ 
  by (auto simp: mr-simp)

lemma test-p-prod-is-meet [simp]:  $(P \cap 1_\sigma) \parallel (Q \cap 1_\sigma) = (P \cap 1_\sigma) \cap (Q \cap 1_\sigma)$ 
  by (auto simp: mr-simp)

lemma test-multiplicativer:  $(P \cap Q \cap 1_\sigma) \cdot T = ((P \cap 1_\sigma) \cdot T) \cap ((Q \cap 1_\sigma) \cdot T)$ 
  by (auto simp: mr-simp)

```

```

lemma cl9 [simp]:  $(R \cap 1_\sigma) \cdot 1_\pi \parallel 1_\sigma = R \cap 1_\sigma$ 
  by (auto simp: mr-simp)

lemma s-subid-closed [simp]:  $R \cap NC \cap 1_\sigma = R \cap 1_\sigma$ 
  using s-le-nc by auto

lemma sub-id-le-nc:  $R \cap 1_\sigma \subseteq NC$ 
  by (simp add: le-infI2 s-le-nc)

lemma x-y-prop:  $1_\sigma \cap ((R \cap NC) \cdot S) = 1_\sigma \cap R \cdot S$ 
  by (auto simp: mr-simp)

lemma s-nc-U:  $1_\sigma \cap R \cdot NC = 1_\sigma \cap R \cdot U$ 
  by (rule set-eqI, clarsimp simp: mr-simp, metis SUP-constant UN-empty
insert-not-empty)

```

```

lemma sid-le-nc-var:  $1_\sigma \cap R \subseteq 1_\sigma \cap (R \parallel NC)$ 
  using meet-le-par s-le-nc by fastforce

lemma s-nc-par-U:  $1_\sigma \cap (R \parallel NC) = 1_\sigma \cap (R \parallel U)$ 
  by (metis c-nc-comp1 c-prod-idr inf-sup-distrib1 le-iff-sup p-prod-distl
sid-le-nc-var)

```

```

lemma s-id-par-s-prod:  $(P \cap 1_\sigma) \parallel (Q \cap 1_\sigma) = (P \cap 1_\sigma) \cdot (Q \cap 1_\sigma)$ 
  by force

```

3.3 Parallel subidentities

```

lemma p-id-zero-st:  $R \cap 1_\pi = \{(a,\{\}) \mid a. (a,\{\}) \in R\}$ 
  by (auto simp: mr-simp)

lemma p-subid-iff:  $R \subseteq 1_\pi \longleftrightarrow R \cdot 1_\pi = R$ 
  by (clarsimp simp: mr-simp, safe, simp-all) blast+

lemma p-subid-iff-var:  $R \subseteq 1_\pi \longleftrightarrow R \cdot \{\} = R$ 
  by (clarsimp simp: mr-simp, safe, simp-all) blast+

lemma term-par-idem [simp]:  $(R \cap 1_\pi) \parallel (R \cap 1_\pi) = (R \cap 1_\pi)$ 
  by (metis Int-Un-eq(4) c-prod-idr p-prod-distl subidem-par subset-Un-eq)

lemma c1 [simp]:  $R \cdot 1_\pi \parallel R = R$ 
  apply (rule set-eqI, clarsimp simp: mr-simp)
  by (metis (no-types, lifting) SUP-bot SUP-bot-conv(2) sup-bot-left)

lemma p-id-zero:  $R \cap 1_\pi = R \cdot \{\}$ 
  by (auto simp: mr-simp)

lemma cl5:  $(R \cdot S) \cdot (T \cdot \{\}) = R \cdot (S \cdot (T \cdot \{\}))$ 
  proof (rule antisym)

```

```

show  $(R \cdot S) \cdot (T \cdot \{\}) \subseteq R \cdot (S \cdot (T \cdot \{\}))$ 
  by (metis s-prod-assoc1)
show  $R \cdot (S \cdot (T \cdot \{\})) \subseteq (R \cdot S) \cdot (T \cdot \{\})$ 
proof clarify
  fix  $a::'a$  and  $A::'f set$ 
  assume  $(a,A) \in R \cdot (S \cdot (T \cdot \{\}))$ 
  hence  $\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (\exists C. (b,C) \in S \wedge (\exists g. (\forall x \in C. (x,g) \in T \cdot \{\}) \wedge f b = \bigcup((\lambda x. g x) \cdot C)))) \wedge A = \bigcup((\lambda x. f x) \cdot B))$ 
    by (clarsimp simp: mr-simp)
  hence  $\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (\exists C. (b,C) \in S \wedge (\forall x \in C. (x,\{) \in T \cdot \{\}) \wedge f b = \{\})) \wedge A = \bigcup((\lambda x. f x) \cdot B))$ 
    by (clarsimp simp: mr-simp) fastforce
  hence  $\exists B. (a,B) \in R \wedge (\forall b \in B. (\exists C. (b,C) \in S \wedge (\forall x \in C. (x,\{) \in T \cdot \{\}))) \wedge A = \{\})$ 
    by fastforce
  hence  $\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (b,f b) \in S \wedge (\forall x \in f b. (x,\{) \in T \cdot \{\}))) \wedge A = \{\})$ 
    by (metis (mono-tags))
  hence  $\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (b,f b) \in S) \wedge (\forall x \in \bigcup((\lambda x. f x) \cdot B). (x,\{) \in T \cdot \{\})) \wedge A = \{\})$ 
    by (metis UN-E)
  hence  $\exists C B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (b,f b) \in S) \wedge C = \bigcup((\lambda x. f x) \cdot B) \wedge (\forall x \in C. (x,\{) \in T \cdot \{\})) \wedge A = \{\})$ 
    by metis
  hence  $\exists C. (a,C) \in R \cdot S \wedge (\forall x \in C. (x,\{) \in T \cdot \{\}) \wedge A = \{\})$ 
    by (auto simp: mr-simp)
  thus  $(a,A) \in (R \cdot S) \cdot (T \cdot \{\})$ 
    by (clarsimp simp: mr-simp) blast
qed
qed

```

```

lemma c4:  $(R \cdot S) \cdot 1_\pi = R \cdot (S \cdot 1_\pi)$ 
proof -
  have  $(R \cdot S) \cdot 1_\pi = \{(a,\{) \mid a. \exists B. (a,B) \in R \cdot S\}$ 
    by (simp add: p-id-st)
  also have ... =  $R \cdot \{(a,\{) \mid a. \exists B. (a,B) \in S\}$ 
    apply (clarsimp simp: mr-simp)
    apply safe
    apply blast
    applyclarsimp
    by metis
  also have ... =  $R \cdot (S \cdot 1_\pi)$ 
    by (simp add: p-id-st)
  finally show ?thesis.
qed

```

```

lemma c3:  $(R \parallel S) \cdot 1_\pi = R \cdot 1_\pi \parallel S \cdot 1_\pi$ 
  by (simp add: Orderings.order-eq-iff cl4 seq-conc-subdistr)

```

lemma *p-id-idem* [simp]: $(R \cdot 1_\pi) \cdot 1_\pi = R \cdot 1_\pi$
by (simp add: c4)

lemma *x-c-par-idem* [simp]: $R \cdot 1_\pi \parallel R \cdot 1_\pi = R \cdot 1_\pi$
by (metis c1 p-id-idem)

lemma *x-zero-le-c*: $R \cdot \{\} \subseteq 1_\pi$
by (simp add: c4 p-subid-iff)

lemma *p-subid-lb1*: $R \cdot \{\} \parallel S \cdot \{\} \subseteq R \cdot \{\}$
by (metis c-prod-idr p-prod-isor x-zero-le-c)

lemma *p-subid-lb2*: $R \cdot \{\} \parallel S \cdot \{\} \subseteq S \cdot \{\}$
using p-prod-comm p-subid-lb1 **by** blast

lemma *p-subid-idem* [simp]: $R \cdot \{\} \parallel R \cdot \{\} = R \cdot \{\}$
by (simp add: p-subid-lb1 subidem-par subset-antisym)

lemma *p-subid-glb*: $T \cdot \{\} \subseteq R \cdot \{\} \implies T \cdot \{\} \subseteq S \cdot \{\} \implies T \cdot \{\} \subseteq (R \cdot \{\}) \parallel (S \cdot \{\})$
by (auto simp: mr-simp)

lemma *p-subid-glb-iff*: $T \cdot \{\} \subseteq R \cdot \{\} \wedge T \cdot \{\} \subseteq S \cdot \{\} \longleftrightarrow T \cdot \{\} \subseteq (R \cdot \{\}) \parallel (S \cdot \{\})$
by (auto simp: mr-simp)

lemma *x-c-glb*: $(T::('a,'b) mrel) \cdot 1_\pi \subseteq (R::('a,'b) mrel) \cdot 1_\pi \implies T \cdot 1_\pi \subseteq (S::('a,'b) mrel) \cdot 1_\pi \implies T \cdot 1_\pi \subseteq (R \cdot 1_\pi) \parallel (S \cdot 1_\pi)$
by (smt (verit, best) order-subst1 p-id-idem p-prod-isol p-prod-isor s-prod-isol x-c-par-idem)

lemma *x-c-lb1*: $R \cdot 1_\pi \parallel S \cdot 1_\pi \subseteq R \cdot 1_\pi$
by (metis c6 c-prod-idr p-prod-isor)

lemma *x-c-lb2*: $R \cdot 1_\pi \parallel S \cdot 1_\pi \subseteq S \cdot 1_\pi$
using p-prod-comm x-c-lb1 **by** blast

lemma *x-c-glb-iff*: $(T::('a,'b) mrel) \cdot 1_\pi \subseteq (R::('a,'b) mrel) \cdot 1_\pi \wedge T \cdot 1_\pi \subseteq (S::('a,'b) mrel) \cdot 1_\pi \longleftrightarrow T \cdot 1_\pi \subseteq (R \cdot 1_\pi) \parallel (S \cdot 1_\pi)$
by (meson subset-trans x-c-glb x-c-lb1 x-c-lb2)

lemma *nc-iff1*: $R \subseteq NC \longleftrightarrow R \cap 1_\pi = \{\}$
by (metis (no-types, lifting) Diff-Diff-Int Int-Diff Int-absorb diff-shunt-var p-id-NC)

lemma *nc-iff2*: $R \subseteq NC \longleftrightarrow R \cdot \{\} = \{\}$
by (metis c4 nc-iff1 p-id-zero s-prod-zerol)

lemma *zero-assoc3*: $(R \cdot S) \cdot \{\} = R \cdot (S \cdot \{\})$

```

by (metis cl5 s-prod-zerol)

lemma x-zero-interr:  $R \cdot \{\} \parallel S \cdot \{\} = (R \parallel S) \cdot \{\}$ 
  by (clar simp simp: mr-simp) blast

lemma p-subid-interr:  $R \cdot T \cdot 1_\pi \parallel S \cdot T \cdot 1_\pi = (R \parallel S) \cdot T \cdot 1_\pi$ 
  proof -
    have  $R \cdot T \cdot 1_\pi \parallel S \cdot T \cdot 1_\pi = (R \cdot \{(a,\{\}) \mid a. \exists B. (a,B) \in T\}) \parallel (S \cdot \{(a,\{\}) \mid a. \exists B. (a,B) \in T\})$ 
      by (metis c4 p-id-st)
    also have ... =  $(R \parallel S) \cdot \{(a,\{\}) \mid a. \exists B. (a,B) \in T\}$ 
      apply (rule antisym)
      apply (metis cl4 dual-order.refl p-id-st x-c-par-idem)
      by (simp add: seq-conc-subdistr)
    also have ... =  $(R \parallel S) \cdot T \cdot 1_\pi$ 
      by (metis c4 p-id-st)
    finally show ?thesis.
  qed

lemma cl2 [simp]:  $1_\pi \cap (R \cup NC) = R \cdot \{\}$ 
  by (metis Diff-disjoint Int-Un-distrib inf-commute p-id-zero sup-bot.right-neutral)

lemma cl6 [simp]:  $R \cdot \{\} \cdot S = R \cdot \{\}$ 
  by (metis c4 p-id-assoc2 s-prod-p-idl s-prod-zerol)

lemma cl11 [simp]:  $(R \cap NC) \cdot 1_\pi \parallel NC = (R \cap NC) \cdot NC$ 
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  apply (metis UN-constant)
  apply (clar simp simp: mr-simp)
  by (metis UN-empty2 UN-insert Un-empty-left equals0I insert-absorb sup-bot-right)

lemma x-split [simp]:  $(R \cap NC) \cup (R \cap 1_\pi) = R$ 
  by (metis Un-Diff-Int p-id-NC)

lemma x-split-var [simp]:  $(R \cap NC) \cup R \cdot \{\} = R$ 
  by (metis p-id-zero x-split)

lemma s-x-c [simp]:  $1_\sigma \cap R \cdot 1_\pi = \{\}$ 
  using c6 s-le-nc by fastforce

lemma s-x-zero [simp]:  $1_\sigma \cap R \cdot \{\} = \{\}$ 
  using cl6 s-x-c by blast

lemma c-nc [simp]:  $R \cdot 1_\pi \cap NC = \{\}$ 
  using c6 by blast

```

```

lemma zero-nc [simp]:  $R \cdot \{\} \cap NC = \{\}$ 
  using x-zero-le-c by fastforce

lemma nc-zero [simp]:  $(R \cap NC) \cdot \{\} = \{\}$ 
  using nc-iff2 by auto

lemma c-def [simp]:  $U \cdot \{\} = 1_\pi$ 
  by (metis c-nc-comp1 cl2 cl6 inf-commute p-id-zero s-prod-p-idl)

lemma U-c [simp]:  $U \cdot 1_\pi = 1_\pi$ 
  by (metis U-U c-def zero-assoc3)

lemma nc-c [simp]:  $NC \cdot 1_\pi = 1_\pi$ 
  by (auto simp: mr-simp)

lemma nc-U [simp]:  $NC \cdot U = U$ 
  using NC-NC c-nc-comp1 cl1 nc-c by blast

lemma x-c-nc-split [simp]:  $((R \cap NC) \cdot NC) \cup (R \cdot \{\} \parallel NC) = (R \cdot 1_\pi) \parallel NC$ 
  by (metis cl11 p-prod-comm p-prod-distl x-y-split)

lemma x-c-U-split [simp]:  $R \cdot U \cup (R \cdot \{\} \parallel U) = R \cdot 1_\pi \parallel U$ 
  apply (rule set-eqI, clarsimp simp: mr-simp)
  by (metis SUP-constant UN-extend-simps(2))

lemma p-subid-par-eq-meet [simp]:  $R \cdot \{\} \parallel S \cdot \{\} = R \cdot \{\} \cap S \cdot \{\}$ 
  by (auto simp: mr-simp)

lemma p-subid-par-eq-meet-var [simp]:  $R \cdot 1_\pi \parallel S \cdot 1_\pi = R \cdot 1_\pi \cap S \cdot 1_\pi$ 
  by (metis c-def p-subid-par-eq-meet zero-assoc3)

lemma x-zero-add-closed:  $R \cdot \{\} \cup S \cdot \{\} = (R \cup S) \cdot \{\}$ 
  by (simp add: s-prod-distr)

lemma x-zero-meet-closed:  $R \cdot \{\} \cap S \cdot \{\} = (R \cap S) \cdot \{\}$ 
  by (force simp: mr-simp)

lemma scomp-univalent-pres: univalent  $R \implies$  univalent  $S \implies$  univalent  $(R \cdot S)$ 
  unfolding univalent-set s-prod-def
  apply clarsimp
  by (metis Sup.SUP-cong)

lemma univalent s-id
  unfolding univalent-set s-id-def by simp

lemma det-peleg: deterministic  $R \implies$  deterministic  $S \implies$  deterministic  $(R \cdot S)$ 
  unfolding deterministic-set s-prod-def
  apply clarsimp
  apply safe

```

```

apply metis
apply (metis UN-I)
by (metis UN-I)

lemma deterministic-sid: deterministic 1σ
  unfolding deterministic-set s-id-def by simp

3.4 Domain

definition Dom :: ('a,'b) mrel ⇒ ('a,'a) mrel where
  Dom R = {(a,{a}) | a. ∃ B. (a,B) ∈ R}

named-theorems mrd-simp
declare mr-simp [mrd-simp] Dom-def [mrd-simp]

lemma d-def-expl: Dom R = R · 1π || 1σ
  by (force simp: mrd-simp set-eqI)

lemma s-subid-iff2: (R ∩ 1σ = R) = (Dom R = R)
  by (metis c6 cl9 d-def-expl inf.order-iff p-prod-comm p-prod-ild p-prod-isor)

lemma cl8-var: Dom R · S = R · 1π || S
  apply (rule antisym)
  apply (metis d-def-expl p-id-assoc2 s-prod-idl s-prod-p-idl seq-conc-subdistr)
  by (force simp: mrd-simp)

lemma cl8 [simp]: R · 1π || 1σ · S = R · 1π || S
  by simp

lemma cl10-var: Dom (R − 1π) = 1σ ∩ ((R − 1π) · NC)
  apply (rule set-eqI, clarsimp simp: mrd-simp)
  apply safe
  apply (metis SUP-constant insert-not-empty)
  by blast

lemma c10: (R ∩ NC) · 1π || 1σ = 1σ ∩ ((R ∩ NC) · NC)
  by (metis Int-Diff cl10-var d-def-expl)

lemma cl9-var [simp]: Dom (R ∩ 1σ) = R ∩ 1σ
  by (simp add: d-def-expl)

lemma d-s-id [simp]: Dom R ∩ 1σ = Dom R
  by (metis cl8-var d-def-expl p-prod-comm p-prod-ild s-subid-iff2)

lemma d-s-id-ax: Dom R ⊆ 1σ
  by (simp add: le-iff-inf)

lemma d-assoc1: Dom R · (S · T) = (Dom R · S) · T
  by (metis d-s-id test-assoc2)

```

lemma *d-meet-distr-var*: $(\text{Dom } R \cap \text{Dom } S) \cdot T = \text{Dom } R \cdot T \cap \text{Dom } S \cdot T$
by (*metis (no-types, lifting) d-s-id inf-assoc test-multiplicatiative*)

lemma *d-idem [simp]*: $\text{Dom}(\text{Dom } R) = \text{Dom } R$
by (*meson d-s-id s-subid-iff2*)

lemma *cd-2-var*: $\text{Dom}(R \cdot 1_\pi) \cdot S = R \cdot 1_\pi \parallel S$
by (*simp add: cl8-var p-id-assoc2*)

lemma *dc-prop1 [simp]*: $\text{Dom } R \cdot 1_\pi = R \cdot 1_\pi$
by (*simp add: cl8-var*)

lemma *dc-prop2 [simp]*: $\text{Dom}(R \cdot 1_\pi) = \text{Dom } R$
by (*simp add: d-def-expl p-id-assoc2*)

lemma *ds-prop [simp]*: $\text{Dom } R \parallel 1_\sigma = \text{Dom } R$
by (*simp add: p-prod-assoc d-def-expl*)

lemma *dc [simp]*: $\text{Dom } 1_\pi = 1_\sigma$
by (*simp add: d-def-expl*)

lemma *cd-iso [simp]*: $\text{Dom}(R \cdot 1_\pi) \cdot 1_\pi = R \cdot 1_\pi$
by *simp*

lemma *dc-iso [simp]*: $\text{Dom}(\text{Dom } R \cdot 1_\pi) = \text{Dom } R$
by *simp*

lemma *d-s-id-inter [simp]*: $\text{Dom } R \cdot \text{Dom } S = \text{Dom } R \cap \text{Dom } S$
by (*metis d-s-id inf-assoc test-s-prod-is-meet*)

lemma *d-conc6*: $\text{Dom}(R \parallel S) = \text{Dom } R \parallel \text{Dom } S$
by (*metis (no-types, lifting) c3 d-def-expl ds-prop p-prod-assoc p-prod-comm*)

lemma *d-conc-inter [simp]*: $\text{Dom } R \parallel \text{Dom } S = \text{Dom } R \cap \text{Dom } S$
by (*metis d-s-id test-p-prod-is-meet*)

lemma *d-conc-s-prod-ax*: $\text{Dom } R \parallel \text{Dom } S = \text{Dom } R \cdot \text{Dom } S$
by *simp*

lemma *d-rest-ax [simp]*: $\text{Dom } R \cdot R = R$
by (*simp add: cl8-var*)

lemma *d-loc-ax [simp]*: $\text{Dom}(R \cdot \text{Dom } S) = \text{Dom}(R \cdot S)$
by (*metis c4 dc-prop1 dc-prop2*)

lemma *assoc-p-subid*: $(R \cdot S) \cdot (T \cdot 1_\pi) = R \cdot (S \cdot (T \cdot 1_\pi))$
by (*smt (verit, del-insts) c4 cd-iso d-idem d-loc-ax p-id-idem s-subid-iff2 test-assoc3*)

lemma *d-exp-ax* [*simp*]: $\text{Dom}(\text{Dom } R \cdot S) = \text{Dom } R \cdot \text{Dom } S$
by (*metis d-conc6 d-conc-s-prod-ax d-idem d-loc-ax*)

lemma *d-comm-ax*: $\text{Dom } R \cdot \text{Dom } S = \text{Dom } S \cdot \text{Dom } R$
by *force*

lemma *d-s-id-prop* [*simp*]: $\text{Dom } 1_\sigma = 1_\sigma$
by (*simp add: d-def-expl*)

lemma *d-s-prod-closed* [*simp*]: $\text{Dom}(\text{Dom } R \cdot \text{Dom } S) = \text{Dom } R \cdot \text{Dom } S$
using *d-exp-ax d-loc-ax* **by** *blast*

lemma *d-p-prod-closed* [*simp*]: $\text{Dom}(\text{Dom } R \parallel \text{Dom } S) = \text{Dom } R \parallel \text{Dom } S$
using *d-s-prod-closed* **by** *auto*

lemma *d-idem2* [*simp*]: $\text{Dom } R \cdot \text{Dom } R = \text{Dom } R$
by (*metis d-exp-ax d-rest-ax*)

lemma *d-assoc*: $(\text{Dom } R \cdot \text{Dom } S) \cdot \text{Dom } T = \text{Dom } R \cdot (\text{Dom } S \cdot \text{Dom } T)$
using *d-assoc1* **by** *blast*

lemma *iso-1* [*simp*]: $\text{Dom } R \cdot 1_\pi \parallel 1_\sigma = \text{Dom } R$
using *d-def-expl* **by** *force*

lemma *d-idem-par* [*simp*]: $\text{Dom } R \parallel \text{Dom } R = \text{Dom } R$
by (*simp add: d-conc-s-prod-ax*)

lemma *d-inter-r*: $\text{Dom } R \cdot (S \parallel T) = \text{Dom } R \cdot S \parallel \text{Dom } R \cdot T$
by (*metis d-s-id seq-conc-subdistr*)

lemma *d-add-ax*: $\text{Dom}(R \cup S) = \text{Dom } R \cup \text{Dom } S$
by (*simp add: d-def-expl p-prod-comm p-prod-distl s-prod-distr*)

lemma *d-sup-add*: $\text{Dom}(\bigcup X) = (\bigcup R \in X. \text{Dom } R)$
by (*auto simp: mrd-simp*)

lemma *d-distl*: $\text{Dom } R \cdot (S \cup T) = \text{Dom } R \cdot S \cup \text{Dom } R \cdot T$
by (*metis d-s-id s-distl-test*)

lemma *d-sup-distl*: $\text{Dom } R \cdot \bigcup X = (\bigcup S \in X. \text{Dom } R \cdot S)$
by (*metis d-s-id s-distl-sup-test*)

lemma *d-zero-ax* [*simp*]: $\text{Dom } \{\} = \{\}$
by (*simp add: d-def-expl p-prod-comm*)

lemma *d-absorb1* [*simp*]: $\text{Dom } R \cup \text{Dom } R \cdot \text{Dom } S = \text{Dom } R$
by *simp*

lemma *d-absorb2* [*simp*]: $\text{Dom } R \cdot (\text{Dom } R \cup \text{Dom } S) = \text{Dom } R$
by (*metis d-absorb1 d-idem2 d-s-id s-distl-test*)

lemma *d-dist1*: $\text{Dom } R \cdot (\text{Dom } S \cup \text{Dom } T) = \text{Dom } R \cdot \text{Dom } S \cup \text{Dom } R \cdot \text{Dom } T$
by (*simp add: cl8-var p-prod-distl*)

lemma *d-dist2*: $\text{Dom } R \cup (\text{Dom } S \cdot \text{Dom } T) = (\text{Dom } R \cup \text{Dom } S) \cdot (\text{Dom } R \cup \text{Dom } T)$
by (*smt (verit) boolean-algebra.disj-conj-distrib d-add-ax d-s-id-inter dc-prop2*)

lemma *d-add-prod-closed* [*simp*]: $\text{Dom } (\text{Dom } R \cup \text{Dom } S) = \text{Dom } R \cup \text{Dom } S$
by (*simp add: d-add-ax*)

lemma *x-zero-prop*: $R \cdot \{\} \parallel S = \text{Dom } (R \cdot \{\}) \cdot S$
by (*simp add: cl8-var*)

lemma *cda-add-ax*: $\text{Dom } ((R \cup S) \cdot T) = \text{Dom } (R \cdot T) \cup \text{Dom } (S \cdot T)$
by (*simp add: d-add-ax s-prod-distr*)

lemma *d-x-zero*: $\text{Dom } (R \cdot \{\}) = R \cdot \{\} \parallel 1_\sigma$
by (*simp add: d-def-expl*)

lemma *cda-ax2*:
assumes $(R \parallel S) \cdot \text{Dom } T = R \cdot \text{Dom } T \parallel S \cdot \text{Dom } T$
shows $\text{Dom } ((R \parallel S) \cdot T) = \text{Dom } (R \cdot T) \cdot \text{Dom } (S \cdot T)$
by (*metis assms d-conc6 d-conc-s-prod-ax d-loc-ax*)

lemma *d-lb1*: $\text{Dom } R \cdot \text{Dom } S \subseteq \text{Dom } R$
using *d-absorb1* **by** *blast*

lemma *d-lb2*: $\text{Dom } R \cdot \text{Dom } S \subseteq \text{Dom } S$
using *d-comm-ax d-lb1* **by** *blast*

lemma *d-glb*: $\text{Dom } T \subseteq \text{Dom } R \wedge \text{Dom } T \subseteq \text{Dom } S \implies \text{Dom } T \subseteq \text{Dom } R \cdot \text{Dom } S$
by *simp*

lemma *d-glb-iff*: $\text{Dom } T \subseteq \text{Dom } R \wedge \text{Dom } T \subseteq \text{Dom } S \longleftrightarrow \text{Dom } T \subseteq \text{Dom } R \cdot \text{Dom } S$
by *force*

lemma *d-interr*: $R \cdot \text{Dom } P \parallel S \cdot \text{Dom } P = (R \parallel S) \cdot \text{Dom } P$
by (*simp add: cl4 seq-conc-subdistr subset-antisym*)

lemma *cl10-d*: $\text{Dom } (R \cap NC) = 1_\sigma \cap (R \cap NC) \cdot NC$
by (*simp add: c10 d-def-expl*)

lemma *cl11-d* [*simp*]: $\text{Dom } (R \cap NC) \cdot NC = (R \cap NC) \cdot NC$

by (*simp add: cl8-var*)

lemma *cl10-d-var1*: $\text{Dom } (R \cap NC) = 1_\sigma \cap R \cdot NC$
by (*simp add: cl10-d x-y-prop*)

lemma *cl10-d-var2*: $\text{Dom } (R \cap NC) = 1_\sigma \cap (R \cap NC) \cdot U$
by (*simp add: cl10-d-var1 s-nc-U x-y-prop*)

lemma *cl10-d-var3*: $\text{Dom } (R \cap NC) = 1_\sigma \cap R \cdot U$
by (*simp add: cl10-d-var1 s-nc-U*)

lemma *d-U* [*simp*]: $\text{Dom } U = 1_\sigma$
by (*metis U-c dc dc-prop2*)

lemma *d-nc* [*simp*]: $\text{Dom } NC = 1_\sigma$
by (*metis dc dc-prop2 nc-c*)

lemma *alt-d-def-nc-nc*: $\text{Dom } (R \cap NC) = 1_\sigma \cap (((R \cap NC) \cdot 1_\pi) \parallel NC)$
using *c10 cl11-d cl8-var d-def-expl* **by** *blast*

lemma *alt-d-def-nc-U*: $\text{Dom } (R \cap NC) = 1_\sigma \cap (((R \cap NC) \cdot 1_\pi) \parallel U)$
using *alt-d-def-nc-nc s-nc-par-U* **by** *blast*

lemma *d-def-split* [*simp*]: $\text{Dom } (R \cap NC) \cup \text{Dom } (R \cdot \{\}) = \text{Dom } R$
by (*metis d-add-ax d-loc-ax d-zero-ax p-id-zero x-split*)

lemma *d-def-split-var* [*simp*]: $\text{Dom } (R \cap NC) \cup ((R \cdot \{\}) \parallel 1_\sigma) = \text{Dom } R$
using *d-def-split d-x-zero* **by** *blast*

lemma *ax7* [*simp*]: $(1_\sigma \cap R \cdot U) \cup (R \cdot \{\}) \parallel 1_\sigma = \text{Dom } R$
using *cl10-d-var3 d-def-split d-x-zero* **by** *blast*

lemma *dom12-d*: $\text{Dom } R = 1_\sigma \cap (R \cdot 1_\pi \parallel NC)$
by (*metis cl10-d-var3 cl8-var d-idem d-s-id inf.orderE s-nc-par-U sub-id-le-nc*)

lemma *dom12-d-U*: $\text{Dom } R = 1_\sigma \cap (R \cdot 1_\pi \parallel U)$
by (*simp add: dom12-d s-nc-par-U*)

lemma *dom-def-var*: $\text{Dom } R = (R \cdot U \cap 1_\pi) \parallel 1_\sigma$
by (*simp add: d-def-expl p-id-zero zero-assoc3*)

lemma *ax5-d* [*simp*]: $\text{Dom } (R \cap NC) \cdot U = (R \cap NC) \cdot U$
by (*metis cl1 cl11-d dc-prop1*)

lemma *ax5-0* [*simp*]: $\text{Dom } (R \cdot \{\}) \cdot U = R \cdot \{\} \parallel U$
using *x-zero-prop* **by** *auto*

lemma *x-c-U-split2*: $\text{Dom } R \cdot NC = (R \cap NC) \cdot NC \cup (R \cdot \{\} \parallel NC)$
by (*simp add: cl8-var*)

lemma *x-c-U-split3*: $\text{Dom } R \cdot U = (R \cap NC) \cdot U \cup (R \cdot \{\} \parallel U)$
by (*metis ax5-0 ax5-d d-def-split s-prod-distr*)

lemma *x-c-U-split-d*: $\text{Dom } R \cdot U = R \cdot U \cup (R \cdot \{\} \parallel U)$
by (*simp add: cl8-var*)

lemma *x-U-prop2*: $R \cdot NC = \text{Dom } (R \cap NC) \cdot NC \cup R \cdot \{\}$
by *simp*

lemma *x-U-prop3*: $R \cdot U = \text{Dom } (R \cap NC) \cdot U \cup R \cdot \{\}$
by (*metis ax5-d x-y-split*)

lemma *d-x-nc [simp]*: $\text{Dom } (R \cdot NC) = \text{Dom } R$
by (*metis d-loc-ax d-nc dc dc-prop2*)

lemma *d-x-U [simp]*: $\text{Dom } (R \cdot U) = \text{Dom } R$
by (*metis d-U d-loc-ax dc dc-prop2*)

lemma *d-lp1*: $\text{Dom } R \subseteq \text{Dom } S \implies R \subseteq \text{Dom } S \cdot R$
by (*metis d-rest-ax s-prod-isol*)

lemma *d-lp2*: $R \subseteq \text{Dom } S \cdot R \implies \text{Dom } R \subseteq \text{Dom } S$
by (*metis d-assoc1 d-exp-ax d-meet-distr-var d-rest-ax d-s-id-inter inf.absorb-iff2*)

lemma *demod1*: $\text{Dom } (R \cdot S) \subseteq \text{Dom } T \implies R \cdot \text{Dom } S \subseteq \text{Dom } T \cdot R$
proof –
 assume *h*: $\text{Dom } (R \cdot S) \subseteq \text{Dom } T$
 have $R \cdot \text{Dom } S = \text{Dom } (R \cdot \text{Dom } S) \cdot (R \cdot \text{Dom } S)$
 using *d-rest-ax* by *blast*
 also have $\dots \subseteq \text{Dom } T \cdot (R \cdot \text{Dom } S)$
 by (*metis d-loc-ax h s-prod-distr subset-Un-eq*)
 also have $\dots \subseteq \text{Dom } T \cdot R$
 by (*metis d-s-id-ax s-prod-idr s-prod-isor*)
 finally show $R \cdot \text{Dom } S \subseteq \text{Dom } T \cdot R$.
qed

lemma *demod2*: $R \cdot \text{Dom } S \subseteq \text{Dom } T \cdot R \implies \text{Dom } (R \cdot S) \subseteq \text{Dom } T$
proof –
 assume *h*: $R \cdot \text{Dom } S \subseteq \text{Dom } T \cdot R$
 have $\text{Dom } (R \cdot S) = \text{Dom } (R \cdot \text{Dom } S)$
 by *auto*
 also have $\dots \subseteq \text{Dom } (\text{Dom } T \cdot R)$
 by (*metis d-add-ax h subset-Un-eq*)
 also have $\dots = \text{Dom } T \cdot \text{Dom } R$
 by *simp*
 also have $\dots \subseteq \text{Dom } T$
 by (*simp add: d-lb1*)

finally show $\text{Dom} (R \cdot S) \subseteq \text{Dom} T$.
qed

lemma $d\text{-meet-closed}$ [*simp*]: $\text{Dom} (\text{Dom} x \cap \text{Dom} y) = \text{Dom} x \cap \text{Dom} y$
by (*metis d-s-id inf-assoc s-subid-iff2*)

lemma $d\text{-add-var}$: $\text{Dom} P \cdot R \cup \text{Dom} Q \cdot R = \text{Dom} (P \cup Q) \cdot R$
by (*simp add: d-add-ax s-prod-distr*)

lemma $d\text{-interr-U}$: $\text{Dom} x \cdot U \parallel \text{Dom} y \cdot U = \text{Dom} (x \parallel y) \cdot U$
by (*metis U-nc U-par-idem cl4 d-conc6 seq-conc-subdistr subset-antisym*)

lemma $d\text{-meet}$: $\text{Dom} x \cdot z \cap \text{Dom} y \cdot z = (\text{Dom} x \cap \text{Dom} y) \cdot z$
by (*simp add: d-meet-distr-var*)

lemma $cs\text{-hom-meet}$: $\text{Dom} (x \cdot 1_\pi \cap y \cdot 1_\pi) = \text{Dom} (x \cdot 1_\pi) \cap \text{Dom} (y \cdot 1_\pi)$
by (*metis d-conc6 d-conc-inter dc-prop2 p-subid-par-eq-meet-var*)

lemma $iso3$ [*simp*]: $\text{Dom} (\text{Dom} x \cdot U) = \text{Dom} x$
by *simp*

lemma $iso4$ [*simp*]: $\text{Dom} (x \cdot 1_\pi \parallel U) \cdot U = x \cdot 1_\pi \parallel U$
by (*metis cl8-var iso3*)

lemma $iso3\text{-sharp}$ [*simp*]: $\text{Dom} (\text{Dom} (x \cap NC) \cdot NC) = \text{Dom} (x \cap NC)$
by *simp*

lemma $iso4\text{-sharp}$ [*simp*]: $\text{Dom} ((x \cap NC) \cdot NC) \cdot NC = (x \cap NC) \cdot NC$
by *simp*

3.5 Vectors

lemma $vec\text{-iff1}$:
assumes $\forall a. (\exists A. (a, A) \in R) \longrightarrow (\forall A. (a, A) \in R)$
shows $R \cdot 1_\pi \parallel U = R$
using assms by (*auto simp: mr-simp*)

lemma $vec\text{-iff2}$:
assumes $R \cdot 1_\pi \parallel U = R$
shows $(\forall a. (\exists A. (a, A) \in R) \longrightarrow (\forall A. (a, A) \in R))$
using assms apply (*clarsimp simp: mr-simp*)
by (*smt (z3) SUP-bot case-prod-conv mem-Collect-eq sup-bot-left*)

lemma $vec\text{-iff}$: $(\forall a. (\exists A. (a, A) \in R) \longrightarrow (\forall A. (a, A) \in R)) \longleftrightarrow R \cdot 1_\pi \parallel U = R$
by (*metis vec-iff1 vec-iff2*)

lemma $U\text{-par-zero}$ [*simp*]: $\{\} \cdot R \parallel U = \{\}$
by (*simp add: p-prod-comm*)

lemma $U\text{-par-}s\text{-id}$ [*simp*]: $1_\sigma \cdot 1_\pi \parallel U = U$
by *auto*

lemma $U\text{-par-}p\text{-id}$ [*simp*]: $1_\pi \cdot 1_\pi \parallel U = U$
by *auto*

lemma $U\text{-par-}nc$ [*simp*]: $NC \cdot 1_\pi \parallel U = U$
by *auto*

3.6 Up-closure and Parikh composition

definition $s\text{-prod-}pa :: ('a,'b) mrel \Rightarrow ('b,'c) mrel \Rightarrow ('a,'c) mrel$ (**infixl** $\langle\otimes\rangle$ 75)
where

$$R \otimes S = \{(a,A). (\exists B. (a,B) \in R \wedge (\forall b \in B. (b,A) \in S))\}$$

lemma $U\text{-par-}st$: $(a,A) \in R \parallel U \longleftrightarrow (\exists B. B \subseteq A \wedge (a,B) \in R)$
by (*auto simp: mr-simp*) *blast*

lemma $p\text{-id-}U$: $R \parallel U = \{(a,B). \exists A. (a,A) \in R \wedge A \subseteq B\}$
by (*clarsimp simp: mr-simp*) *blast*

lemma $ucl\text{-iff}$: $(\forall a A B. (a,A) \in R \wedge A \subseteq B \longrightarrow (a,B) \in R) \longleftrightarrow R \parallel U = R$
by (*clarsimp simp: mr-simp*) *blast*

lemma $upclosed\text{-ext}$: $R \subseteq R \parallel U$
by (*clarsimp simp: mr-simp*) *blast*

lemma $onelem$: $R \cdot S \parallel U \subseteq R \otimes (S \parallel U)$
by (*auto simp: s-prod-def p-prod-def U-def s-prod-pa-def*)

lemma $twoelem$: $R \otimes (S \parallel U) \subseteq R \cdot S \parallel U$

proof *clarify*

fix $a A$

assume $(a,A) \in R \otimes (S \parallel U)$

hence $\exists B. (a,B) \in R \wedge (\forall b \in B. (b,A) \in S \parallel U)$

by (*auto simp: s-prod-pa-def*)

hence $\exists B. (a,B) \in R \wedge (\forall b \in B. \exists C. C \subseteq A \wedge (b,C) \in S)$

by (*clarsimp simp: mr-simp*) *blast*

hence $\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. f b \subseteq A \wedge (b,f b) \in S))$

by *metis*

hence $\exists C. C \subseteq A \wedge (\exists B. (a,B) \in R \wedge (\exists f. (\forall b \in B. (b,f b) \in S) \wedge C = \bigcup ((\lambda x. f x) ` B)))$

by *clarsimp blast*

hence $\exists C. C \subseteq A \wedge (a,C) \in R \cdot S$

by (*clarsimp simp: mr-simp*)

thus $(a,A) \in (R \cdot S) \parallel U$

by (*clarsimp simp: mr-simp*) *blast*

qed

lemma *pe-pa-sim*: $R \cdot S \parallel U = R \otimes (S \parallel U)$
by (*metis antisym onelem twolem*)

lemma *pe-pa-sim-var*: $(R \parallel U) \cdot (S \parallel U) \parallel U = (R \parallel U) \otimes (S \parallel U)$
apply (*rule order.antisym*)
by (*simp add: p-prod-assoc pe-pa-sim*)+

lemma *pa-assoc1*: $((R \parallel U) \otimes (S \parallel U)) \otimes (T \parallel U) \subseteq (R \parallel U) \otimes ((S \parallel U) \otimes (T \parallel U))$
by (*clarsimp simp: p-prod-def s-prod-pa-def U-def, metis*)

lemma *up-closed-par-is-meet*: $(R \parallel U) \parallel (S \parallel U) = (R \parallel U) \cap (S \parallel U)$
by (*auto simp: mr-simp*)

lemma *U-nc-par* [*simp*]: $NC \parallel U = NC$
by (*metis c-nc-comp1 c-prod-idr nc-par-idem p-prod-distl sup-idem*)

lemma *uc-par-meet*: $(R \parallel U) \cap (S \parallel U) = R \parallel U \parallel S \parallel U$
using *p-prod-assoc up-closed-par-is-meet* **by** *blast*

lemma *uc-unc* [*simp*]: $R \parallel U \parallel R \parallel U = R \parallel U$
using *uc-par-meet* **by** *auto*

lemma *uc-interr*: $(R \parallel S) \cdot (T \parallel U) = R \cdot (T \parallel U) \parallel S \cdot (T \parallel U)$
by (*simp add: Orderings.order-eq-iff cl4 seq-conc-subdistr up-closed-par-is-meet*)

lemma *iso5* [*simp*]: $(R \cdot 1_\pi \parallel U) \cdot 1_\pi = R \cdot 1_\pi$
by (*simp add: c3*)

lemma *iso6* [*simp*]: $(R \cdot 1_\pi \parallel U) \cdot 1_\pi \parallel U = R \cdot 1_\pi \parallel U$
by *simp*

lemma *sv-hom-par*: $(R \parallel S) \cdot U = R \cdot U \parallel S \cdot U$
by (*metis U-par-idem uc-interr*)

lemma *vs-hom-meet*: $\text{Dom}((R \cdot 1_\pi \parallel U) \cap (S \cdot 1_\pi \parallel U)) = \text{Dom}(R \cdot 1_\pi \parallel U)$
 $\cap \text{Dom}(S \cdot 1_\pi \parallel U)$
by (*metis d-conc6 d-conc-inter dc-prop2 iso5 up-closed-par-is-meet*)

lemma *cv-hom-meet*: $(R \cdot 1_\pi \cap S \cdot 1_\pi) \parallel U = (R \cdot 1_\pi \parallel U) \cap (S \cdot 1_\pi \parallel U)$
by (*metis U-par-idem p-prod-assoc p-prod-comm p-subid-par-eq-meet-var uc-par-meet*)

lemma *cv-hom-par* [*simp*]: $R \parallel U \parallel S \parallel U = (R \parallel S) \parallel U$
by (*metis U-par-idem p-prod-assoc p-prod-comm*)

lemma *vc-hom-meet*: $((R \cdot 1_\pi \parallel U) \cap (S \cdot 1_\pi \parallel U)) \cdot 1_\pi = ((R \cdot 1_\pi \parallel U) \cdot 1_\pi)$
 $\cap ((S \cdot 1_\pi \parallel U) \cdot 1_\pi)$
by (*metis c4 cl8-var cv-hom-meet iso5 p-subid-par-eq-meet-var*)

```

lemma vc-hom-seq:  $((R \cdot 1_\pi \parallel U) \cdot (S \cdot 1_\pi \parallel U)) \cdot 1_\pi = ((R \cdot 1_\pi \parallel U) \cdot 1_\pi) \cdot ((S \cdot 1_\pi \parallel U) \cdot 1_\pi)$ 
proof -
  have  $((R \cdot 1_\pi \parallel U) \cdot (S \cdot 1_\pi \parallel U)) \cdot 1_\pi = (R \cdot 1_\pi \parallel U) \cdot (S \cdot 1_\pi)$ 
    by (metis c4 iso5)
  also have ... =  $R \cdot 1_\pi \parallel U \cdot (S \cdot 1_\pi)$ 
    by (metis cl8-var d-assoc1)
  also have ... =  $R \cdot 1_\pi \parallel (NC \cdot (S \cdot 1_\pi) \cup 1_\pi \cdot (S \cdot 1_\pi))$ 
    by (metis c-nc-comp1 s-prod-distr sup-commute)
  also have ... =  $R \cdot 1_\pi \parallel 1_\pi$ 
    by (metis Un-absorb1 c4 c6 s-prod-p-idl)
  thus ?thesis
    by (simp add: assoc-p-subid calculation)
qed

```

3.7 Nonterminal and terminal multirelations

```

definition tau :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel ( $\langle\tau\rangle$ ) where
   $\tau R = R \cdot \{\}$ 

```

```

definition nu :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel ( $\langle\nu\rangle$ ) where
   $\nu R = R \cap NC$ 

```

```

lemma nc-s [simp]:  $\nu 1_\sigma = 1_\sigma$ 
  using nu-def s-le-nc by auto

```

```

lemma nc-scomp-closed:  $\nu R \cdot \nu S \subseteq NC$ 
  by (simp add: nu-def nc-iff1 p-id-zero zero-assoc3)

```

```

lemma nc-scomp-closed-alt [simp]:  $\nu (\nu R \cdot \nu S) = \nu R \cdot \nu S$ 
  by (metis inf.orderE nc-scomp-closed nu-def)

```

```

lemma nc-ccomp-closed:  $\nu R \parallel \nu S \subseteq NC$ 
  unfolding nu-def by (clar simp simp: mr-simp)

```

```

lemma nc-ccomp-closed-alt [simp]:  $\nu (R \parallel \nu S) = R \parallel \nu S$ 
  unfolding nu-def by (clar simp simp: mr-simp) blast

```

```

lemma tarski-prod:  $(\nu R \cdot NC) \cdot (\nu S \cdot NC) = (\text{if } \nu S = \{\} \text{ then } \{} \text{ else } \nu R \cdot NC)$ 

```

```

proof (cases  $\nu S = \{\}$ )
  case True
  show ?thesis
    by (metis True nc-zero nu-def p-id-NC s-prod-zerol zero-assoc3)

```

```

next
  case False
  hence a:  $NC \cdot (\nu S \cdot NC) = NC$ 
  unfolding nu-def by (metis p-id-NC tarski)

```

```

have  $(\nu R \cdot NC) \cdot (\nu S \cdot NC) = (Dom(\nu R) \cdot NC) \cdot (\nu S \cdot NC)$ 
  by (simp add: nu-def)
also have ... = Dom( $\nu R$ )  $\cdot (NC \cdot (\nu S \cdot NC))$ 
  using d-assoc1 by blast
also have ... = Dom( $\nu R$ )  $\cdot NC$ 
  by (simp add: a)
also have ... =  $\nu R \cdot NC$ 
  by (simp add: nu-def)
finally have  $(\nu R \cdot NC) \cdot (\nu S \cdot NC) = \nu R \cdot NC.$ 
thus ?thesis
  using False by auto
qed

lemma nc-prod-aux [simp]:  $(\nu R \cdot NC) \cdot NC = \nu R \cdot NC$ 
apply (clar simp simp: mr-simp)
apply safe
apply clar simp
apply (smt (verit) SUP-bot-conv(1) ex-in-conv)
apply clar simp
by (smt (verit, best) SUP-bot-conv(2) UNIV-I UN-constant)

lemma nc-vec-add-closed:  $(\nu R \cdot NC \cup \nu S \cdot NC) \cdot NC = \nu R \cdot NC \cup \nu S \cdot NC$ 
by (simp add: s-prod-distr)

lemma nc-vec-par-is-meet:  $\nu R \cdot NC \parallel \nu S \cdot NC = \nu R \cdot NC \cap \nu S \cdot NC$ 
by (metis (no-types, lifting) U-nc-par cl11 nu-def p-prod-assoc
up-closed-par-is-meet)

lemma nc-vec-meet-closed:  $(\nu R \cdot NC \cap \nu S \cdot NC) \cdot NC = \nu R \cdot NC \cap \nu S \cdot NC$ 
apply (clar simp simp: nu-def mr-simp)
apply safe
apply (metis SUP-const UN-I ex-in-conv)
apply (clar simp, smt (verit) SUP-bot-conv(2) ex-in-conv)
by (clar simp, smt (verit, ccfv-threshold) SUP-bot-conv(1) SUP-const UNIV-I
all-not-in-conv)

lemma nc-vec-par-closed:  $(\nu R \cdot NC \parallel \nu S \cdot NC) \cdot NC = \nu R \cdot NC \parallel \nu S \cdot NC$ 
by (metis U-nc-par nc-prod-aux uc-interr)

lemma nc-vec-seq-closed:  $((\nu R \cdot NC) \cdot (\nu S \cdot NC)) \cdot NC = (\nu R \cdot NC) \cdot (\nu S \cdot NC)$ 
proof (cases  $\nu S = \{\}$ )
case True thus ?thesis
  by simp
next
case False thus ?thesis
  by (simp add: tarski-prod)
qed

```

lemma *iso5-sharp* [*simp*]: $(\nu R \cdot 1_\pi \parallel NC) \cdot 1_\pi = \nu R \cdot 1_\pi$
by (*simp add: c3*)

lemma *iso6-sharp* [*simp*]: $(\nu R \cdot NC \cdot 1_\pi) \parallel NC = \nu R \cdot NC$
by (*simp add: c4 nu-def*)

lemma *nsv-hom-par*: $(R \parallel S) \cdot NC = R \cdot NC \parallel S \cdot NC$
by (*simp add: cl4 seq-conc-subdistr subset-antisym*)

lemma *nvs-hom-meet*: $Dom(\nu R \cdot NC \cap \nu S \cdot NC) = Dom(\nu R \cdot NC) \cap Dom(\nu S \cdot NC)$
by (*rule antisym*) (*fastforce simp: nu-def mrd-simp*)+

lemma *ncv-hom-meet*: $R \cdot 1_\pi \cap S \cdot 1_\pi \parallel NC = (R \cdot 1_\pi \parallel NC) \cap (S \cdot 1_\pi \parallel NC)$
by (*metis c4 cl8-var d-exp-ax d-meet d-s-id-inter p-subid-par-eq-meet-var*)

lemma *ncv-hom-par*: $(R \parallel S) \parallel NC = R \parallel NC \parallel S \parallel NC$
by (*metis nc-par-idem p-prod-assoc p-prod-comm*)

lemma *nvc-hom-meet*: $(\nu R \cdot NC \cap \nu S \cdot NC) \cdot 1_\pi = (\nu R \cdot NC) \cdot 1_\pi \cap (\nu S \cdot NC) \cdot 1_\pi$
by (*rule antisym*) (*fastforce simp: nu-def mr-simp*)+

lemma *tau-int*: $\tau R \leq R$
using *p-id-zero tau-def* **by** *auto*

lemma *nu-int*: $\nu R \leq R$
by (*simp add: nu-def*)

lemma *tau-ret* [*simp*]: $\tau(\tau R) = \tau R$
by (*simp add: tau-def*)

lemma *nu-ret* [*simp*]: $\nu(\nu R) = \nu R$
by (*simp add: nu-def*)

lemma *tau-iso*: $R \leq S \implies \tau R \leq \tau S$
by (*simp add: inf.order-iff tau-def x-zero-meet-closed*)

lemma *nu-iso*: $R \leq S \implies \nu R \leq \nu S$
by (*metis Int-mono nu-def order-refl*)

lemma *tau-zero* [*simp*]: $\tau \{\} = \{\}$
by (*simp add: tau-def*)

lemma *nu-zero* [*simp*]: $\nu \{\} = \{\}$
using *nu-def* **by** *auto*

lemma *tau-s* [*simp*]: $\tau 1_\sigma = \{\}$
by (*simp add: tau-def*)

```

lemma tau-c [simp]:  $\tau 1_\pi = 1_\pi$ 
  by (simp add: tau-def)

lemma nu-c [simp]:  $\nu 1_\pi = \{\}$ 
  by (simp add: nu-def)

lemma tau-nc [simp]:  $\tau NC = \{\}$ 
  by (metis nc-iff2 order-refl tau-def)

lemma nu-nc [simp]:  $\nu NC = NC$ 
  using nu-def by auto

lemma tau-U [simp]:  $\tau U = 1_\pi$ 
  by (simp add: tau-def)

lemma nu-U [simp]:  $\nu U = NC$ 
  by (simp add: Diff-eq nu-def)

lemma tau-add [simp]:  $\tau (R \cup S) = \tau R \cup \tau S$ 
  by (simp add: tau-def x-zero-add-closed)

lemma nu-add [simp]:  $\nu (R \cup S) = \nu R \cup \nu S$ 
  by (simp add: Int-Un-distrib2 nu-def)

lemma tau-meet [simp]:  $\tau (R \cap S) = \tau R \cap \tau S$ 
  by (simp add: tau-def x-zero-meet-closed)

lemma nu-meet [simp]:  $\nu (R \cap S) = \nu R \cap \nu S$ 
  by (simp add: inf-commute inf-left-commute nu-def)

lemma tau-seq:  $\tau (R \cdot S) = \tau R \cup \nu R \cdot \tau S$ 
  unfolding nu-def tau-def
  by (metis sup-commute x-y-split zero-assoc3)

lemma tau-par [simp]:  $\tau (R \parallel S) = \tau R \parallel \tau S$ 
  by (metis U-par-zero tau-def uc-interr)

lemma nu-par-aux1:  $R \parallel \tau S = Dom(\tau S) \cdot R$ 
  by (metis p-prod-comm tau-def x-zero-prop)

lemma nu-par-aux3 [simp]:  $\nu (\nu R \parallel \tau S) = \nu R \parallel \tau S$ 
  by (metis nc-ccomp-closed-alt p-prod-comm)

lemma nu-par-aux4 [simp]:  $\nu (\tau R \parallel \tau S) = \{\}$ 
  by (metis nu-def tau-def tau-par zero-nc)

lemma nu-par:  $\nu (R \parallel S) = Dom(\tau R) \cdot \nu S \cup Dom(\tau S) \cdot \nu R \cup (\nu R \parallel \nu S)$ 
  apply (rule antisym)

```

```

apply (fastforce simp: mrd-simp nu-def tau-def)
by (auto simp: mrd-simp nu-def tau-def)

lemma sprod-tau-nu:  $R \cdot S = \tau R \cup \nu R \cdot S$ 
by (metis nu-def sup-commute tau-def x-y-split)

lemma pprod-tau-nu:  $R \parallel S = (\nu R \parallel \nu S) \cup \text{Dom}(\tau R) \cdot \nu S \cup \text{Dom}(\tau S) \cdot \nu R \cup (\tau R \parallel \tau S)$ 
by (smt (verit) nu-def nu-par sup-assoc sup-left-commute tau-def tau-par x-split-var)

lemma tau-idem [simp]:  $\tau R \cdot \tau R = \tau R$ 
by (simp add: tau-def)

lemma tau-interr:  $(R \parallel S) \cdot \tau T = R \cdot \tau T \parallel S \cdot \tau T$ 
by (simp add: tau-def cl4 seq-conc-subdistr subset-antisym)

lemma tau-le-c:  $\tau R \leq 1_\pi$ 
by (simp add: tau-def x-zero-le-c)

lemma c-le-tauc:  $1_\pi \leq \tau 1_\pi$ 
by simp

lemma x-alpha-tau [simp]:  $\nu R \cup \tau R = R$ 
by (simp add: nu-def tau-def)

lemma alpha-tau-zero [simp]:  $\nu(\tau R) = \{\}$ 
by (simp add: nu-def tau-def)

lemma tau-alpha-zero [simp]:  $\tau(\nu R) = \{\}$ 
by (simp add: nu-def tau-def)

lemma sprod-tau-nu-var [simp]:  $\nu(\nu R \cdot S) = \nu(R \cdot S)$ 
by (metis nu-add nu-def nu-ret x-y-split zero-nc)

lemma tau-s-prod [simp]:  $\tau(R \cdot S) = R \cdot \tau S$ 
by (simp add: tau-def zero-assoc3)

lemma alpha-fp:  $\nu R = R \longleftrightarrow R \cdot \{\} = \{\}$ 
by (metis inf.orderE nc-iff2 nc-zero nu-def)

lemma p-prod-tau-alpha:  $R \parallel S = (R \parallel \nu S) \cup (\nu R \parallel S) \cup (\tau R \parallel \tau S)$ 
by (smt (verit) p-prod-comm p-prod-distl sup.idem sup-assoc sup-commute x-alpha-tau)

lemma p-prod-tau-alpha-var:  $R \parallel S = (R \parallel \nu S) \cup (\nu R \parallel S) \cup \tau(R \parallel S)$ 
using p-prod-tau-alpha tau-par by blast

lemma alpha-par:  $\nu(R \parallel S) = (\nu R \parallel S) \cup (R \parallel \nu S)$ 

```

by (*metis alpha-tau-zero nc-ccomp-closed-alt nu-add p-prod-comm p-prod-tau-alpha sup-bot-right tau-par*)

lemma *alpha-tau* [*simp*]: $\nu(R \cdot \tau S) = \{\}$
by (*metis alpha-tau-zero tau-s-prod*)

lemma *nu-par-prop*: $\nu R = R \implies \nu(R \parallel S) = R \parallel S$
by (*metis nc-ccomp-closed-alt p-prod-comm*)

lemma *tau-seq-prop*: $\tau R = R \implies R \cdot S = R$
by (*metis cl6 tau-def*)

lemma *tau-seq-prop2*: $\tau R = R \implies \tau(R \cdot S) = R \cdot S$
by (*metis cl6 tau-def*)

lemma *d-nu*: $\nu(\text{Dom } R \cdot S) = \text{Dom } R \cdot \nu S$
by (*auto simp: nu-def mrd-simp*)

lemma *nu-ideal1*: $\nu R = R \implies S \leq R \implies \nu S = S$
unfolding *nu-def* **by** *blast*

lemma *tau-ideal1*: $\tau R = R \implies S \leq R \implies \tau S = S$
by (*metis dual-order.trans p-subid-iff-var tau-def*)

lemma *nu-ideal2*: $\nu R = R \implies \nu S = S \implies \nu(R \cup S) = R \cup S$
by *simp*

lemma *tau-ideal2*: $\tau R = R \implies \tau S = S \implies \tau(R \cup S) = R \cup S$
by *simp*

lemma *tau-add-precong*: $\tau R \leq \tau S \implies \tau(R \cup T) \leq \tau(S \cup T)$
by *auto*

lemma *tau-meet-precong*: $\tau R \leq \tau S \implies \tau(R \cap T) \leq \tau(S \cap T)$
by *force*

lemma *tau-par-precong*: $\tau R \leq \tau S \implies \tau(R \parallel T) \leq \tau(S \parallel T)$
by (*simp add: p-prod-isol*)

lemma *tau-seq-precongl*: $\tau R \leq \tau S \implies \tau(T \cdot R) \leq \tau(T \cdot S)$
by (*simp add: s-prod-isor*)

lemma *nu-add-precong*: $\nu R \leq \nu S \implies \nu(R \cup T) \leq \nu(S \cup T)$
by *auto*

lemma *nu-meet-precong*: $\nu R \leq \nu S \implies \nu(R \cap T) \leq \nu(S \cap T)$
by *force*

lemma *nu-seq-precongr*: $\nu R \leq \nu S \implies \nu(R \cdot T) \leq \nu(S \cdot T)$

```

by (metis nu-iso s-prod-isol sprod-tau-nu-var)

definition
  tcg R S = (τ R ≤ τ S ∧ τ S ≤ τ R)

definition
  ncg R S = (ν R ≤ ν S ∧ ν S ≤ ν R)

lemma tcg-refl: tcg R R
  by (simp add: tcg-def)

lemma tcg-trans: tcg R S ==> tcg S T ==> tcg R T
  by (meson subset-trans tcg-def)

lemma tcg-sym: tcg R S ==> tcg S R
  by (simp add: tcg-def)

lemma ncg-refl: ncg R R
  using ncg-def by blast

lemma ncg-trans: ncg R S ==> ncg S T ==> ncg R T
  by (meson ncg-def order-trans)

lemma ncg-sym: ncg R S ==> ncg S R
  by (simp add: ncg-def)

lemma tcg-alt: tcg R S = (τ R = τ S)
  using tcg-def by auto

lemma ncg-alt: ncg R S = (ν R = ν S)
  by (simp add: Orderings.order-eq-iff ncg-def)

lemma tcg-add: τ R = τ S ==> τ (R ∪ T) = τ (S ∪ T)
  by simp

lemma tcg-meet: τ R = τ S ==> τ (R ∩ T) = τ (S ∩ T)
  by simp

lemma tcg-par: τ R = τ S ==> τ (R ∥ T) = τ (S ∥ T)
  by simp

lemma tcg-seql: τ R = τ S ==> τ (T · R) = τ (T · S)
  by simp

lemma ncg-add: ν R = ν S ==> ν (R ∪ T) = ν (S ∪ T)
  by simp

lemma ncg-meet: ν R = ν S ==> ν (R ∩ T) = ν (S ∩ T)
  by simp

```

lemma *ncg-seqr*: $\nu R = \nu S \implies \nu (R \cdot T) = \nu (S \cdot T)$
by (*metis sprod-tau-nu-var*)

3.8 Powers

primrec *p-power* :: $('a, 'a) mrel \Rightarrow nat \Rightarrow ('a, 'a) mrel$ **where**
 $p\text{-power } R 0 = 1_\sigma$ |
 $p\text{-power } R (\text{Suc } n) = R \cdot p\text{-power } R n$

primrec *power-rd* :: $('a, 'a) mrel \Rightarrow nat \Rightarrow ('a, 'a) mrel$ **where**
 $power\text{-rd } R 0 = \{\}$ |
 $power\text{-rd } R (\text{Suc } n) = 1_\sigma \cup R \cdot power\text{-rd } R n$

primrec *power-sq* :: $('a, 'a) mrel \Rightarrow nat \Rightarrow ('a, 'a) mrel$ **where**
 $power\text{-sq } R 0 = 1_\sigma$ |
 $power\text{-sq } R (\text{Suc } n) = 1_\sigma \cup R \cdot power\text{-sq } R n$

lemma *power-rd-chain*: $power\text{-rd } R n \leq power\text{-rd } R (n + 1)$
apply (*induct n*)
apply *simp*
by (*smt (verit, best) Suc-eq-plus1 Un-subset-iff le-iff-sup power-rd.simps(2)*
s-prod-subdistl subsetI)

lemma *power-sq-chain*: $power\text{-sq } R n \leq power\text{-sq } R (n + 1)$
apply (*induct n*)
apply *clarsimp*
by (*smt (verit, ccfv-SIG) UnCI Un-subset-iff add.commute le-iff-sup*
plus-1-eq-Suc power-sq.simps(2) s-prod-subdistl subsetI)

lemma *pow-chain*: $p\text{-power } (1_\sigma \cup R) n \leq p\text{-power } (1_\sigma \cup R) (n + 1)$
apply (*induct n*)
apply *simp*
by (*simp add: s-prod-isor*)

lemma *pow-prop*: $p\text{-power } (1_\sigma \cup R) (n + 1) = 1_\sigma \cup R \cdot p\text{-power } (1_\sigma \cup R) n$
apply (*induct n*)
apply *simp*
by (*smt (verit, best) add.commute p-power.simps(2) plus-1-eq-Suc s-prod-distr*
s-prod-idl s-prod-subdistl subset-antisym sup.commute sup.left-commute
sup.right-idem sup-ge1)

lemma *power-rd-le-sq*: $power\text{-rd } R n \leq power\text{-sq } R n$
apply (*induct n*)
apply *simp*
by (*smt (verit, best) UnCI UnE le-iff-sup power-rd.simps(2) power-sq.simps(2)*
s-prod-subdistl subsetI)

lemma *power-sq-le-rd*: $power\text{-sq } R n \leq power\text{-rd } R (\text{Suc } n)$

```

apply (induct n)
apply simp
by (smt (verit, del-insts) UnCI UnE power-rd.simps(2) power-sq.simps(2)
s-prod-subdistI subsetI sup.absorb-iff1)

```

lemma *power-sq-power*: *power-sq R n = p-power (1_σ ∪ R) n*

```

apply (induct n)
apply simp
using pow-prop by auto

```

3.9 Star

lemma *iso-prop*: *mono (λX. S ∪ R · X)*

```

by (rule monoI, (clar simp simp: mr-simp), blast)

```

lemma *gfp-lfp-prop*: *gfp (λX. R · X) ∪ lfp (λX. S ∪ R · X) ⊆ gfp (λX. S ∪ R · X)*

```

by (simp add: lfp-le-gfp gfp-mono iso-prop)

```

definition *star* :: *('a,'a) mrel ⇒ ('a,'a) mrel* **where**

```

star R = lfp (λX. s-id ∪ R · X)

```

lemma *star-unfold*: *1_σ ∪ R · star R ≤ star R*

```

unfolding star-def using iso-prop lfp-unfold by blast

```

lemma *star-induct*: *1_σ ∪ R · S ≤ S ⇒ star R ≤ S*

```

unfolding star-def by (meson lfp-lowerbound)

```

lemma *star-refl*: *1_σ ≤ star R*

```

using star-unfold by auto

```

lemma *star-unfold-part*: *R · star R ≤ star R*

```

using star-unfold by auto

```

lemma *star-ext-aux*: *R ≤ R · star R*

```

by (metis s-prod-idr s-prod-isor star-refl)

```

lemma *star-ext*: *R ≤ star R*

```

using star-ext-aux star-unfold-part by blast

```

lemma *star-co-trans*: *star R ≤ star R · star R*

```

by (metis s-prod-idr s-prod-isor star-refl)

```

lemma *star-iso*: *R ≤ S ⇒ star R ≤ star S*

```

by (metis (no-types, lifting) le-sup-iff s-prod-distr star-induct star-refl
star-unfold-part subset-Un-eq)

```

lemma *star-unfold-eq [simp]*: *1_σ ∪ R · star R = star R*

```

by (metis iso-prop lfp-unfold star-def)

```

```

lemma nu-star1:
  assumes  $\bigwedge (R::('a,'a) mrel) (S::('a,'a) mrel) (T::('a,'a) mrel). R \cdot (S \cdot T) = (R \cdot S) \cdot T$ 
  shows  $\text{star } (R::('a,'a) mrel) \leq \text{star } (\nu R) \cdot (1_\sigma \cup \tau R)$ 
  by (smt (verit, ccfv-threshold) assms s-prod-distr s-prod-idl sprod-tau-nu star-induct star-unfold-eq subsetI sup-assoc)

lemma nu-star2:
  assumes  $\bigwedge (R::('a,'a) mrel). \text{star } R \cdot \text{star } R \leq \text{star } R$ 
  shows  $\text{star } (\nu (R::('a,'a) mrel)) \cdot (1_\sigma \cup \tau R) \leq \text{star } R$ 
  by (smt (verit) assms le-sup-iff nu-int s-prod-isol s-prod-isor star-ext star-refl star-iso subset-trans tau-int)

lemma nu-star:
  assumes  $\bigwedge (R::('a,'a) mrel). \text{star } R \cdot \text{star } R \leq \text{star } R$ 
  and  $\bigwedge (R::('a,'a) mrel) (S::('a,'a) mrel) (T::('a,'a) mrel). R \cdot (S \cdot T) = (R \cdot S) \cdot T$ 
  shows  $\text{star } (\nu (R::('a,'a) mrel)) \cdot (1_\sigma \cup \tau R) = \text{star } R$ 
  using assms nu-star1 nu-star2 by blast

lemma tau-star:  $\text{star } (\tau R) = 1_\sigma \cup \tau R$ 
  by (metis cl6 tau-def star-unfold-eq)

lemma tau-star-var:
  assumes  $\bigwedge (R::('a,'a) mrel) (S::('a,'a) mrel) (T::('a,'a) mrel). R \cdot (S \cdot T) = (R \cdot S) \cdot T$ 
  and  $\bigwedge (R::('a,'a) mrel). \text{star } R \cdot \text{star } R \leq \text{star } R$ 
  shows  $\tau (\text{star } (R::('a,'a) mrel)) = \text{star } (\nu R) \cdot \tau R$ 
  by (metis (mono-tags, lifting) assms nu-star s-prod-distr s-prod-zerol sup-bot-left tau-def tau-s)

lemma nu-star-sub:  $\text{star } (\nu R) \leq \nu (\text{star } R)$ 
proof -
  have  $a: 1_\sigma \subseteq \text{star } R$ 
    by (simp add: star-refl)
  have  $b: (R \cap NC) \cdot (\text{star } R \cap NC) \subseteq \text{star } R$ 
    by (metis nu-def nu-int s-prod-isol s-prod-isor star-unfold-part subset-trans)
  have  $c: 1_\sigma \subseteq NC$ 
    by (simp add: s-le-nc)
  have  $(R \cap NC) \cdot (\text{star } R \cap NC) \subseteq NC$ 
    by (metis nc-scomp-closed nu-def)
  thus ?thesis
    by (metis Un-least a b c le-inflI nu-def star-induct)
qed

lemma nu-star-nu [simp]:  $\nu (\text{star } (\nu R)) = \text{star } (\nu R)$ 
  using nu-int nu-star-sub by fastforce

```

```

lemma nu-star-tau [simp]:  $\nu(\text{star}(\tau R)) = 1_\sigma$ 
  using tau-star by (metis alpha-tau-zero nu-add tau-s x-alpha-tau)

lemma tau-star-tau [simp]:  $\tau(\text{star}(\tau R)) = \tau R$ 
  by (simp add: tau-star)

lemma tau-star-nu [simp]:  $\tau(\text{star}(\nu R)) = \{\}$ 
  using alpha-fp tau-def nu-star-nu by metis

lemma d-star-unfold [simp]:  $\text{Dom } S \cup \text{Dom}(R \cdot \text{Dom}(\text{star } R \cdot S)) = \text{Dom}(\text{star } R \cdot S)$ 
proof -
  have  $\text{Dom } S \cup \text{Dom}(R \cdot \text{Dom}(\text{star } R \cdot S)) = \text{Dom } S \cup \text{Dom}(R \cdot (\text{star } R \cdot \text{Dom } S))$ 
    by (metis d-loc-ax)
  also have ... =  $\text{Dom}(1_\sigma \cdot \text{Dom } S \cup (R \cdot (\text{star } R \cdot \text{Dom } S)))$ 
    by (simp add: d-add-ax)
  also have ... =  $\text{Dom}(1_\sigma \cdot \text{Dom } S \cup (R \cdot \text{star } R) \cdot \text{Dom } S)$ 
    by (metis d-comm-ax d-s-id-inter d-s-id-prop s-prod-idl test-assoc3)
  moreover have ... =  $\text{Dom}((1_\sigma \cup R \cdot \text{star } R) \cdot \text{Dom } S)$ 
    using s-prod-distr by metis
  ultimately show ?thesis
    by simp
qed

lemma d-star-sim1:
  assumes  $\bigwedge R S T. \text{Dom}(T::('a,'b) mrel) \cup (R::('a,'a) mrel) \cdot (S::('a,'a) mrel) \leq S \implies \text{star } R \cdot \text{Dom } T \leq S$ 
  shows  $(R::('a,'a) mrel) \cdot \text{Dom}(T::('a,'b) mrel) \leq \text{Dom } T \cdot (S::('a,'a) mrel) \implies \text{star } R \cdot \text{Dom } T \leq \text{Dom } T \cdot \text{star } S$ 
proof -
  fix  $R S::('a,'a) mrel$  and  $T ::('a,'b) mrel$ 
  assume  $a: R \cdot \text{Dom } T \leq \text{Dom } T \cdot S$ 
  hence  $(R \cdot \text{Dom } T) \cdot \text{star } S \leq (\text{Dom } T \cdot S) \cdot \text{star } S$ 
    by (simp add: s-prod-isol)
  hence  $b: R \cdot (\text{Dom } T \cdot \text{star } S) \leq \text{Dom } T \cdot (S \cdot \text{star } S)$ 
    by (metis d-assoc1 d-s-id-ax inf.order-iff test-assoc1)
  hence  $R \cdot (\text{Dom } T \cdot \text{star } S) \leq \text{Dom } T \cdot \text{star } S$ 
    by (meson order-trans s-prod-isor star-unfold-part)
  hence  $\text{Dom } T \cup R \cdot (\text{Dom } T \cdot \text{star } S) \leq \text{Dom } T \cdot \text{star } S$ 
    by (metis le-supI s-prod-idr s-prod-isor star-refl)
  thus  $\text{star } R \cdot \text{Dom } T \leq \text{Dom } T \cdot \text{star } S$ 
    using assms by presburger
qed

lemma d-star-induct:
  assumes  $\bigwedge R S T. \text{Dom}(T::('a,'b) mrel) \cup (R::('a,'a) mrel) \cdot (S::('a,'a) mrel) \leq S \implies \text{star } R \cdot \text{Dom } T \leq S$ 
  shows  $\text{Dom}((R::('a,'a) mrel) \cdot (S::('a,'a) mrel)) \leq \text{Dom } S \implies \text{Dom}(\text{star } R \cdot$ 

```

$S) \leq \text{Dom } S$
by (metis assms d-star-sim1 dc-prop2 demod1 demod2)

3.10 Omega

```

definition omega :: ('a,'a) mrel  $\Rightarrow$  ('a,'a) mrel where
  omega R  $\equiv$  gfp ( $\lambda X$ . R  $\cdot$  X)

lemma om-unfold: omega R  $\leq$  R  $\cdot$  omega R
  unfolding omega-def
  by (metis (no-types, lifting) gfp-least gfp-upperbound order-trans s-prod-isor)

lemma om-coinduct: S  $\leq$  R  $\cdot$  S  $\implies$  S  $\leq$  omega R
  unfolding omega-def by (simp add: gfp-upperbound omega-def)

lemma om-unfold-eq [simp]: R  $\cdot$  omega R = omega R
  by (rule antisym) (auto simp: om-coinduct om-unfold s-prod-isor)

lemma om-iso: R  $\leq$  S  $\implies$  omega R  $\leq$  omega S
  by (metis om-coinduct s-prod-isol om-unfold-eq)

lemma zero-om [simp]: omega {} = {}
  using om-unfold-eq s-prod-zerol by blast

lemma s-id-om [simp]: omega 1 $_{\sigma}$  = U
  by (simp add: U-def eq-iff om-coinduct)

lemma p-id-om [simp]: omega 1 $_{\pi}$  = 1 $_{\pi}$ 
  using om-unfold-eq s-prod-p-idl by blast

lemma nc-om [simp]: omega NC = U
  by (metis dual-order.refl nc-U om-coinduct s-id-om s-prod-idl subset-antisym)

lemma U-om [simp]: omega U = U
  by (metis U-U dual-order.refl om-coinduct s-id-om s-prod-idl subset-antisym)

lemma tau-om1:  $\tau$  R  $\leq$   $\tau$  (omega R)
  by (metis om-unfold-eq order-refl sup.boundedE tau-seq)

lemma tau-om2 [simp]: omega ( $\tau$  R) =  $\tau$  R
  by (metis cl6 om-unfold-eq tau-def)

lemma tau-om3: omega ( $\tau$  R)  $\leq$   $\tau$  (omega R)
  by (simp add: tau-om1)

lemma om-nu-tau: omega ( $\nu$  R)  $\cup$  star ( $\nu$  R)  $\cdot$   $\tau$  R  $\leq$  omega R
  proof -
    have omega ( $\nu$  R)  $\cup$  star ( $\nu$  R)  $\cdot$   $\tau$  R = omega ( $\nu$  R)  $\cup$  (1 $_{\sigma}$   $\cup$   $\nu$  R  $\cdot$  star ( $\nu$  R))  $\cdot$   $\tau$  R

```

```

    by auto
also have ... = omega (ν R) ∪ τ R ∪ ν R · star (ν R) · τ R
  using s-prod-distr s-prod-idl by blast
also have ... = τ R ∪ ν R · omega (ν R) ∪ ν R · (star (ν R) · τ R)
  by (simp add: cl5 sup-commute tau-def)
also have ... ≤ τ R ∪ ν R · (omega (ν R) ∪ star (ν R) · τ R)
  by (smt (verit) Un-subset-iff s-prod-isor sup.cobounded2 sup.coboundedI2
sup-commute)
also have ... = R · (omega (ν R) ∪ star (ν R) · τ R)
  using sprod-tau-nu by blast
finally show ?thesis
  using om-coinduct by blast
qed

end

```

4 Multirelational Properties of Power Allegories

theory Power-Allegories-Multirelations

imports Multirelations-Basics

begin

We start with random little properties.

lemma eta-s-id: $\eta = s\text{-id}$
unfolding s-id-def eta-set **by** force

lemma Lambda-empty [simp]: $\Lambda \{\} = p\text{-id}$
unfolding Lambda-def p-id-def **by** blast

lemma alpha-pid [simp]: $\alpha p\text{-id} = \{\}$
unfolding alpha-def epsilon-off-def p-id-def **by** force

4.1 Peleg lifting

definition plift :: ('a,'b) mrel ⇒ ('a set,'b set) rel (⟨-_*⟩ [1000] 999) **where**
 $R_* = \{(A,B). \exists f. (\forall a \in A. (a,f(a)) \in R) \wedge B = \bigcup(f ` A)\}$

lemma pcomp-plift: $R \cdot S = R ; S_*$
unfolding s-prod-def plift-def relcomp-unfold **by** simp

lemma det-plift-klift: deterministic $R \implies R_* = (R)_P$
unfolding deterministic-set plift-def klift-set-var
apply (simp add: set-eq-iff)
apply safe
by metis+

lemma plift-ext2 [simp]: $\eta ; R_* = R$

```

by (metis eta-s-id pcomp-plift s-prod-idl)

lemma plifttext-3 [simp]:  $\eta_* = Id$ 
  by (simp add: det-eta det-plift-klift)

lemma d-dom-plift:  $(\text{Dom } R)_* = \text{dom } (R_*)$ 
  unfolding Dom-def dom-set plift-def
  apply clarsimp
  apply safe
  apply (metis (full-types) UN-singleton image-cong)
  by (metis UN-singleton)

lemma d-pid-plift:  $(\text{Dom } R)_* \subseteq Id$ 
  by (metis d-dom-plift d-idem dom-subid inf.absorb-iff2)

lemma d-plift-sub:  $A \subseteq B \implies (B,B) \in (\text{Dom } R)_* \implies (A,A) \in (\text{Dom } R)_*$ 
  by (smt (z3) Pair-inject UN-singleton Dom-def mem-Collect-eq plift-def
      split-conv subsetD)

lemma plift-empty:  $(\{\}, A) \in R_* \longleftrightarrow A = \{\}$ 
  using plift-def by auto

lemma univ-plift-klift:
  assumes univalent R
  shows  $R_* = (\text{Dom } R)_* ; (R)_{\mathcal{P}}$ 
proof -
  have  $\forall A B . (A,B) \in R_* \longleftrightarrow (A,B) \in (\text{Dom } R)_* ; (R)_{\mathcal{P}}$ 
  proof (intro allI, rule iffI)
    fix A B
    assume 1:  $(A,B) \in R_*$ 
    hence  $(A,B) \in (R)_{\mathcal{P}}$ 
      unfolding klift-set-var
      apply clarsimp
      by (smt (z3) Collect-cong Pair-inject Union-eq assms case-prodE image-iff
          mem-Collect-eq plift-def univalent-set)
    thus  $(A,B) \in (\text{Dom } R)_* ; (R)_{\mathcal{P}}$ 
      using 1 d-dom-plift dom-set by fastforce
  next
    fix A B
    assume  $(A,B) \in (\text{Dom } R)_* ; (R)_{\mathcal{P}}$ 
    from this obtain C where 2:  $(A,C) \in (\text{Dom } R)_* \wedge (C,B) \in (R)_{\mathcal{P}}$ 
      by auto
    from this obtain f where 3:  $(\forall a \in A . (a,f(a)) \in \text{Dom } R) \wedge C = \bigcup(f ` A)$ 
      by (smt (verit) Pair-inject case-prodE mem-Collect-eq plift-def)
    hence  $\forall a \in A . \exists D . (a,D) \in R$ 
      by (simp add: Dom-def)
    from this obtain g where 4:  $\forall a \in A . (a,g(a)) \in R$ 
      by metis
    hence  $\forall a \in A . f(a) = \{a\}$ 

```

```

using 3 by (simp add: Dom-def)
hence A = C
  using 2 3 by (smt (verit) Pair-inject UN-singleton case-prodE image-cong
mem-Collect-eq plift-def)
  hence (A,B) ∈ (R)ₚ
    using 2 by simp
    thus (A,B) ∈ R*
      unfolding plift-def klift-set-var
      apply clar simp
      apply (rule exI[where ?x=g])
      using 4 by (smt (verit, ccfv-SIG) Collect-cong assms image-def
univalent-set)
      qed
      thus R* = (Dom R)* ; (R)ₚ
        by force
    qed
lemma plift-ext1:
  assumes univalent f
  shows (R ; f*)* = R* ; f*
proof -
  have ∀ A B . (A,B) ∈ (R ; (Dom f)* ; (f)ₚ)* ↔ (A,B) ∈ R* ; (Dom f)* ; (f)ₚ
  proof (intro allI, rule iffI)
    fix A B
    assume 1: (A,B) ∈ (R ; (Dom f)* ; (f)ₚ)*
    from this obtain g where 2: (∀ a ∈ A. (a,g(a)) ∈ R ; (Dom f)* ; (f)ₚ) ∧ B
    = ∪(g ` A)
      by (smt (verit) Pair-inject case-prodE mem-Collect-eq plift-def)
    hence ∀ a ∈ A . ∃ C D . (a,C) ∈ R ∧ (C,D) ∈ (Dom f)* ∧ (D,g(a)) ∈ (f)ₚ
      by (meson relcompEpair)
    hence ∀ a ∈ A . ∃ C . (a,C) ∈ R ∧ (C,C) ∈ (Dom f)* ∧ (C,g(a)) ∈ (f)ₚ
      using d-pid-plift by fastforce
    from this obtain h where 3: ∀ a ∈ A . (a,h a) ∈ R ∧ (h a,h a) ∈ (Dom f)*
    ∧ (h a,g(a)) ∈ (f)ₚ
      by metis
    let ?h = ∪(h ` A)
    have 4: (A,?h) ∈ R*
      using 3 plift-def by fastforce
    have 5: (?h,?h) ∈ (Dom f)*
      using 3
      unfolding plift-def Dom-def
      apply clar simp
      by (metis UN-extend-simps(9) UN-singleton)
    have (?h,B) ∈ (f)ₚ
      using 2 3
      unfolding klift-set
      by auto
    thus (A,B) ∈ R* ; (Dom f)* ; (f)ₚ
      using 4 5 by blast

```

```

next
fix A B
assume (A,B) ∈ R* ; (Dom f)* ; (f)P
from this obtain C D where 6: (A,C) ∈ R* ∧ (C,D) ∈ (Dom f)* ∧ (D,B)
    ∈ (f)P
by blast
hence 7: C = D
using d-pid-plift by auto
from 6 obtain g where 8: (∀ a ∈ A. (a,g(a)) ∈ R) ∧ C = ∪(g ` A)
by (smt (verit) Pair-inject case-prodE mem-Collect-eq plift-def)
hence 9: ∀ a ∈ A. (g(a),g(a)) ∈ (Dom f)*
using 6 7 by (metis d-plift-sub UN-iff subsetI)
let ?h = λa . ∪(f `` g a)
have ∀ a ∈ A. (g(a),?h a) ∈ (f)P
using 6 by (simp add: klift-set)
hence 10: ∀ a ∈ A. (a,?h a) ∈ R ; (Dom f)* ; (f)P
using 8 9 by blast
have B = ∪(f `` D)
using 6 klift-set by fastforce
hence 11: B = (∪ a ∈ A . ?h a)
using 7 8 image-empty by blast
show (A,B) ∈ (R ; (Dom f)* ; (f)P)*
apply (subst plift-def)
apply clar simp
apply (rule exI[where ?x=?h])
using 10 11 by simp
qed
hence (R ; (Dom f)* ; (f)P)* = R* ; (Dom f)* ; (f)P
by force
thus ?thesis
by (metis (no-types, opaque-lifting) assms univ-plift-klift O-assoc)
qed

lemma plift-assoc-univ: univalent f  $\implies$  (R · S) · f = R · (S · f)
by (simp add: pcomp-plift O-assoc plift-ext1)

lemma Lambda-funct:  $\Lambda$  (R ; S) =  $\Lambda$  R ·  $\Lambda$  S
by (simp add: Lambda-pow det-lambda det-plift-klift klift-def pcomp-plift)

lemma eta-funct: R ; S ; η = (R ; η) · (S ; η)
proof -
have (R ; η) · (S ; η) = R ; η ; (S ; η)*
by (simp add: pcomp-plift)
also have ... = R ; (η · (S ; η))
by (simp add: O-assoc pcomp-plift)
also have ... = R ; S ; η
by (simp add: O-assoc pcomp-plift)
finally show ?thesis..
qed

```

```

lemma alpha-funct-det: deterministic R  $\implies$  deterministic S  $\implies$   $\alpha(R \cdot S) = \alpha R ; \alpha S$ 
by (metis Lambda-epsilon-off-up2 Lambda-funct alpha-Lambda-canc)

lemma pcomp-det: deterministic S  $\implies$   $R \cdot S = R ; (S)_P$ 
by (simp add: det-plift-klift pcomp-plift)

lemma pcomp-det2: deterministic R  $\implies$  deterministic S  $\implies$   $(R \cdot S)_P = (R)_P ; (S)_P$ 
by (simp add: klift-ext1 pcomp-det)

lemma pcomp-alpha:  $\alpha(R \cdot S) = R ; \alpha((S)_*)$ 
by (simp add: pcomp-plift)

```

4.2 Fusion and fission

```

definition fus :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel where
  fus R =  $\Lambda(\alpha R)$ 

definition fis :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel where
  fis R =  $\alpha R ; \eta$ 

lemma fus-set: fus R =  $\{(a,B) \mid a \in B. B = \bigcup(\text{Image } R \{a\})\}$ 
  unfolding fus-def Lambda-def alpha-set by force

lemma fis-set: fis R =  $\{(a,\{b\}) \mid a \in b. b \in \bigcup(\text{Image } R \{a\})\}$ 
  unfolding fis-def alpha-set eta-set relcomp-unfold by force

lemma fis-det-comp: deterministic R  $\implies$  deterministic S  $\implies$  fis (R · S) = fis R
  · fis S
  by (simp add: alpha-funct-det eta-funct fis-def)

lemma fis-fix-det: deterministic R = (fus R = R)
  by (metis Lambda-alpha-canc det-lambda fus-def)

```

4.3 Galois connections for multirelations

```

lemma sub-subbh:  $R \subseteq S \implies R \subseteq S ; (\text{epsilon-off} // \text{epsilon-off})$ 
  by (metis R-O-Id alpha-def alpha-epsilon-off lres-galois order-refl relcomp-mono)

lemma alpha-Lambda-galois:  $(\alpha R \subseteq S) = (R \subseteq \Lambda S ; (\text{epsilon-off} // \text{epsilon-off}))$ 
proof -
  have a:  $(\alpha R \subseteq S) = (R \subseteq S // \text{epsilon-off})$ 
    by (simp add: alpha-def lres-galois)
  have S // epsilon-off =  $(\Lambda S ; \text{epsilon-off}) // \text{epsilon-off}$ 
    by (metis alpha-Lambda-canc alpha-def)
  also have ... =  $\Lambda S ; (\text{epsilon-off} // \text{epsilon-off})$ 
    by (simp add: det-lambda det-lres)
  finally have S // epsilon-off =  $\Lambda S ; (\text{epsilon-off} // \text{epsilon-off})$ 

```

```

. . .
thus ?thesis
  using a by presburger
qed

lemma alpha-Lambda-galois-set: ( $\alpha R \subseteq S$ ) = ( $R \subseteq \{(a,A). \exists B. (a,B) \in \Lambda S \wedge A \subseteq B\}$ )
  unfolding alpha-set Lambda-def by blast

lemma epsilonoff-eta-lres: epsilonoff ;  $\eta \subseteq \text{epsilonoff} // \text{epsilonoff}$ 
proof -
  have epsilonoff ;  $\eta$  ; epsilonoff = epsilonoff ;  $\alpha (\Lambda \text{Id})$ 
    by (simp add: O-assoc alpha-def eta-def)
  also have ... = epsilonoff
    by simp
  finally have epsilonoff ;  $\eta$  ; epsilonoff = epsilonoff.
  thus ?thesis
    by (smt (verit) lres-galois order-refl)
qed

lemma eta-alpha-galois: ( $R ; \eta \subseteq S ; (\text{epsilonoff} // \text{epsilonoff})$ ) = ( $R \subseteq \alpha S$ )
proof
  assume  $R ; \eta \subseteq S ; (\text{epsilonoff} // \text{epsilonoff})$ 
  hence  $R \subseteq S ; (\text{epsilonoff} // \text{epsilonoff}) ; \text{epsilonoff}$ 
    by (metis R-O-Id alpha-def alpha-eta alpha-ord-pres alpha-relcomp)
  thus  $R \subseteq \alpha S$ 
    by (simp add: O-assoc alpha-def)
next
  assume  $R \subseteq \alpha S$ 
  hence  $R ; \eta \subseteq \alpha S ; \eta$ 
    by (simp add: relcomp-mono)
  hence  $R ; \eta \subseteq S ; \text{epsilonoff} ; \eta$ 
    by (simp add: alpha-def)
  thus  $R ; \eta \subseteq S ; (\text{epsilonoff} // \text{epsilonoff})$ 
    using epsilonoff-eta-lres in-mono by fastforce
qed

lemma eta-alpha-galois-set: ( $R ; \eta \subseteq \{(a,A). \exists B. (a,B) \in S \wedge A \subseteq B\}$ ) = ( $R \subseteq \alpha S$ )
  unfolding eta-set alpha-set by auto

lemma Lambda-iso:  $R \subseteq S \implies \Lambda R \subseteq \Lambda S ; (\text{epsilonoff} // \text{epsilonoff})$ 
  by (metis alpha-Lambda-canc alpha-Lambda-galois)

lemma eta-iso:  $R \subseteq S \implies R ; \eta \subseteq S ; \eta ; (\text{epsilonoff} // \text{epsilonoff})$ 
  by (simp add: eta-alpha-galois)

lemma alpha-iso:  $R \subseteq S ; (\text{epsilonoff} // \text{epsilonoff}) \implies \alpha R \subseteq \alpha S$ 
  by (metis (no-types, lifting) alpha-def alpha-ord-pres alpha-relcomp conv-Omega)

```

conv-Omega-epsiloff)

lemma *Lambda-canc-dcl*: $R \subseteq \Lambda (\alpha R) ; (\text{epsiloff} // \text{epsiloff})$
using *alpha-Lambda-galois* **by** *blast*

lemma *eta-canc-dcl*: $\alpha R ; \eta \subseteq R ; (\text{epsiloff} // \text{epsiloff})$
by (*simp add: eta-alpha-galois*)

lemma *alpha-surj*: *surj* α
using *alpha-Lambda-canc* **by** *blast*

lemma *Lambda-inj*: *inj* Λ
by (*metis alpha-Lambda-canc injI*)

lemma *eta-inj*: *inj* $(\lambda x. x ; \eta)$
by (*metis alpha-eta-id injI*)

lemma *fus-least-odet*:
assumes $\Lambda (\alpha S) = S$
and $R \subseteq S ; (\text{epsiloff} // \text{epsiloff})$
shows $\Lambda (\alpha R) \subseteq S ; (\text{epsiloff} // \text{epsiloff})$
proof –
have $\alpha R \subseteq \alpha S$
by (*simp add: alpha-iso assms(2)*)
hence $\Lambda (\alpha R) \subseteq \Lambda (\alpha S) ; (\text{epsiloff} // \text{epsiloff})$
by (*simp add: Lambda-iso*)
thus *?thesis*
using *assms(1)* **by** *auto*
qed

lemma *fis-greatest-ide*:
assumes $\alpha S ; \eta = S$
and $S \subseteq R ; (\text{epsiloff} // \text{epsiloff})$
shows $S \subseteq \alpha R ; \eta ; (\text{epsiloff} // \text{epsiloff})$
proof –
have $\alpha S \subseteq \alpha R$
by (*simp add: alpha-iso assms(2)*)
hence $\alpha S ; \eta \subseteq \alpha R ; \eta ; (\text{epsiloff} // \text{epsiloff})$
by (*simp add: eta-iso*)
thus *?thesis*
using *assms(1)* **by** *auto*
qed

lemma *fis-fus-galois*: $(\alpha R ; \eta \subseteq S ; (\text{epsiloff} // \text{epsiloff})) = (R \subseteq \Lambda (\alpha S) ; (\text{epsiloff} // \text{epsiloff}))$
by (*simp add: alpha-Lambda-galois eta-alpha-galois*)

4.4 Properties of alpha, fission and fusion

lemma *alpha-lax*: $\alpha(R \cdot S) \subseteq \alpha R ; \alpha S$
unfolding *alpha-def s-prod-def epsiloff-def relcomp-unfold* **by** *blast*

lemma *alpha-down* [simp]: $\alpha(R ; \Omega^\sim) = \alpha R$
by (*metis alpha-def alpha-relcomp conv-Omega-epsiloff*)

lemma *fis-fis* [simp]: $fis \circ fis = fis$
unfolding *fun-eq-iff fis-def* **by** *simp*

lemma *fus-fus* [simp]: $fus \circ fus = fus$
unfolding *fun-eq-iff fus-def* **by** *simp*

lemma *fis-fus* [simp]: $fis \circ fus = fis$
unfolding *fun-eq-iff fus-def fis-def* **by** *simp*

lemma *fus-fis* [simp]: $fus \circ fis = fus$
unfolding *fun-eq-iff fus-def fis-def* **by** *simp*

lemma *fis-alpha*: $fis R \cdot S = \alpha R ; S$
by (*simp add: O-assoc fis-def pcomp-plift*)

lemma *fis-lax*: $fis(R \cdot S) \subseteq fis R \cdot fis S$
by (*metis fis-def alpha-lax eta-funct relcomp-mono subsetI*)

lemma *klift-fus*: $(R)_P = fus(\text{epsiloff} ; R)$
by (*simp add: alpha-def fus-def klift-var*)

lemma *fus-eta-klift*: $fus R = \eta ; (R)_P$
by (*metis Id-O-R Lambda-pow eta-def fus-def klift-def*)

lemma *fus-Lambda-mu*: $fus R = \Lambda R ; \mu$
by (*simp add: fus-def lambda-alpha-mu*)

4.5 Properties of fusion, fission, nu and tau

lemma *alpha-tau* [simp]: $\alpha(\tau R) = \{\}$
by (*metis alpha-ord-pres alpha-pid subset-empty tau-le-c*)

lemma *alpha-nu* [simp]: $\alpha(\nu R) = \alpha R$
unfolding *alpha-def nu-def epsiloff-def U-def p-id-def relcomp-unfold* **by** *force*

lemma *nu-fis* [simp]: $\nu(fis R) = fis R$
by (*metis alpha-fp empty-iff equalsOI fis-alpha relcompE*)

lemma *nu-fis-var*: $\nu(fis R) = fis(\nu R)$
by (*metis alpha-nu fis-def nu-fis*)

lemma *tau-fis* [simp]: $\tau(fis R) = \{\}$

by (*metis nu-fis tau-alpha-zero*)

Properties of tests and domain

lemma *subid-plift*: $(P \cap \eta)_* = \{(A, A) | A. \forall a \in A. (a, \{a\}) \in (P \cap \eta)\}$
unfolding *plift-def eta-set* **by** *safe auto*

lemma *U-subid*: $R ; (P \cap \eta)_* = R \cap U ; (P \cap \eta)_*$
unfolding *plift-def U-def eta-set relcomp-unfold*
apply *safe*
apply *force*
apply *blast*
by *force*

lemma *subid-plift-down*: $U ; (P \cap \eta)_* ; \Omega^\sim = U ; (P \cap \eta)_*$
unfolding *U-def relcomp-unfold plift-def Omega-set eta-set converse-def*
apply *safe*
apply *clar simp*
apply (*metis IntE UN-singleton inf.orderE*)
by *blast*

lemma *nu-subid-plift*: $\nu (R ; (P \cap \eta)_*) = \nu R ; (P \cap \eta)_*$
unfolding *nu-def relcomp-unfold plift-def U-def p-id-def eta-set* **by** *safe fastforce+*

lemma *dom-fis1*: $\text{dom} (\text{fis } R) = \text{dom} (\alpha R)$
unfolding *dom-set fis-set alpha-set* **by** *blast*

lemma *dom-fis2*: $\text{dom} (\text{fis } R) = \text{dom} (\alpha (\nu R))$
by (*simp add: dom-fis1*)

lemma *dom-fis3*: $\text{dom} (\text{fis } R) = \text{dom} (\nu R)$
unfolding *dom-set fis-set nu-def U-def p-id-def* **by** *safe fastforce+*

lemma *dom-fis4*: $\text{dom} (\text{fis } R) = \text{dom} (\nu (\text{fus } R))$
by (*metis comp-eq-dest-lhs dom-fis3 fis-fus*)

lemma *dom-alpha*: $\text{dom} (\alpha R ; (P \cap \eta)) = \text{dom} (\nu (R ; \Omega^\sim) ; (P \cap \eta)_*)$
unfolding *dom-set alpha-set eta-set Omega-set plift-def nu-def relcomp-unfold U-def p-id-def*
apply *safe*
apply (*clar simp, metis empty-iff empty-subsetI insert-subset singletonD singletonI UN-singleton*)
by *clar simp fastforce*

4.6 Box and diamond

definition *box* :: $('a, 'b) \text{ mrel} \Rightarrow ('b \text{ set}, 'a \text{ set}) \text{ rel where}$
box $R = rbox (\alpha R)$

definition *dia* :: $('a, 'b) \text{ mrel} \Rightarrow ('b \text{ set}, 'a \text{ set}) \text{ rel where}$

```

dia R =  $\mathcal{P}((\alpha R)^\sim)$ 

lemma box-set: box R = {(B, A). A = {a.  $\forall C. (a, C) \in R \rightarrow C \subseteq B\}}}
  unfolding box-def rbox-set alpha-set by force

lemma dia-set: dia R = {(B, A). A = {a.  $\exists C. (a, C) \in R \wedge C \cap B \neq \{\}\}}$ }
  unfolding dia-def pow-set Image-def alpha-set converse-def by force

lemma box-Omega: box R =  $\Lambda(\Omega^\sim // R)$ 
  unfolding box-set Lambda-def lres-def Omega-set by auto

end
theory Multirelations

imports Power-Allegories-Multirelations

begin

lemma nonempty-set-card:
  assumes finite S
  shows S ≠ {}  $\longleftrightarrow$  card S ≥ 1
  using assms card-0-eq by fastforce

no-notation one-class.one ⟨1⟩
no-notation times-class.times (infixl ⟨*⟩ 70)

no-notation rel-fdia ⟨(|)-⟩ [61,81] 82)
no-notation rel-bdia ⟨⟨(|)-⟩ [61,81] 82)
no-notation rel-fbox ⟨(|)-⟩ [61,81] 82)
no-notation rel-bbox ⟨⟨(|)-⟩ [61,81] 82)

declare s-prod-pa-def [mr-simp]

notation s-prod (infixl ⟨*⟩ 70)
notation s-id ⟨1⟩

lemma sp-oi-subdist:
  (P ∩ Q) * (R ∩ S) ⊆ P * R
  unfolding s-prod-def by blast

lemma sp-oi-subdist-2:
  (P ∩ Q) * (R ∩ S) ⊆ (P * R) ∩ (Q * S)
  unfolding s-prod-def by blast$ 
```

5 Inner Structure

5.1 Inner union, inner intersection and inner complement

abbreviation inner-union (infixl ⟨UU⟩ 65)

where *inner-union* \equiv *p-prod*

definition *inner-intersection* :: ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('a,'b) mrel (**infixl** $\langle \cap \cap \rangle$ 65) **where**
 $R \cap \cap S \equiv \{ (a,B) . \exists C D . B = C \cap D \wedge (a,C) \in R \wedge (a,D) \in S \}$

definition *inner-complement* :: ('a,'b) mrel \Rightarrow ('a,'b) mrel ($\langle \sim \rightarrow \rangle$ [80] 80) **where**
 $\sim R \equiv \{ (a,B) . (a,\sim B) \in R \}$

abbreviation *iu-unit* ($\langle 1_{\cup \cup} \rangle$)
where $1_{\cup \cup} \equiv$ *p-id*

definition *ii-unit* :: ('a,'a) mrel ($\langle 1_{\cap \cap} \rangle$)
where $1_{\cap \cap} \equiv \{ (a,UNIV) | a . True \}$

declare *inner-intersection-def* [mr-simp] *inner-complement-def* [mr-simp]
ii-unit-def [mr-simp]

lemma *iu-assoc*:
 $(R \cup \cup S) \cup \cup T = R \cup \cup (S \cup \cup T)$
by (simp add: *p-prod-assoc*)

lemma *iu-commute*:
 $R \cup \cup S = S \cup \cup R$
by (simp add: *p-prod-comm*)

lemma *iu-unit*:
 $R \cup \cup 1_{\cup \cup} = R$
by simp

lemma *ii-assoc*:
 $(R \cap \cap S) \cap \cap T = R \cap \cap (S \cap \cap T)$
apply (clar simp simp: mr-simp)
by (metis (no-types, opaque-lifting) semilattice-inf-class.inf-assoc)

lemma *ii-commute*:
 $R \cap \cap S = S \cap \cap R$
by (auto simp: mr-simp)

lemma *ii-unit* [simp]:
 $R \cap \cap 1_{\cap \cap} = R$
by (simp add: mr-simp)

lemma *pa-ic*:
 $\sim(R \otimes \sim S) = R \otimes S$
by (clar simp simp: mr-simp)

lemma *ic-involutive* [simp]:
 $\sim \sim R = R$

```

by (simp add: mr-simp)

lemma ic-injective:
 $\sim R = \sim S \implies R = S$ 
by (metis ic-involutive)

lemma ic-antidist-iu:
 $\sim(R \cup\cup S) = \sim R \cap\cap \sim S$ 
apply (clar simp simp: mr-simp)
by (metis (no-types, opaque-lifting) compl-sup double-compl)

lemma ic-antidist-ii:
 $\sim(R \cap\cap S) = \sim R \cup\cup \sim S$ 
by (metis ic-antidist-iu ic-involutive)

lemma ic-iu-unit [simp]:
 $\sim 1_{\cup\cup} = 1_{\cap\cap}$ 
unfolding ii-unit-def p-id-def inner-complement-def by force

lemma ic-ii-unit [simp]:
 $\sim 1_{\cap\cap} = 1_{\cup\cup}$ 
by (metis ic-involutive ic-iu-unit)

lemma ii-unit-split-iu [simp]:
 $1 \cup\cup \sim 1 = 1_{\cap\cap}$ 
by (force simp: mr-simp)

lemma aux-1:
 $B = \{a\} \cap D \implies \neg D = \{a\} \implies B = \{\}$ 
by auto

lemma iu-unit-split-ii [simp]:
 $1 \cap\cap \sim 1 = 1_{\cup\cup}$ 
by (metis ic-antidist-iu ic-ii-unit ic-involutive ii-commute ii-unit-split-iu)

lemma iu-right-dist-ou:
 $(R \cup S) \cup\cup T = (R \cup\cup T) \cup (S \cup\cup T)$ 
unfolding p-prod-def by auto

lemma ii-right-dist-ou:
 $(R \cup S) \cap\cap T = (R \cap\cap T) \cup (S \cap\cap T)$ 
by (auto simp: mr-simp)

lemma iu-left-isotone:
 $R \subseteq S \implies R \cup\cup T \subseteq S \cup\cup T$ 
by (metis iu-right-dist-ou subset-Un-eq)

lemma iu-right-isotone:
 $R \subseteq S \implies T \cup\cup R \subseteq T \cup\cup S$ 

```

```

by (simp add: iu-commute iu-left-isotone)

lemma iu-isotone:
   $R \subseteq S \implies P \subseteq Q \implies R \cup P \subseteq S \cup Q$ 
  by (meson dual-order.trans iu-left-isotone iu-right-isotone)

lemma ii-left-isotone:
   $R \subseteq S \implies R \cap T \subseteq S \cap T$ 
  by (metis ii-right-dist-ou subset-Un-eq)

lemma ii-right-isotone:
   $R \subseteq S \implies T \cap R \subseteq T \cap S$ 
  by (simp add: ii-commute ii-left-isotone)

lemma ii-isotone:
   $R \subseteq S \implies P \subseteq Q \implies R \cap P \subseteq S \cap Q$ 
  by (meson ii-left-isotone ii-right-isotone order-trans)

lemma iu-right-subdist-ii:
   $(R \cap S) \cup T \subseteq (R \cup T) \cap (S \cup T)$ 
  apply (clar simp simp: mr-simp)
  by (metis sup-inf-distrib2)

lemma ii-right-subdist-iu:
   $(R \cup S) \cap T \subseteq (R \cap T) \cup (S \cap T)$ 
  apply (clar simp simp: mr-simp)
  by (metis inf-sup-distrib2)

lemma ic-isotone:
   $R \subseteq S \implies \sim R \subseteq \sim S$ 
  by (simp add: inner-complement-def subset-eq)

lemma ic-bot [simp]:
   $\sim \{ \} = \{ \}$ 
  by (simp add: mr-simp)

lemma ic-top [simp]:
   $\sim U = U$ 
  by (auto simp: mr-simp)

lemma ic-dist-ou:
   $\sim(R \cup S) = \sim R \cup \sim S$ 
  by (auto simp: mr-simp)

lemma ic-dist-oi:
   $\sim(R \cap S) = \sim R \cap \sim S$ 
  by (auto simp: mr-simp)

lemma ic-dist-oc:

```

$\sim -R = -(\sim R)$
by (auto simp: mr-simp)

lemma ii-sub-idempotent:
 $R \subseteq R \cap R$
unfolding inner-intersection-def **by** force

definition inner-Union :: ('i \Rightarrow ('a,'b) mrel) \Rightarrow 'i set \Rightarrow ('a,'b) mrel ($\langle \bigcup \bigcup - \rangle$ [80,80] 80) **where**
 $\bigcup \bigcup X|I \equiv \{ (a,B) . \exists f . B = (\bigcup i \in I . f i) \wedge (\forall i \in I . (a,f i) \in X i) \}$

definition inner-Intersection :: ('i \Rightarrow ('a,'b) mrel) \Rightarrow 'i set \Rightarrow ('a,'b) mrel ($\langle \bigcap \bigcap - \rangle$ [80,80] 80) **where**
 $\bigcap \bigcap X|I \equiv \{ (a,B) . \exists f . B = (\bigcap i \in I . f i) \wedge (\forall i \in I . (a,f i) \in X i) \}$

declare inner-Union-def [mr-simp] inner-Intersection-def [mr-simp]

lemma iU-empty:
 $\bigcup \bigcup X|\{\} = 1_{\cup \cup}$
by (auto simp: mr-simp)

lemma iI-empty:
 $\bigcap \bigcap X|\{\} = 1_{\cap \cap}$
by (auto simp: mr-simp)

lemma ic-antidist-iU:
 $\sim \bigcup \bigcup X|I = \bigcap \bigcap (\text{inner-complement } o X)|I$
apply (rule antisym)
apply (clar simp simp: mr-simp)
apply (metis (mono-tags, lifting) Compl-UN double-compl)
by (clar simp simp: mr-simp) blast

lemma ic-antidist-iI:
 $\sim \bigcap \bigcap X|I = \bigcup \bigcup (\text{inner-complement } o X)|I$
apply (rule antisym)
apply (clar simp simp: mr-simp)
apply (metis Compl-INT double-complement)
by (clar simp simp: mr-simp) blast

lemma iu-right-dist-oU:
 $\bigcup X \cup \bigcup T = (\bigcup R \in X . R \cup \bigcup T)$
by (clar simp simp: mr-simp) blast

lemma ii-right-dist-oU:
 $\bigcup X \cap \bigcup T = (\bigcup R \in X . R \cap \bigcup T)$
by (clar simp simp: mr-simp) blast

lemma iu-right-subdist-iI:
 $\bigcap \bigcap X|I \cup \bigcup T \subseteq \bigcap \bigcap (\lambda i . X i \cup \bigcup T)|I$

apply (*clarsimp simp: mr-simp*)
by (*metis INT-simps(6)*)

emma *ii-right-subdist-iU*:
 $\bigcup\bigcup X|I \cap T \subseteq \bigcup\bigcup (\lambda i . X i \cap T)|I$
by (*clarsimp simp: mr-simp, metis UN-extend-simps(4)*)

emma *ic-dist-oU*:
 $\sim\bigcup X = \bigcup (\text{inner-complement} ` X)$
by (*auto simp: mr-simp*)

emma *ic-dist-oI*:
 $\sim\bigcap X = \bigcap (\text{inner-complement} ` X)$
by (*auto simp: mr-simp*)

emma *sp-left-subdist-iU*:
 $R * (\bigcup\bigcup X|I) \subseteq \bigcup\bigcup (\lambda i . R * X i)|I$
apply (*clarsimp simp: mr-simp*)
subgoal for a B f proof –
assume 1: $(a, B) \in R$
assume $\forall b \in B . \exists g . f b = \bigcup(g ` I) \wedge (\forall i \in I . (b, g i) \in X i)$
from this obtain g **where** 2: $\forall b \in B . f b = \bigcup(g b ` I) \wedge (\forall i \in I . (b, g b i) \in X i)$
by *metis*
hence 3: $\bigcup(f ` B) = (\bigcup b \in B . \bigcup(g b ` I))$
by (*meson SUP-cong*)
let $?h = \lambda i . \bigcup b \in B . g b i$
have $\bigcup(f ` B) = \bigcup(?h ` I) \wedge (\forall i \in I . \exists B . (a, B) \in R \wedge (\exists f . (\forall b \in B . (b, f b) \in X i) \wedge ?h i = \bigcup(f ` B)))$
using 1 2 3 **by** (*metis SUP-commute*)
thus *?thesis*
by *auto*
qed
done

emma *sp-right-subdist-iU*:
 $(\bigcup\bigcup X|I) * R \subseteq \bigcup\bigcup (\lambda i . X i * R)|I$
by (*clarsimp simp: mr-simp, blast*)

emma *sp-right-dist-iU*:
assumes $\forall J::'a set . J \neq \{\} \longrightarrow (\bigcup\bigcup (\lambda j . R)|J) \subseteq R$
shows $(\bigcup\bigcup X|I) * R = \bigcup\bigcup (\lambda i . X i * R)|(I::'a set)$
apply (*rule antisym*)
using *sp-right-subdist-iU* **apply** *blast*
apply (*clarsimp simp: mr-simp*)
subgoal for a f proof –
assume $\forall i \in I . \exists B . (a, B) \in X i \wedge (\exists g . (\forall b \in B . (b, g b) \in R) \wedge f i = \bigcup(g b))$
from this obtain $B g$ **where** 1: $\forall i \in I . (a, B i) \in X i \wedge (\forall b \in B i . (b, g b) \in R)$

```

 $R) \wedge f i = \bigcup(g i ` B i)$ 
  by metis
let ?B =  $\bigcup(B ` I)$ 
let ?g =  $\lambda b . \bigcup\{g i b \mid i \in I \wedge b \in B i\}$ 
have  $\forall b \in ?B . (b, ?g b) \in R$ 
proof (rule ballI)
  fix b
  let ?I = { i | i . i  $\in I \wedge b \in B i$  }
  assume ? :  $b \in \bigcup(B ` I)$ 
  have  $(b, ?g b) \in \bigcup\bigcup(\lambda j . R) | ?I$ 
    apply (clar simp simp: mr-simp)
    apply (rule exI[of - λi . g i b])
    using 1 by blast
  thus  $(b, ?g b) \in R$ 
    using 2 by (smt (verit) assms UN-E empty-Collect-eq subset-iff)
  qed
  hence ?B =  $\bigcup(B ` I) \wedge (\forall i \in I . (a, B i) \in X i) \wedge (\forall b \in ?B . (b, ?g b) \in R) \wedge$ 
 $\bigcup(f ` I) = \bigcup(?g ` ?B)$ 
  using 1 by auto
  thus  $\exists B . (\exists f . B = \bigcup(f ` I) \wedge (\forall i \in I . (a, f i) \in X i)) \wedge (\exists g . (\forall b \in B . (b, g$ 
 $b) \in R) \wedge \bigcup(f ` I) = \bigcup(g ` B))$ 
    by (metis (no-types, lifting))
  qed
  done

```

5.2 Dual

```

abbreviation dual :: ('a, 'b) mrel  $\Rightarrow$  ('a, 'b) mrel ( $\leftarrow^d$  [100] 100)
  where  $R^d \equiv \sim -R$ 

```

```

lemma dual:
 $R^d = \{ (a, B) . (a, -B) \notin R \}$ 
  by (simp add: inner-complement-def)

```

```

declare dual [mr-simp]

```

```

lemma dual-antitone:
 $R \subseteq S \implies S^d \subseteq R^d$ 
  by (simp add: ic-isotone)

```

```

lemma ic-oc-dual:
 $\sim R = -R^d$ 
  by (simp add: ic-dist-oc)

```

```

lemma dual-involutive [simp]:
 $R^{dd} = R$ 
  by (simp add: ic-dist-oc)

```

```

lemma dual-antidist-ou:

```

$(R \cup S)^d = R^d \cap S^d$
by (*simp add: ic-dist-oi*)

lemma *dual-antidist-oi*:

$(R \cap S)^d = R^d \cup S^d$
by (*simp add: ic-dist-ou*)

lemma *dual-dist-oc*:

$(\sim R)^d = \sim R^d$
by (*fact ic-dist-oc*)

lemma *dual-dist-ic*:

$(\sim R)^d = \sim R^d$
by (*simp add: ic-dist-oc*)

lemma *dual-antidist-oU*:

$(\bigcup X)^d = \bigcap (\text{dual} ` X)$
by (*simp add: ic-dist-oI uminus-Sup*)

lemma *dual-antidist-oI*:

$(\bigcap X)^d = \bigcup (\text{dual} ` X)$
by (*simp add: ic-dist-oU uminus-Inf*)

5.3 Co-composition

definition *co-prod* :: $('a,'b) mrel \Rightarrow ('b,'c) mrel \Rightarrow ('a,'c) mrel$ (**infixl** \odot 70)
where

$R \odot S \equiv \{ (a,C) . \exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f b) \in S) \wedge C = \bigcap \{ f | b . b \in B \}) \}$

lemma *co-prod-im*:

$R \odot S = \{ (a,C) . \exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f b) \in S) \wedge C = \bigcap ((\lambda x . f x) ` B)) \}$
by (*auto simp: co-prod-def*)

lemma *co-prod-iff*:

$(a,C) \in (R \odot S) \longleftrightarrow (\exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f b) \in S) \wedge C = \bigcap \{ f | b . b \in B \}))$
by (*unfold co-prod-im, auto*)

declare *co-prod-im* [*mr-simp*]

lemma *co-prod*:

$R \odot S = \sim(R * \sim S)$
apply (*clarify simp: mr-simp*)
by (*smt (verit) Collect-cong Compl-INT Compl-UN case-prodI2 double-complement old.prod.case*)

lemma *cp-left-isotone*:

$R \subseteq S \implies R \odot T \subseteq S \odot T$
by (*simp add: co-prod ic-isotone s-prod-isol*)

lemma *cp-right-isotone*:

$R \subseteq S \implies T \odot R \subseteq T \odot S$
by (*smt (verit) co-prod-iff in-mono subrelI*)

lemma *cp-isotone*:

$R \subseteq S \implies P \subseteq Q \implies R \odot P \subseteq S \odot Q$
by (*meson cp-left-isotone cp-right-isotone order-trans*)

lemma *ic-dist-cp*:

$\sim(R \odot S) = R * \sim S$
by (*simp add: co-prod*)

lemma *ic-dist-sp*:

$\sim(R * S) = R \odot \sim S$
by (*simp add: co-prod*)

lemma *ic-cp-ic-unit*:

$\sim R = R \odot \sim 1$
by (*simp add: co-prod*)

lemma *cp-left-zero [simp]*:

$\{\} \odot R = \{\}$
by (*simp add: co-prod-im*)

lemma *cp-left-unit [simp]*:

$1 \odot R = R$
by (*simp add: co-prod*)

lemma *cp-ic-unit [simp]*:

$\sim 1 \odot \sim 1 = 1$
using *ic-cp-ic-unit ic-involutive* **by** *blast*

lemma *cp-right-dist-ou*:

$(R \cup S) \odot T = (R \odot T) \cup (S \odot T)$
by (*simp add: co-prod ic-dist-ou s-prod-distr*)

lemma *cp-left-iu-unit [simp]*:

$1_{\cup\cup} \odot R = 1_{\cap\cap}$
by (*simp add: co-prod*)

lemma *cp-right-ii-unit*:

$R \odot 1_{\cap\cap} \subseteq R \cup\cup \sim R$
apply (*clar simp simp: mr-simp*)
by (*metis double-compl sup-compl-top*)

lemma *sp-right-iu-unit*:

$R * 1_{\cup\cup} \subseteq R \cap\cap \sim R$
apply (*clarsimp simp: mr-simp*)
by (*metis Compl-disjoint double-complement*)

lemma *cp-left-subdist-ii*:

$R \odot (S \cap\cap T) \subseteq (R \odot S) \cap\cap (R \odot T)$
by (*metis cl3 co-prod ic-antidist-ii ic-antidist-iu ic-isotone*)

lemma *cp-right-subantidist-iu*:

$(R \cup\cup S) \odot T \subseteq (R \odot T) \cap\cap (S \odot T)$
by (*metis co-prod ic-antidist-iu ic-isotone seq-conc-subdistr*)

lemma *cp-right-antidist-iu*:

assumes $T \cap\cap T \subseteq T$
shows $(R \cup\cup S) \odot T = (R \odot T) \cap\cap (S \odot T)$
by (*smt (verit) assms cl4 co-prod cp-right-subantidist-iu ic-antidist-ii ic-involutive ic-isotone subset-antisym*)

lemma *cp-right-dist-oU*:

$\bigcup X \odot T = (\bigcup_{R \in X} R \odot T)$
by (*auto simp: mr-simp*)

lemma *cp-left-subdist-iI*:

$R \odot (\bigcap\bigcap X|I) \subseteq \bigcap\bigcap (\lambda i . R \odot X i)|I$

proof –

have $R \odot (\bigcap\bigcap X|I) = \sim(R * (\bigcup\bigcup (\text{inner-complement } o X)|I))$
by (*simp add: co-prod ic-antidist-iI*)
also have ... $\subseteq \sim(\bigcup\bigcup (\lambda i . R * \sim(X i))|I)$
apply (*rule ic-isotone*)
using *sp-left-subdist-iU* **by** *force*
also have ... $= \bigcap\bigcap (\lambda i . R \odot X i)|I$
apply (*subst ic-antidist-iU*)
by (*metis co-prod comp-apply*)
finally show ?thesis

.

qed

lemma *cp-right-subantidist-iU*:

$(\bigcup\bigcup X|I) \odot R \subseteq \bigcap\bigcap (\lambda i . X i \odot R)|I$

proof –

have $(\bigcup\bigcup X|I) \odot R = \sim((\bigcup\bigcup X|I) * \sim R)$
by (*simp add: co-prod*)
also have ... $\subseteq \sim((\bigcup\bigcup (\lambda i . X i * \sim R)|I))$
by (*simp add: ic-isotone sp-right-subdist-iU*)
also have ... $= \bigcap\bigcap (\lambda i . X i \odot R)|I$
apply (*subst ic-antidist-iU*)
by (*metis co-prod comp-apply*)
finally show ?thesis

.

qed

```

lemma cp-right-antidist-iU:
  assumes  $\forall J::'a\ set\ .\ J \neq \{\} \longrightarrow (\bigcap\bigcap(\lambda j\ .\ R)|J) \subseteq R$ 
  shows  $(\bigcup\bigcup X|I) \odot R = \bigcap\bigcap(\lambda i\ .\ X\ i \odot R)|(I::'a\ set)$ 
proof -
  have 1:  $\bigwedge J\ .\ \bigcup\bigcup(\lambda j\ .\ \sim R)|J = \sim\bigcap\bigcap(\lambda j\ .\ R)|J$ 
  apply (subst ic-antidist-iI)
  by (metis comp-apply)
  have  $(\bigcup\bigcup X|I) \odot R = \sim((\bigcup\bigcup X|I) * \sim R)$ 
  by (simp add: co-prod)
  also have ... =  $\sim((\bigcup\bigcup(\lambda i\ .\ X\ i * \sim R)|I))$ 
  by (simp add: 1 assms sp-right-dist-iU ic-isotone)
  also have ... =  $\bigcap\bigcap(\lambda i\ .\ X\ i \odot R)|I$ 
  apply (subst ic-antidist-iU)
  by (metis co-prod comp-apply)
  finally show ?thesis
.
```

qed

5.4 Inner order

definition inner-order-iu :: $'a \times 'b\ set \Rightarrow 'a \times 'b\ set \Rightarrow \text{bool}$ (**infix** $\trianglelefteq_{\cup\cup}$ 50)
where

$$x \trianglelefteq_{\cup\cup} y \equiv \text{fst } x = \text{fst } y \wedge \text{snd } x \subseteq \text{snd } y$$

definition inner-order-ii :: $'a \times 'b\ set \Rightarrow 'a \times 'b\ set \Rightarrow \text{bool}$ (**infix** $\trianglelefteq_{\cap\cap}$ 50)
where

$$x \trianglelefteq_{\cap\cap} y \equiv \text{fst } x = \text{fst } y \wedge \text{snd } x \supseteq \text{snd } y$$

lemma inner-order-dual:

$$x \trianglelefteq_{\cup\cup} y \longleftrightarrow y \trianglelefteq_{\cap\cap} x$$

by (metis inner-order-ii-def inner-order-iu-def)

interpretation inner-order-iu: *order* ($\trianglelefteq_{\cup\cup}$) $\lambda x\ y\ .\ x \trianglelefteq_{\cup\cup} y \wedge x \neq y$
by (unfold-locales, auto simp add: inner-order-iu-def)

5.5 Up-closure, down-closure and convex-closure

abbreviation up :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (\leftrightarrow [100] 100)
where $R\uparrow \equiv R \cup\cup U$

abbreviation down :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (\leftrightarrow [100] 100)
where $R\downarrow \equiv R \cap\cap U$

abbreviation convex :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (\leftrightarrow [100] 100)
where $R\updownarrow \equiv R\uparrow \cap R\downarrow$

lemma up:

$$R\uparrow = \{ (a,B) . \exists C . (a,C) \in R \wedge C \subseteq B \}$$

```

by (simp add: p-id-U)

lemma down:
 $R\downarrow = \{ (a,B) . \exists C . (a,C) \in R \wedge B \subseteq C \}$ 
by (auto simp: mr-simp)

lemma convex:
 $R\uparrow = \{ (a,B) . \exists C D . (a,C) \in R \wedge (a,D) \in R \wedge C \subseteq B \wedge D \subseteq B \}$ 
by (auto simp: mr-simp)

declare up [mr-simp] down [mr-simp] convex [mr-simp]

lemma ic-up:
 $\sim(R\uparrow) = (\sim R)\downarrow$ 
by (simp add: ic-antidist-iu)

lemma ic-down:
 $\sim(R\downarrow) = (\sim R)\uparrow$ 
by (simp add: ic-antidist-ii)

lemma ic-convex:
 $\sim(R\uparrow) = (\sim R)\uparrow$ 
by (simp add: ic-dist-oi ic-down ic-up inf-commute)

lemma up-isotone:
 $R \subseteq S \implies R\uparrow \subseteq S\uparrow$ 
by (fact iu-left-isotone)

lemma up-increasing:
 $R \subseteq R\uparrow$ 
by (simp add: upclosed-ext)

lemma up-idempotent [simp]:
 $R\uparrow\uparrow = R\uparrow$ 
by (simp add: iu-assoc)

lemma up-dist-ou:
 $(R \cup S)\uparrow = R\uparrow \cup S\uparrow$ 
by (simp add: iu-right-dist-ou)

lemma up-dist-iu:
 $(R \cup\cup S)\uparrow = R\uparrow \cup\cup S\uparrow$ 
using cv-hom-par p-prod-assoc by blast

lemma up-dist-ii:
 $(R \cap S)\uparrow = R\uparrow \cap S\uparrow$ 
proof (rule antisym)
show  $(R \cap S)\uparrow \subseteq R\uparrow \cap S\uparrow$ 
by (simp add: iu-right-subdist-ii)

```

```

next
have  $\bigwedge a B C D E . (a,B) \in R \implies (a,C) \in S \implies \exists F . (\exists G . (B \cup D) \cap (C \cup E) = F \cup G) \wedge (\exists H I . F = H \cap I \wedge (a,H) \in R \wedge (a,I) \in S)$ 
proof -
  fix a B C D E
  assume 1:  $(a,B) \in R$ 
  assume 2:  $(a,C) \in S$ 
  let ?F = B ∩ C
  let ?G = (B ∩ E) ∪ (D ∩ C) ∪ (D ∩ E)
  have (B ∪ D) ∩ (C ∪ E) = ?F ∪ ?G
    by auto
  thus  $\exists F . (\exists G . (B \cup D) \cap (C \cup E) = F \cup G) \wedge (\exists H I . F = H \cap I \wedge (a,H) \in R \wedge (a,I) \in S)$ 
    using 1 2 by auto
  qed
  thus  $R \uparrow \cap \cap S \uparrow \subseteq (R \cap \cap S) \uparrow$ 
    by (clar simp simp: mr-simp)
qed

lemma down-isotone:
R ⊆ S implies  $R \downarrow \subseteq S \downarrow$ 
by (fact ii-left-isotone)

lemma down-increasing:
R ⊆ R↓
by (metis ic-involutive ic-isotone ic-up up-increasing)

lemma down-idempotent [simp]:
R↓↓ = R↓
by (simp add: ic-down ic-injective)

lemma down-dist-ou:
(R ∪ S)↓ = R↓ ∪ S↓
by (fact ii-right-dist-ou)

lemma down-dist-iu:
(R ∪\cup S)↓ = R↓ ∪\cup S↓
by (simp add: ic-antidist-ii ic-antidist-iu ic-injective up-dist-ii)

lemma down-dist-ii:
(R ∩\cap S)↓ = R↓ ∩\cap S↓
by (metis down-idempotent ii-assoc ii-commute)

lemma convex-isotone:
R ⊆ S implies  $R \uparrow \subseteq S \uparrow$ 
by (meson Int-mono down-isotone up-isotone)

lemma convex-increasing:
R ⊆ R↑

```

by (*simp add: down-increasing up-increasing*)

lemma *convex-idempotent* [*simp*]:

$$R^{\uparrow\downarrow} = R^{\uparrow}$$

by (*smt (verit, ccfv-threshold) U-par-idem convex-increasing convex-isotone ic-top ic-up ii-assoc iu-assoc le-inf-iff subsetI subset-antisym*)

lemma *down-sp*:

$$R_{\downarrow} = R * (1_{\cup\cup} \cup 1)$$

proof –

have $\forall a B . (\exists C . (\exists D . B = C \cap D) \wedge (a, C) \in R) \longleftrightarrow (\exists C . (a, C) \in R \wedge (\exists f . (\forall c \in C . f c = \{\}) \vee f c = \{c\}) \wedge B = (\bigcup c \in C . f c))$

proof (*intro allI, rule iffI*)

fix $a B$

assume $\exists C . (\exists D . B = C \cap D) \wedge (a, C) \in R$

from this obtain C **where** 1: $\exists D . B = C \cap D$ **and** 2: $(a, C) \in R$

by auto

let $?f = \lambda c . \text{if } c \in B \text{ then } \{c\} \text{ else } \{\}$

have $(\bigcup c \in C . ?f c) = (\bigcup c \in B . ?f c) \cup (\bigcup c \in C \cap B . ?f c)$

using 1 by blast

hence 3: $B = (\bigcup c \in C . ?f c)$

by auto

have $\forall c \in C . ?f c = \{\} \vee ?f c = \{c\}$

by auto

thus $\exists C . (a, C) \in R \wedge (\exists f . (\forall c \in C . f c = \{\}) \vee f c = \{c\}) \wedge B = (\bigcup c \in C . f c))$

using 2 3 by smt

next

fix $a B$

assume $\exists C . (a, C) \in R \wedge (\exists f . (\forall c \in C . f c = \{\}) \vee f c = \{c\}) \wedge B = (\bigcup c \in C . f c))$

from this obtain C **where** 4: $(a, C) \in R$ **and** $\exists f . (\forall c \in C . f c = \{\}) \vee f c = \{c\} \wedge B = (\bigcup c \in C . f c)$

by auto

hence $B \subseteq C$

by auto

thus $\exists C . (\exists D . B = C \cap D) \wedge (a, C) \in R$

using 4 by auto

qed

thus *?thesis*

by (*clarsimp simp: mr-simp*)

qed

lemma *up-cp*:

$$R^{\uparrow} = \sim R \odot (1_{\cap\cap} \cup \sim 1)$$

by (*metis co-prod down-sp ic-dist-ou ic-ii-unit ic-involutive ic-up*)

lemma *down-dist-sp*:

$$(R * S)_{\downarrow} = R * S_{\downarrow}$$

```

proof (rule antisym)
  show  $(R * S) \downarrow \subseteq R * S \downarrow$ 
    by (simp add: down-sp s-prod-assoc1)
  next
    have  $\bigwedge a B f . (a, B) \in R \implies \forall b \in B . \exists C . (\exists D . f b = C \cap D) \wedge (b, C) \in S$ 
     $\implies \exists E . (\exists F . (\bigcup b \in B . f b) = E \cap F) \wedge (\exists B . (a, B) \in R \wedge (\exists g . (\forall b \in B . (b, g) \in S) \wedge E = (\bigcup b \in B . g b)))$ 
    proof -
      fix  $a B f$ 
      assume  $1: (a, B) \in R$ 
      assume  $\forall b \in B . \exists C . (\exists D . f b = C \cap D) \wedge (b, C) \in S$ 
      hence  $\exists g . \forall b \in B . (\exists D . f b = g b \cap D) \wedge (b, g b) \in S$ 
        by metis
      from this obtain  $g$  where  $2: \forall b \in B . (\exists D . f b = g b \cap D) \wedge (b, g b) \in S$ 
        by auto
      hence  $(\bigcup b \in B . f b) \subseteq (\bigcup b \in B . g b)$ 
        by blast
      thus  $\exists E . (\exists F . (\bigcup b \in B . f b) = E \cap F) \wedge (\exists B . (a, B) \in R \wedge (\exists g . (\forall b \in B . (b, g b) \in S) \wedge E = (\bigcup b \in B . g b)))$ 
        using  $1\ 2$  by (metis semilattice-inf-class.inf.absorb-iff2)
    qed
    thus  $R * S \downarrow \subseteq (R * S) \downarrow$ 
      by (clarsimp simp: mr-simp)
  qed

lemma up-dist-cp:
 $(R \odot S) \uparrow = R \odot S \uparrow$ 
by (metis co-prod down-dist-sp ic-down ic-up)

lemma iu-up-oi:
 $R \uparrow \cup \cup S \uparrow = R \uparrow \cap S \uparrow$ 
by (fact up-closed-par-is-meet)

lemma ii-down-oi:
 $R \downarrow \cap \cap S \downarrow = R \downarrow \cap S \downarrow$ 
by (metis ic-antidist-ii ic-dist-oi ic-down ic-involutive up-closed-par-is-meet)

lemma down-dist-ii-oi:
 $R \downarrow \cap S \downarrow = (R \cap S) \downarrow$ 
by (simp add: down-dist-ii ii-down-oi)

lemma up-dist-iu-oi:
 $R \uparrow \cap S \uparrow = (R \cup \cup S) \uparrow$ 
by (simp add: up-closed-par-is-meet up-dist-iu)

lemma oi-down-sub-up:
 $R \downarrow \cap S \uparrow \subseteq (R \downarrow \cap S) \uparrow$ 
by (auto simp: mr-simp)

```

lemma *oi-down-up*:
 $R \downarrow \cap S = \{\} \implies R \cap S \uparrow = \{\}$
by (*metis (no-types, lifting) cp-left-zero down-increasing ic-bot inf.orderE inf-assoc inf-bot-right oi-down-sub-up up-cp*)

lemma *oi-down-up-iff*:
 $R \downarrow \cap S = \{\} \longleftrightarrow R \cap S \uparrow = \{\}$
proof (*rule iffI*)
show $R \downarrow \cap S = \{\} \implies R \cap S \uparrow = \{\}$
by (*simp add: oi-down-up*)
next
assume 1: $R \cap S \uparrow = \{\}$
have $(\sim S) \downarrow = \sim(S \uparrow)$
by (*metis (no-types) ic-down ic-involutive*)
hence $\sim(R \downarrow \cap S) = \{\}$
using 1 **by** (*metis Int-commute ic-bot ic-dist-oi ic-down oi-down-up*)
thus $R \downarrow \cap S = \{\}$
by (*metis (no-types) ic-bot ic-involutive*)
qed

lemma *down-double-complement-up*:
 $R \downarrow \subseteq S \longleftrightarrow R \subseteq \neg((\neg S) \uparrow)$
by (*metis disjoint-eq-subset-Compl double-compl oi-down-up-iff*)

lemma *up-double-complement-down*:
 $R \uparrow \subseteq S \longleftrightarrow R \subseteq \neg((\neg S) \downarrow)$
by (*metis Compl-subset-Compl-iff double-compl down-double-complement-up*)

lemma *below-up-oi-down*:
 $R \subseteq R \uparrow \cap R \downarrow$
by (*fact convex-increasing*)

lemma *cp-pa-sim*:
 $(R \odot S) \downarrow = R \otimes S \downarrow$
by (*metis co-prod ic-involutive ic-up pa-ic pe-pa-sim*)

lemma *domain-up-down-conjugate*:
 $(R \uparrow \cap S) * 1_{\cup\cup} = (R \cap S \downarrow) * 1_{\cup\cup}$
apply (*rule set-eqI, clarsimp simp: mr-simp*)
by (*smt (verit, del-insts) Int-Un-eq(1) SUP-bot SUP-bot-conv(1) Un-Int-eq(1)*)

lemma *down-below-sp-top*:
 $R \downarrow \subseteq R * U$
apply (*clarsimp simp: mr-simp*)
by (*metis Int-Union UN-constant image-empty inf-commute*)

lemma *down-oi-up-closed*:
assumes $Q \uparrow = Q$
shows $R \downarrow \cap Q \subseteq (R \cap Q) \downarrow$

```

using assms apply (clarsimp simp: mr-simp)
by (metis (no-types, lifting) assms inf.cobounded1 ucl-iff)

lemma up-dist-oU:
  ( $\bigcup X$ )↑ =  $\bigcup (up`X)$ 
  by (simp add: iu-right-dist-oU)

lemma up-dist-iU:
  assumes I ≠ {}
  shows ( $\bigcup\bigcup X|I$ )↑ =  $\bigcup\bigcup (up o X)|I$ 
  apply (rule antisym)
  apply (clarsimp simp: mr-simp)
  apply (metis UN-simps(2) assms)
  apply (clarsimp simp: mr-simp)
  subgoal for a f
    proof -
      fix a f
      assume ∀ i∈I . ∃ B . (∃ C . f i = B ∪ C) ∧ (a,B) ∈ X i
      from this obtain g where ∀ i∈I . (∃ C . f i = g i ∪ C) ∧ (a,g i) ∈ X i
        by metis
      hence (∃ C . ∪(f ` I) = ∪(g ` I) ∪ C) ∧ (∪(g ` I) = ∪(g ` I) ∧ (∀ i∈I .
        (a,g i) ∈ X i))
        by auto
      thus ∃ B . (∃ C . ∪(f ` I) = B ∪ C) ∧ (∃ f . B = ∪(f ` I) ∧ (∀ i∈I .
        (a,f i) ∈ X i))
        by auto
      qed
      done

lemma up-dist-iI:
  ( $\bigcap\bigcap X|I$ )↑ =  $\bigcap\bigcap (up o X)|I$ 
  apply (rule antisym)
  apply (clarsimp simp: mr-simp)
  apply (smt (z3) INT-simps(10) sup-Inf sup-commute)
  apply (clarsimp simp: mr-simp)
  subgoal for a f
    proof -
      assume ∀ i∈I . ∃ B . (∃ C . f i = B ∪ C) ∧ (a,B) ∈ X i
      from this obtain g where ∀ i∈I . (∃ C . f i = g i ∪ C) ∧ (a,g i) ∈ X i
        by metis
      hence (∃ C . ∩(f ` I) = ∩(g ` I) ∪ C) ∧ (∩(g ` I) = ∩(g ` I) ∧ (∀ i∈I .
        (a,g i) ∈ X i))
        by auto
      thus ∃ B . (∃ C . ∩(f ` I) = B ∪ C) ∧ (∃ f . B = ∩(f ` I) ∧ (∀ i∈I .
        (a,f i) ∈ X i))
        by auto
      qed
      done

```

lemma *down-dist-oU*:

$(\bigcup X) \downarrow = \bigcup (\text{down}^{\downarrow} X)$
by (*simp add: ii-right-dist-oU*)

lemma *down-dist-iU*:

$(\bigcup \bigcup X|I) \downarrow = \bigcup \bigcup (\text{down}^{\downarrow} o X)|I$
apply (*rule antisym*)
apply (*clarsimp simp: mr-simp*)
apply (*metis UN-extend-simps(4)*)
apply (*clarsimp simp: mr-simp*)
subgoal for *a f*
proof –
assume $\forall i \in I . \exists B . (\exists C . f i = B \cap C) \wedge (a, B) \in X i$
from this obtain *g* **where** $\forall i \in I . (\exists C . f i = g i \cap C) \wedge (a, g i) \in X i$
by *metis*
hence $(\exists C . \bigcup (f^{\downarrow} I) = \bigcup (g^{\downarrow} I) \cap C) \wedge (\bigcup (g^{\downarrow} I) = \bigcup (g^{\downarrow} I) \wedge (\forall i \in I . (a, g i) \in X i))$
by *auto*
thus $\exists B . (\exists C . \bigcup (f^{\downarrow} I) = B \cap C) \wedge (\exists f . B = \bigcup (f^{\downarrow} I) \wedge (\forall i \in I . (a, f i) \in X i))$
by *auto*
qed
done

lemma *down-dist-iI*:

assumes $I \neq \{\}$
shows $(\bigcap \bigcap X|I) \downarrow = \bigcap \bigcap (\text{down}^{\downarrow} o X)|I$
apply (*rule antisym*)
apply (*clarsimp simp: mr-simp*)
apply (*smt (verit, del-insts) INF-const INT-absorb Int-commute assms semilattice-inf-class.inf-left-commute*)
apply (*clarsimp simp: mr-simp*)
subgoal for *a f*
proof –
assume $\forall i \in I . \exists B . (\exists C . f i = B \cap C) \wedge (a, B) \in X i$
from this obtain *g* **where** $\forall i \in I . (\exists C . f i = g i \cap C) \wedge (a, g i) \in X i$
by *metis*
hence $(\exists C . \bigcap (f^{\downarrow} I) = \bigcap (g^{\downarrow} I) \cap C) \wedge (\bigcap (g^{\downarrow} I) = \bigcap (g^{\downarrow} I) \wedge (\forall i \in I . (a, g i) \in X i))$
by *auto*
thus $\exists B . (\exists C . \bigcap (f^{\downarrow} I) = B \cap C) \wedge (\exists f . B = \bigcap (f^{\downarrow} I) \wedge (\forall i \in I . (a, f i) \in X i))$
by *auto*
qed
done

lemma *iU-up-oI*:

assumes $I \neq \{\}$

```

shows  $\bigcup\bigcup (up \circ X)|I = \bigcap (up ` X ` I)$ 
apply (rule antisym)
apply (clar simp simp: mr-simp)
apply (metis UN-absorb sup-assoc)
apply (clar simp simp: mr-simp)
by (metis UN-constant assms)

lemma iI-down-oI:
assumes I ≠ {}
shows  $\bigcap\bigcap (down \circ X)|I = \bigcap (down ` X ` I)$ 
apply (rule antisym)
apply (clar simp simp: mr-simp)
apply (metis INF-absorb Int-assoc)
apply (clar simp simp: mr-simp)
using INF-eq-const assms by auto

lemma down-dist-iI-oI:
 $\bigcap (down ` X ` I) = (\bigcap\bigcap X|I)\downarrow$ 
apply (rule antisym)
apply (clar simp simp: mr-simp)
apply (metis INF-const INF-greatest INT-absorb empty-iff
semilattice-inf-class.inf.absorb-iff2 semilattice-inf-class.le-inf-iff)
apply (clar simp simp: mr-simp)
by blast

lemma up-dist-iU-oI:
 $\bigcap (up ` X ` I) = (\bigcup\bigcup X|I)\uparrow$ 
apply (rule antisym)
apply (clar simp simp: mr-simp)
subgoal for a D proof -
assume  $\forall i \in I . \exists B . (\exists C . D = B \cup C) \wedge (a, B) \in X i$ 
from this obtain f where 1:  $\forall i \in I . (\exists C . D = f i \cup C) \wedge (a, f i) \in X i$ 
by metis
hence  $\exists C . D = \bigcup (f ` I) \cup C$ 
by auto
thus ?thesis
using 1 by auto
qed
apply (clar simp simp: mr-simp)
by blast

lemma iu-up:
 $(R \cup\cup R)\uparrow = R\uparrow$ 
using up-dist-iu-oi by auto

lemma ii-down:
 $(R \cap\cap R)\downarrow = R\downarrow$ 
using down-dist-ii-oi by blast

```

```

lemma iU-up:
  assumes  $I \neq \{\}$ 
  shows  $(\bigcup \bigcup (\lambda j . R)|I)^\uparrow = R^\uparrow$ 
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  using assms apply blast
  apply (clar simp simp: mr-simp)
  by (metis UN-constant assms)

lemma iI-down:
  assumes  $I \neq \{\}$ 
  shows  $(\bigcap \bigcap (\lambda j . R)|I)^\downarrow = R^\downarrow$ 
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  using assms apply blast
  apply (clar simp simp: mr-simp)
  by (metis INF-const assms)

lemma iu-unit-up:
   $1_{\cup\cup}^\uparrow = U$ 
  by (simp add: iu-commute)

lemma iu-unit-down:
   $1_{\cup\cup}^\downarrow = 1_{\cup\cup}$ 
  by (simp add: down-sp)

lemma iu-unit-convex:
   $1_{\cup\cup}^{\uparrow\downarrow} = 1_{\cup\cup}$ 
  by (simp add: iu-unit-down p-id-zero)

lemma ii-unit-up:
   $1_{\cap\cap}^\uparrow = 1_{\cap\cap}$ 
  by (simp add: up-cp)

lemma ii-unit-down:
   $1_{\cap\cap}^\downarrow = U$ 
  using ii-commute ii-unit by blast

lemma ii-unit-convex:
   $1_{\cap\cap}^{\uparrow\downarrow} = 1_{\cap\cap}$ 
  using down-increasing ii-unit-up by blast

lemma sp-unit-down:
   $1^\downarrow = 1 \cup 1_{\cup\cup}$ 
  by (simp add: down-sp inf-sup-aci(5))

lemma sp-unit-convex:
   $1^{\uparrow\downarrow} = 1$ 
  unfolding convex s-id-def by force

```

```

lemma top-up:
   $U^\uparrow = U$ 
  by simp

lemma top-down:
   $U^\downarrow = U$ 
  by (metis U-par-idem ic-top ic-up)

lemma top-convex:
   $U^\uparrow\downarrow = U$ 
  by (simp add: top-down)

lemma bot-up:
   $\{\}^\uparrow = \{\}$ 
  by (simp add: p-prod-comm)

lemma bot-down:
   $\{\}^\downarrow = \{\}$ 
  using oi-down-up-iff by fastforce

lemma bot-convex:
   $\{\}^{\uparrow\downarrow} = \{\}$ 
  by (simp add: bot-down)

lemma down-oi-up-convex:
   $(R^\downarrow \cap S^\uparrow)^\uparrow\downarrow = R^\downarrow \cap S^\uparrow$ 
  unfolding up down convex by blast

lemma convex-iff-down-oi-up:
   $Q = Q^{\uparrow\downarrow} \longleftrightarrow (\exists R S . Q = R^\downarrow \cap S^\uparrow)$ 
  using down-oi-up-convex by blast

lemma convex-closed-oi:
   $(\bigcap R \in X . R^{\uparrow\downarrow})^\uparrow\downarrow = (\bigcap R \in X . R^{\uparrow\downarrow})$ 
  apply (rule antisym)
  apply (clarsimp simp: mr-simp)
  apply (smt (verit, best) semilattice-inf-class.inf-commute
semilattice-inf-class.inf-left-commute sup-commute sup-left-commute)
  by (meson convex-increasing)

lemma convex-closed-oi:
   $(R^{\uparrow\downarrow} \cap S^{\uparrow\downarrow})^\uparrow\downarrow = R^{\uparrow\downarrow} \cap S^{\uparrow\downarrow}$ 
  using convex-closed-oi[of {R,S}] by simp

lemma
   $(R^{\uparrow\downarrow} \cup S^{\uparrow\downarrow})^\uparrow\downarrow = R^{\uparrow\downarrow} \cup S^{\uparrow\downarrow}$ 
  nitpick[expect=genuine,card=1,3]
  oops

```

6 Powerdomain Preorders

```

abbreviation lower-less-eq :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel  $\Rightarrow$  bool (infixl  $\sqsubseteq\downarrow$  50)
where
   $R \sqsubseteq\downarrow S \equiv R \subseteq S\downarrow$ 

abbreviation upper-less-eq :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel  $\Rightarrow$  bool (infixl  $\sqsubseteq\uparrow$  50) where
   $R \sqsubseteq\uparrow S \equiv S \subseteq R\uparrow$ 

abbreviation convex-less-eq :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel  $\Rightarrow$  bool (infixl  $\sqsubseteq\uparrow\downarrow$  50) where
   $R \sqsubseteq\uparrow\downarrow S \equiv R \sqsubseteq\downarrow S \wedge R \sqsubseteq\uparrow S$ 

abbreviation Convex-less-eq :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel  $\Rightarrow$  bool (infixl  $\sqsubseteq\Downarrow\Updownarrow$  50) where
   $R \sqsubseteq\Downarrow\Updownarrow S \equiv R \subseteq S\Downarrow$ 

lemma lower-less-eq:
   $R \sqsubseteq\downarrow S \longleftrightarrow (\forall a B . (a,B) \in R \longrightarrow (\exists C . (a,C) \in S \wedge B \subseteq C))$ 
  apply (clar simp simp: mr-simp)
  apply safe
  apply blast
  by (metis inf.absorb-iff2)

lemma upper-less-eq:
   $R \sqsubseteq\uparrow S \longleftrightarrow (\forall a C . (a,C) \in S \longrightarrow (\exists B . (a,B) \in R \wedge B \subseteq C))$ 
  by (meson U-par-st subrelI subsetD)

lemma Convex-less-eq:
   $R \sqsubseteq\uparrow\downarrow S \longleftrightarrow (\forall a C . (a,C) \in R \longrightarrow (\exists B D . (a,B) \in S \wedge (a,D) \in S \wedge B \subseteq C \wedge C \subseteq D))$ 
  by (meson lower-less-eq semilattice-inf-class.le-inf-iff upper-less-eq)

lemma Convex-lower-upper:
   $R \sqsubseteq\Downarrow\Updownarrow S \longleftrightarrow R \sqsubseteq\downarrow S \wedge S \sqsubseteq\uparrow R$ 
  by auto

lemma lower-reflexive:
   $R \sqsubseteq\downarrow R$ 
  by (fact down-increasing)

lemma upper-reflexive:
   $R \sqsubseteq\uparrow R$ 
  by (fact up-increasing)

lemma convex-reflexive:
   $R \sqsubseteq\uparrow\downarrow R$ 
  by (simp add: lower-reflexive upper-reflexive)

```

lemma *Convex-reflexive*:

$R \sqsubseteq \sqcup R$

by (*fact convex-increasing*)

lemma *lower-transitive*:

$R \sqsubseteq \downarrow S \implies S \sqsubseteq \downarrow T \implies R \sqsubseteq \downarrow T$

using *down-idempotent down-isotone* **by** *blast*

lemma *upper-transitive*:

$R \sqsubseteq \uparrow S \implies S \sqsubseteq \uparrow T \implies R \sqsubseteq \uparrow T$

using *up-idempotent up-isotone* **by** *blast*

lemma *convex-transitive*:

$R \sqsubseteq \sqcup S \implies S \sqsubseteq \sqcup T \implies R \sqsubseteq \sqcup T$

by (*meson lower-transitive upper-transitive*)

lemma *Convex-transitive*:

$R \sqsubseteq \sqcup S \implies S \sqsubseteq \sqcup T \implies R \sqsubseteq \sqcup T$

by (*metis le-inf-iff lower-transitive upper-transitive*)

lemma *bot-lower-least*:

$\{\} \sqsubseteq \downarrow R$

by *simp*

lemma *top-upper-least*:

$U \sqsubseteq \uparrow R$

by (*metis U-par-idem iu-assoc le-inf-iff up-dist-iu-oi upper-reflexive*)

lemma *bot-Convex-least*:

$\{\} \sqsubseteq \sqcup R$

by *simp*

lemma *top-lower-greatest*:

$R \sqsubseteq \downarrow U$

using *U-par-idem top-down top-upper-least* **by** *blast*

lemma *bot-upper-greatest*:

$R \sqsubseteq \uparrow \{\}$

by *simp*

lemma *top-Convex-greatest*:

$R \sqsubseteq \sqcup U$

using *U-par-idem top-down top-upper-least* **by** *auto*

lemma *lower-iu-increasing*:

$R \sqsubseteq \downarrow R \cup R$

by (*meson dual-order.trans lower-reflexive subidem-par*)

lemma *upper-iu-increasing*:
 $R \sqsubseteq \sqcup R \cup S$
using *p-prod-isor top-upper-least* **by** *auto*

lemma *convex-iu-increasing*:
 $R \sqsubseteq \sqcup R \cup R$
by (*simp add: lower-iu-increasing upper-iu-increasing*)

lemma *Convex-iu-increasing*:
 $R \sqsubseteq \sqcup R \cup R$
by (*simp add: iu-up lower-iu-increasing upper-reflexive*)

lemma *lower-ii-decreasing*:
 $R \cap S \sqsubseteq \sqdownarrow R$
by (*metis ii-right-isotone top-down top-lower-greatest*)

lemma *upper-ii-decreasing*:
 $R \cap R \sqsubseteq \squparrow R$
using *convex-reflexive ii-sub-idempotent* **by** *fastforce*

lemma *convex-ii-decreasing*:
 $R \cap R \sqsubseteq \sqdownarrow R$
by (*simp add: lower-ii-decreasing upper-ii-decreasing*)

lemma *Convex-ii-increasing*:
 $R \sqsubseteq \sqcup R \cap R$
by (*simp add: ii-down lower-reflexive upper-ii-decreasing*)

lemma *iu-lower-left-isotone*:
 $R \sqsubseteq \sqdownarrow S \implies R \cup T \sqsubseteq \sqdownarrow S \cup T$
by (*simp add: down-dist-iu iu-isotone lower-reflexive*)

lemma *iu-upper-left-isotone*:
 $R \sqsubseteq \squparrow S \implies R \cup T \sqsubseteq \squparrow S \cup T$
by (*metis (no-types, lifting) iu-assoc iu-commute iu-left-isotone*)

lemma *iu-convex-left-isotone*:
 $R \sqsubseteq \sqcup S \implies R \cup T \sqsubseteq \sqcup S \cup T$
by (*simp add: iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-Convex-left-isotone*:
 $R \sqsubseteq \sqcup S \implies R \cup T \sqsubseteq \sqcup S \cup T$
by (*simp add: iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-lower-right-isotone*:
 $R \sqsubseteq \sqdownarrow S \implies T \cup R \sqsubseteq \sqdownarrow T \cup S$
by (*simp add: iu-commute iu-lower-left-isotone*)

lemma *iu-upper-right-isotone*:

$R \sqsubseteq \sqcup S \implies T \sqcup\sqcup R \sqsubseteq \sqcup T \sqcup\sqcup S$
by (*simp add: iu-assoc iu-right-isotone*)

lemma *iu-convex-right-isotone*:

$R \sqsubseteq \squparrow S \implies T \sqcup\sqcup R \sqsubseteq \squparrow T \sqcup\sqcup S$
by (*simp add: iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-Convex-right-isotone*:

$R \sqsubseteq \squparrow\sqdownarrow S \implies T \sqcup\sqcup R \sqsubseteq \squparrow\sqdownarrow T \sqcup\sqcup S$
by (*simp add: iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-lower-isotone*:

$R \sqsubseteq \sqdownarrow S \implies P \sqsubseteq \sqdownarrow Q \implies R \sqcup\sqcup P \sqsubseteq \sqdownarrow S \sqcup\sqcup Q$
by (*simp add: down-dist-iu iu-isotone*)

lemma *iu-upper-isotone*:

$R \sqsubseteq \squparrow S \implies P \sqsubseteq \squparrow Q \implies R \sqcup\sqcup P \sqsubseteq \squparrow S \sqcup\sqcup Q$
by (*simp add: iu-isotone up-dist-iu*)

lemma *iu-convex-isotone*:

$R \sqsubseteq \squparrow\sqdownarrow S \implies P \sqsubseteq \squparrow\sqdownarrow Q \implies R \sqcup\sqcup P \sqsubseteq \squparrow\sqdownarrow S \sqcup\sqcup Q$
by (*simp add: iu-lower-isotone iu-upper-isotone*)

lemma *iu-Convex-isotone*:

$R \sqsubseteq \squparrow\sqdownarrow S \implies P \sqsubseteq \squparrow\sqdownarrow Q \implies R \sqcup\sqcup P \sqsubseteq \squparrow\sqdownarrow S \sqcup\sqcup Q$
by (*simp add: down-dist-iu iu-isotone up-dist-iu*)

lemma *ii-lower-left-isotone*:

$R \sqsubseteq \sqdownarrow S \implies R \sqcap\sqcap T \sqsubseteq \sqdownarrow S \sqcap\sqcap T$
by (*simp add: down-dist-ii ii-isotone lower-reflexive*)

lemma *ii-upper-left-isotone*:

$R \sqsubseteq \squparrow S \implies R \sqcap\sqcap T \sqsubseteq \squparrow S \sqcap\sqcap T$
by (*simp add: ii-isotone up-dist-ii upper-reflexive*)

lemma *ii-convex-left-isotone*:

$R \sqsubseteq \squparrow\sqdownarrow S \implies R \sqcap\sqcap T \sqsubseteq \squparrow\sqdownarrow S \sqcap\sqcap T$
by (*simp add: ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-Convex-left-isotone*:

$R \sqsubseteq \squparrow\sqdownarrow S \implies R \sqcap\sqcap T \sqsubseteq \squparrow\sqdownarrow S \sqcap\sqcap T$
by (*simp add: ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-lower-right-isotone*:

$R \sqsubseteq \sqdownarrow S \implies T \sqcap\sqcap R \sqsubseteq \sqdownarrow T \sqcap\sqcap S$
by (*simp add: ii-assoc ii-right-isotone*)

lemma *ii-upper-right-isotone*:

$R \sqsubseteq \squparrow S \implies T \sqcap\sqcap R \sqsubseteq \squparrow T \sqcap\sqcap S$

by (*simp add: ii-commute ii-upper-left-isotone*)

lemma *ii-convex-right-isotone*:
 $R \sqsubseteq\uparrow S \implies T \cap R \sqsubseteq\uparrow T \cap S$
by (*simp add: ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-Convex-right-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies T \cap R \sqsubseteq\uparrow\downarrow T \cap S$
by (*simp add: ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-lower-isotone*:
 $R \sqsubseteq\downarrow S \implies P \sqsubseteq\downarrow Q \implies R \cap P \sqsubseteq\downarrow S \cap Q$
by (*simp add: down-dist-ii ii-isotone*)

lemma *ii-upper-isotone*:
 $R \sqsubseteq\uparrow S \implies P \sqsubseteq\uparrow Q \implies R \cap P \sqsubseteq\uparrow S \cap Q$
by (*simp add: ii-isotone up-dist-ii*)

lemma *ii-convex-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies P \sqsubseteq\uparrow\downarrow Q \implies R \cap P \sqsubseteq\uparrow\downarrow S \cap Q$
by (*simp add: ii-lower-isotone ii-upper-isotone*)

lemma *ii-Convex-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies P \sqsubseteq\uparrow\downarrow Q \implies R \cap P \sqsubseteq\uparrow\downarrow S \cap Q$
by (*simp add: ii-lower-isotone ii-upper-isotone*)

lemma *ou-lower-left-isotone*:
 $R \sqsubseteq\downarrow S \implies R \cup T \sqsubseteq\downarrow S \cup T$
by (*meson le-sup-iff lower-reflexive lower-transitive*)

lemma *ou-upper-left-isotone*:
 $R \sqsubseteq\uparrow S \implies R \cup T \sqsubseteq\uparrow S \cup T$
by (*metis Un-subset-iff sup.coboundedI1 up-dist-ou upclosed-ext*)

lemma *ou-convex-left-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies R \cup T \sqsubseteq\uparrow\downarrow S \cup T$
by (*meson ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-Convex-left-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies R \cup T \sqsubseteq\uparrow\downarrow S \cup T$
by (*meson le-inf-iff ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-lower-right-isotone*:
 $R \sqsubseteq\downarrow S \implies T \cup R \sqsubseteq\downarrow T \cup S$
by (*metis Un-commute ou-lower-left-isotone*)

lemma *ou-upper-right-isotone*:
 $R \sqsubseteq\uparrow S \implies T \cup R \sqsubseteq\uparrow T \cup S$
by (*metis Un-commute ou-upper-left-isotone*)

lemma *ou-convex-right-isotone*:
 $R \sqsubseteq\uparrow S \implies T \cup R \sqsubseteq\uparrow T \cup S$
by (*meson ou-lower-right-isotone ou-upper-right-isotone*)

lemma *ou-Convex-right-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies T \cup R \sqsubseteq\uparrow\downarrow T \cup S$
by (*metis Un-commute ou-Convex-left-isotone*)

lemma *ou-lower-isotone*:
 $R \sqsubseteq\downarrow S \implies P \sqsubseteq\downarrow Q \implies R \cup P \sqsubseteq\downarrow S \cup Q$
using *down-dist-ou* **by** *blast*

lemma *ou-upper-isotone*:
 $R \sqsubseteq\uparrow S \implies P \sqsubseteq\uparrow Q \implies R \cup P \sqsubseteq\uparrow S \cup Q$
by (*simp add: iu-right-dist-ou sup.coboundedI1 sup.coboundedI2*)

lemma *ou-convex-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies P \sqsubseteq\uparrow\downarrow Q \implies R \cup P \sqsubseteq\uparrow\downarrow S \cup Q$
by (*meson ou-lower-isotone ou-upper-isotone*)

lemma *ou-Convex-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies P \sqsubseteq\uparrow\downarrow Q \implies R \cup P \sqsubseteq\uparrow\downarrow S \cup Q$
by (*metis le-inf-iff ou-lower-isotone ou-upper-isotone*)

lemma *sp-lower-left-isotone*:
 $R \sqsubseteq\downarrow S \implies T * R \sqsubseteq\downarrow T * S$
by (*simp add: down-dist-sp s-prod-isor*)

lemma *sp-upper-left-isotone*:
 $R \sqsubseteq\uparrow S \implies T * R \sqsubseteq\uparrow T * S$
by (*meson cl3 dual-order.trans s-prod-isor upper-iu-increasing*)

lemma *sp-convex-left-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies T * R \sqsubseteq\uparrow\downarrow T * S$
by (*simp add: sp-lower-left-isotone sp-upper-left-isotone*)

lemma *sp-Convex-left-isotone*:
 $R \sqsubseteq\uparrow\downarrow S \implies T * R \sqsubseteq\uparrow\downarrow T * S$
by (*simp add: sp-lower-left-isotone sp-upper-left-isotone*)

lemma *cp-lower-left-isotone*:
 $R \sqsubseteq\downarrow S \implies T \odot R \sqsubseteq\downarrow T \odot S$
by (*smt (verit) co-prod ic-antidist-ii ic-antidist-iu ic-isotone ic-top sp-upper-left-isotone*)

lemma *cp-upper-left-isotone*:
 $R \sqsubseteq\uparrow S \implies T \odot R \sqsubseteq\uparrow T \odot S$
by (*simp add: cp-right-isotone up-dist-cp*)

lemma *cp-convex-left-isotone*:
 $R \sqsubseteq\downarrow S \implies T \odot R \sqsubseteq\downarrow T \odot S$
by (*simp add: cp-lower-left-isotone cp-upper-left-isotone*)

lemma *cp-Convex-left-isotone*:
 $R \sqsubseteq\uparrow S \implies T \odot R \sqsubseteq\uparrow T \odot S$
by (*simp add: cp-lower-left-isotone cp-upper-left-isotone*)

lemma *lower-ic-upper*:
 $R \sqsubseteq\downarrow S \longleftrightarrow \sim S \sqsubseteq\uparrow \sim R$
by (*metis ic-down ic-involutive ic-isotone*)

lemma *upper-ic-lower*:
 $R \sqsubseteq\uparrow S \longleftrightarrow \sim S \sqsubseteq\downarrow \sim R$
by (*simp add: lower-ic-upper*)

lemma *convex-ic*:
 $R \sqsubseteq\uparrow S \longleftrightarrow \sim S \sqsubseteq\uparrow \sim R$
by (*meson lower-ic-upper upper-ic-lower*)

lemma *Convex-ic*:
 $R \sqsubseteq\uparrow S \longleftrightarrow \sim R \sqsubseteq\uparrow \sim S$
by (*metis le-inf iff lower-ic-upper upper-ic-lower*)

lemma *up-lower-isotone*:
 $R \sqsubseteq\downarrow S \implies R\uparrow \sqsubseteq\downarrow S\uparrow$
by (*fact iu-lower-left-isotone*)

lemma *up-upper-isotone*:
 $R \sqsubseteq\uparrow S \implies R\uparrow \sqsubseteq\uparrow S\uparrow$
by (*fact iu-left-isotone*)

lemma *up-convex-isotone*:
 $R \sqsubseteq\uparrow S \implies R\uparrow \sqsubseteq\uparrow S\uparrow$
by (*fact iu-convex-left-isotone*)

lemma *up-Convex-isotone*:
 $R \sqsubseteq\uparrow S \implies R\uparrow \sqsubseteq\uparrow S\uparrow$
by (*fact iu-Convex-left-isotone*)

lemma *down-lower-isotone*:
 $R \sqsubseteq\downarrow S \implies R\downarrow \sqsubseteq\downarrow S\downarrow$
by (*fact down-isotone*)

lemma *down-upper-isotone*:
 $R \sqsubseteq\uparrow S \implies R\downarrow \sqsubseteq\uparrow S\downarrow$
by (*fact ii-upper-left-isotone*)

lemma *down-convex-isotone*:
 $R \sqsubseteq\downarrow S \implies R\downarrow \sqsubseteq\downarrow S\downarrow$
by (*fact ii-convex-left-isotone*)

lemma *down-Convex-isotone*:
 $R \sqsubseteq\Downarrow S \implies R\downarrow \sqsubseteq\Downarrow S\downarrow$
by (*fact ii-Convex-left-isotone*)

lemma *convex-lower-isotone*:
 $R \sqsubseteq\downarrow S \implies R\uparrow \sqsubseteq\downarrow S\uparrow$
by (*metis convex-idempotent convex-increasing le-inf-iff lower-transitive*)

lemma *convex-upper-isotone*:
 $R \sqsubseteq\uparrow S \implies R\uparrow \sqsubseteq\uparrow S\uparrow$
by (*simp add: convex-lower-isotone ic-convex upper-ic-lower*)

lemma *convex-convex-isotone*:
 $R \sqsubseteq\Downarrow S \implies R\uparrow \sqsubseteq\Downarrow S\uparrow$
by (*simp add: convex-lower-isotone convex-upper-isotone*)

lemma *convex-Convex-isotone*:
 $R \sqsubseteq\Downarrow S \implies R\uparrow \sqsubseteq\Downarrow S\uparrow$
by (*fact convex-isotone*)

lemma *subset-lower*:
 $R \subseteq S \implies R \sqsubseteq\downarrow S$
using *lower-reflexive* **by** *auto*

lemma *subset-upper*:
 $R \subseteq S \implies S \sqsubseteq\uparrow R$
using *upper-reflexive* **by** *blast*

lemma *subset-Convex*:
 $R \subseteq S \implies R \sqsubseteq\Downarrow S$
by (*simp add: subset-lower subset-upper*)

lemma *oi-subset-lower-left-isotone*:
 $R \subseteq S \implies R \cap T \sqsubseteq\downarrow S \cap T$
using *lower-reflexive* **by** *fastforce*

lemma *oi-subset-upper-left-antitone*:
 $R \subseteq S \implies S \cap T \sqsubseteq\uparrow R \cap T$
using *upper-reflexive* **by** *force*

lemma *oi-subset-Convex-left-isotone*:
 $R \subseteq S \implies R \cap T \sqsubseteq\Downarrow S \cap T$
by (*simp add: oi-subset-lower-left-isotone oi-subset-upper-left-antitone*)

lemma *oi-subset-lower-right-isotone*:

$R \subseteq S \implies T \cap R \sqsubseteq \downarrow T \cap S$
by (*simp add: oi-subset-lower-left-isotone semilattice-inf-class.inf-commute*)

lemma *oi-subset-upper-right-antitone*:

$R \subseteq S \implies T \cap S \sqsubseteq \uparrow T \cap R$
by (*simp add: oi-subset-upper-left-antitone semilattice-inf-class.inf-commute*)

lemma *oi-subset-Convex-right-isotone*:

$R \subseteq S \implies T \cap R \sqsubseteq \Downarrow T \cap S$
using *oi-subset-Convex-left-isotone* **by** *blast*

lemma *oi-subset-lower-isotone*:

$R \subseteq S \implies P \subseteq Q \implies R \cap P \sqsubseteq \downarrow S \cap Q$
by (*meson Int-mono subset-lower*)

lemma *oi-subset-upper-antitone*:

$R \subseteq S \implies P \subseteq Q \implies S \cap Q \sqsubseteq \uparrow R \cap P$
by (*meson Int-mono subset-upper*)

lemma *oi-subset-Convex-isotone*:

$R \subseteq S \implies P \subseteq Q \implies R \cap P \sqsubseteq \Downarrow S \cap Q$
by (*simp add: oi-subset-lower-isotone oi-subset-upper-antitone*)

lemma *sp-iu-unit-lower*:

$R * 1_{\cup\cup} \sqsubseteq \downarrow R$
using *lower-ii-decreasing sp-right-iu-unit* **by** *blast*

lemma *cp-ii-unit-upper*:

$R \sqsubseteq \uparrow R \odot 1_{\cap\cap}$
by (*meson cp-right-ii-unit in-mono subsetI upper-iu-increasing*)

lemma *lower-ii-down*:

$R \sqsubseteq \downarrow S \longleftrightarrow R \downarrow = (R \cap\cap S) \downarrow$
apply *safe*
apply (*metis down-dist-ii-oi inf.orderE lower-ii-decreasing lower-transitive*)
using *ii-assoc lower-ii-decreasing* **apply** *blast*
by (*metis IntE down-dist-ii-oi lower-reflexive subset-eq*)

lemma *lower-ii-lower-bound*:

$R \sqsubseteq \downarrow S \longleftrightarrow R \subseteq R \cap\cap S$
by (*clarsimp simp: mr-simp*) *blast*

lemma *upper-ii-up*:

$R \sqsubseteq \uparrow S \longleftrightarrow S \uparrow = (R \cup\cup S) \uparrow$
by (*metis inf.absorb-iff2 up-dist-iu-oi upclosed-ext upper-iu-increasing upper-transitive*)

lemma *upper-ii-upper-bound*:

$R \sqsubseteq \uparrow S \longleftrightarrow S \subseteq R \cup\cup S$

```

by (clar simp simp: mr-simp) blast

lemma
   $R \sqsubseteq \downarrow S \longleftrightarrow R = R \cap S$ 
  nitpick[expect=genuine,card=1]
  oops

lemma
   $R \sqsubseteq \uparrow S \longleftrightarrow S = R \cup S$ 
  nitpick[expect=genuine,card=1]
  oops

lemma convex-oi-Convex-iu:
   $R \uparrow \cap S \downarrow \sqsubseteq \uparrow \downarrow R \cup \cup S$ 
  by (meson inf-le1 inf-le2 iu-Convex-isotone order-trans subidem-par)

lemma convex-oi-Convex-ii:
   $R \uparrow \cap S \downarrow \sqsubseteq \uparrow \downarrow R \cap S$ 
  by (meson ii-Convex-isotone ii-sub-idempotent inf-le1 inf-le2 order-trans)

lemma convex-oi-iu-ii:
   $R \uparrow \cap S \downarrow = (R \cup S) \uparrow \cap (R \cap S) \downarrow$ 
  by (metis down-dist-ii-oi inf-assoc inf-left-commute up-dist-iu-oi)

lemma ii-lower-iu:
   $R \cap S \sqsubseteq \downarrow R \cup \cup S$ 
  apply (clar simp simp: mr-simp)
  by (metis Un-Int-eq(2) inf-left-commute)

lemma ii-upper-iu:
   $R \cap S \sqsubseteq \uparrow R \cup \cup S$ 
  by (simp add: ic-antidist-ii ic-antidist-iu ii-lower-iu upper-ic-lower)

lemma ii-convex-iu:
   $R \cap S \sqsubseteq \uparrow \downarrow R \cup \cup S$ 
  by (simp add: ii-lower-iu ii-upper-iu)

lemma convex-oi-iu-ii-convex:
   $R \uparrow \cap S \downarrow = (R \cup S) \uparrow \cap (R \cap S) \downarrow$ 
  by (metis convex-oi-iu-ii ii-lower-iu ii-upper-iu inf.commute lower-ii-down
upper-ii-up)

```

6.1 Functional properties of multirelations

```

lemma id-one-converse:
  Id = 1 ; 1
  unfolding Id-def converse-def relcomp-unfold s-id-def by force

lemma dom-explicit:

```

```

Dom R = R ; U ∩ 1
by (clar simp simp: mr-simp Dom-def) blast

lemma dom-explicit-2:
Dom R = R ; top ∩ 1
apply (clar simp simp: mr-simp Dom-def)
apply safe
apply (simp add: relcomp.relcompI top-def)
apply blast
by blast

lemma total-dom:
total R  $\longleftrightarrow$  Dom R = 1
unfolding total-def dom-explicit-2
apply (rule iffI)
using top-def apply fastforce
by (metis Int-subset-iff dom-def dom-gla-top dom-top id-one-converse inf.idem
inf-le1)

lemma total-eq:
total R  $\longleftrightarrow$  1UU = R * 1UU
by (metis total-dom U-c cd-iso dc dc-prop2)

lemma domain-pointwise:
x ∈ R * 1UU  $\longleftrightarrow$  (∃ a B . (a,B) ∈ R ∧ x = (a,{}))
by (smt mem-Collect-eq p-id-st)

card only works for finite sets

lemma univalent-2:
univalent R  $\longleftrightarrow$  (∀ a . finite { B . (a,B) ∈ R } ∧ card { B . (a,B) ∈ R } ≤
one-class.one)
proof
assume 1: univalent R
show ∀ a . finite { B . (a,B) ∈ R } ∧ card { B . (a,B) ∈ R } ≤ one-class.one
proof
fix a
let ?B = { B . (a,B) ∈ R }
show finite ?B ∧ card ?B ≤ one-class.one
proof (rule conjI)
show 2: finite ?B
proof (rule ccontr)
assume 3: infinite ?B
from this obtain B where 4: (a,B) ∈ R
using not-finite-existsD by auto
have ?B = {B}
proof
show ?B ⊆ {B}
using 1 4 by (metis (no-types, lifting) univalent-set insertCI
mem-Collect-eq subsetI)

```

```

next
  show {B} ⊆ ?B
    using 4 by simp
  qed
  thus False
    using 3 by auto
  qed
show card ?B ≤ one-class.one
proof (rule ccontr)
  assume 5: ¬ card ?B ≤ one-class.one
  from this obtain B where 6: (a,B) ∈ R
    by fastforce
  hence card (?B - {B}) ≥ one-class.one
    using 2 5 by auto
  from this obtain C where (a,C) ∈ R ∧ B ≠ C
    using 5 by (metis (no-types, lifting) CollectD One-nat-def
card.insert-remove card-Diff-singleton-if card.empty card-mono empty-iff
finite.emptyI finite.insertI insert-iff subsetI)
    thus False
    using 1 6 by (meson univalent-set)
  qed
qed
next
  assume 5: ∀ a . finite { B . (a,B) ∈ R } ∧ card { B . (a,B) ∈ R } ≤
one-class.one
  have ∀ a B C . (a,B) ∈ R ∧ (a,C) ∈ R → B = C
  proof (intro allI, rule impI)
    fix a B C
    let ?B = { B . (a,B) ∈ R }
    have 6: finite ?B
      using 5 by simp
    assume (a,B) ∈ R ∧ (a,C) ∈ R
    hence {B,C} ⊆ ?B
      by simp
    hence card {B,C} ≤ one-class.one
      using 5 6 by (meson card-mono le-trans)
    thus B = C
      by (metis One-nat-def card.empty card-insert-disjoint empty-iff finite.emptyI
finite.insertI insert-absorb lessI not-le singleton-insert-inj-eq)
    qed
    thus univalent R
      by (simp add: univalent-set)
  qed

lemma univalent-3:
  univalent R ↔ (∀ S . R * 1_UU = S * 1_UU ∧ S ⊆ R → S = R)
proof
  assume 1: ∀ S . R * 1_UU = S * 1_UU ∧ S ⊆ R → S = R

```

```

have  $\forall a B C . (a,B) \in R \wedge (a,C) \in R \longrightarrow B = C$ 
proof (intro allI, rule impI)
fix a B C
assume 2:  $(a,B) \in R \wedge (a,C) \in R$ 
show  $B = C$ 
proof (rule ccontr)
assume 3:  $B \neq C$ 
let ?S =  $R - \{ (a,C) \}$ 
have 4:  $R * 1_{\cup\cup} = ?S * 1_{\cup\cup}$ 
proof
show  $R * 1_{\cup\cup} \subseteq ?S * 1_{\cup\cup}$ 
proof
fix x::'a × 'f set
assume  $x \in R * 1_{\cup\cup}$ 
from this obtain b D where  $(b,D) \in R \wedge x = (b,\{ \})$ 
by (meson domain-pointwise)
thus  $x \in ?S * 1_{\cup\cup}$ 
using 2 3 by (metis domain-pointwise Pair-inject insertE insert-Diff)
qed
next
show  $?S * 1_{\cup\cup} \subseteq R * 1_{\cup\cup}$ 
by (simp add: s-prod-isol)
qed
have ?S ≠ R
using 2 by blast
thus False
using 1 4 by blast
qed
qed
thus univalent R
by (simp add: univalent-set)
next
assume 5: univalent R
show  $\forall S . R * 1_{\cup\cup} = S * 1_{\cup\cup} \wedge S \subseteq R \longrightarrow S = R$ 
proof
fix S
show  $R * 1_{\cup\cup} = S * 1_{\cup\cup} \wedge S \subseteq R \longrightarrow S = R$ 
proof
assume 6:  $R * 1_{\cup\cup} = S * 1_{\cup\cup} \wedge S \subseteq R$ 
have  $R \subseteq S$ 
proof
fix x
assume 7:  $x \in R$ 
from this obtain a B where 8:  $x = (a,B)$ 
by fastforce
show  $x \in S$ 
proof (cases  $\exists C . C \neq B \wedge (a,C) \in S$ )
case True
thus ?thesis

```

```

        using 5 6 7 8 by (metis subsetD univalent-set)
next
  case False
  thus ?thesis
    using 6 7 8 by (metis (no-types, lifting) domain-pointwise prod.inject)
  qed
qed
thus S = R
  using 6 by simp
qed
qed
qed

lemma total-2:
  total R  $\longleftrightarrow$  ( $\forall a . \{ B . (a,B) \in R \} \neq \{\}$ )
  by (simp add: total-set)

lemma total-3:
  total R  $\longleftrightarrow$  ( $\forall a . \text{finite } \{ B . (a,B) \in R \} \longrightarrow \text{card } \{ B . (a,B) \in R \} \geq \text{one-class.one}$ )
  by (metis finite.emptyI nonempty-set-card total-2)

lemma total-4: total R  $\longleftrightarrow$   $1_{\cup\cup} \subseteq R * 1_{\cup\cup}$ 
  by (simp add: c6 order-antisym-conv total-eq)

lemma deterministic-2:
  deterministic R  $\longleftrightarrow$  ( $\forall a . \text{card } \{ B . (a,B) \in R \} = \text{one-class.one}$ )
  apply (rule iffI)
  apply (metis One-nat-def bot-nat-0.extremum-unique deterministic-def
le-simps(2) less-Suc-eq nonempty-set-card total-2 univalent-2)
  by (metis card-1-singletonE deterministic-def finite.emptyI finite-insert
order.refl total-3 univalent-2)

lemma univalent-convex:
  assumes univalent S
  shows S = S $\uparrow$ 
  apply (rule antisym)
  apply (simp add: lower-reflexive upper-reflexive)
  apply (clarify simp: mr-simp)
  by (metis assms lattice-class.sup-inf-absorb sup-left-idem univalent-set)

lemma univalent-iu-idempotent:
  assumes univalent S
  shows S = S  $\cup\cup$  S
  apply (rule antisym)
  apply (meson convex-reflexive upper-ii-upper-bound)
  apply (clarify simp: mr-simp)
  by (metis assms sup.idem univalent-set)

```

```

lemma univalent-ii-idempotent:
  assumes univalent S
  shows S = S ∩ S
  apply (rule antisym)
  apply (simp add: ii-sub-idempotent)
  apply (clar simp simp: mr-simp)
  by (metis assms semilattice-inf-class.inf.idem univalent-set)

lemma univalent-down-iu-idempotent:
  assumes univalent S
  shows S = S↓ ∪ S
  apply (rule antisym)
  apply (meson convex-reflexive subset-upper upper-ii-upper-bound)
  apply (clar simp simp: mr-simp)
  by (metis assms lattice-class.sup-inf-absorb sup-commute univalent-set)

lemma univalent-up-ii-idempotent:
  assumes univalent S
  shows S = S↑ ∩ S
  apply (rule antisym)
  apply (metis assms ii-left-isotone univalent-ii-idempotent upclosed-ext)
  apply (clar simp simp: mr-simp)
  by (metis Int-commute assms lattice-class.inf-sup-absorb univalent-set)

lemma univalent-convex-iu-idempotent:
  assumes univalent S
  shows S = S↓ ∪ S
  by (metis assms univalent-convex univalent-iu-idempotent)

lemma univalent-convex-ii-idempotent:
  assumes univalent S
  shows S = S↓ ∩ S
  by (metis assms univalent-convex univalent-ii-idempotent)

lemma univalent-iu-closed:
  univalent R ==> univalent S ==> univalent (R ∪ S)
  by (smt (verit, best) case-prodD mem-Collect-eq p-prod-def univalent-set)

lemma univalent-ii-closed:
  univalent R ==> univalent S ==> univalent (R ∩ S)
  by (smt (verit, ccfv-SIG) CollectD Pair-inject case-prodE inner-intersection-def
       univalent-set)

lemma total-lower:
  total R <=> 1 ∪ ⊥ ∘ R
  unfolding lower-less-eq
  by (simp add: p-id-def total-set)

lemma total-upper:

```

```

total R  $\longleftrightarrow$   $R \sqsubseteq \uparrow 1 \cap \cap$ 
unfolding upper-less-eq
by (simp add: ii-unit-def total-set)

lemma total-lower-iu:
assumes total T
shows  $R \sqsubseteq \downarrow R \cup \cup T$ 
by (metis assms iu-lower-right-isotone iu-unit total-lower)

lemma total-upper-ii:
assumes total T
shows  $R \cap \cap T \sqsubseteq \uparrow R$ 
by (smt (verit, ccfv-threshold) U-par-idem assms iu-assoc iu-commute
lower-ii-lower-bound total-lower-iu up-dist-ii upper-ii-up)

lemma total-univalent-lower-iu:
assumes total T
and univalent S
and  $T \sqsubseteq \downarrow S$ 
shows  $T \cup \cup S = S$ 
proof -
have 1:  $\forall a. \exists B. (a, B) \in T$ 
by (meson assms(1) total-set)
have 2:  $\forall a B C. (a, B) \in S \wedge (a, C) \in S \longrightarrow B = C$ 
by (meson assms(2) univalent-set)
hence 3:  $T \cup \cup S \subseteq S$ 
by (metis assms(2,3) iu-left-isotone univalent-down-iu-idempotent)
hence  $S \subseteq T \cup \cup S$ 
apply (clar simp simp: mr-simp)
using 1 2 by (metis (mono-tags, lifting) CollectI Un-iff case-prodI
subset-Un-eq)
thus ?thesis
using 3 by (simp add: subset-antisym)
qed

lemma total-iu-closed:
total R  $\implies$  total S  $\implies$  total ( $R \cup \cup S$ )
by (meson lower-transitive total-lower total-lower-iu)

lemma total-ii-closed:
total R  $\implies$  total S  $\implies$  total ( $R \cap \cap S$ )
by (metis down-dist-ii-oi le-inf-iff total-lower)

lemma deterministic-lower:
assumes deterministic V
shows  $R \sqsubseteq \downarrow V \longleftrightarrow (\forall a B C. (a, B) \in R \wedge (a, C) \in V \longrightarrow B \subseteq C)$ 
proof -
have  $R \sqsubseteq \downarrow V \longleftrightarrow (\forall a B. (a, B) \in R \longrightarrow (\exists C. (a, C) \in V \wedge B \subseteq C))$ 
by (simp add: lower-less-eq)

```

```

also have ...  $\longleftrightarrow$  ( $\forall a B . (a,B) \in R \longrightarrow (\forall C . (a,C) \in V \longrightarrow B \subseteq C)$ )
  by (metis assms deterministic-set)
finally show ?thesis
  by blast
qed

lemma deterministic-upper:
assumes deterministic V
shows  $V \sqsubseteq \sqcup R \longleftrightarrow (\forall a B C . (a,B) \in R \wedge (a,C) \in V \longrightarrow C \subseteq B)$ 
proof -
have  $V \sqsubseteq \sqcup R \longleftrightarrow (\forall a C . (a,C) \in R \longrightarrow (\exists B . (a,B) \in V \wedge B \subseteq C))$ 
  by (simp add: upper-less-eq)
also have ...  $\longleftrightarrow (\forall a C . (a,C) \in R \longrightarrow (\forall B . (a,B) \in V \longrightarrow B \subseteq C))$ 
  by (metis assms deterministic-set)
finally show ?thesis
  by blast
qed

lemma deterministic-iu-closed:
deterministic R  $\Longrightarrow$  deterministic S  $\Longrightarrow$  deterministic ( $R \cup\cup S$ )
by (simp add: deterministic-def univalent-iu-closed total-iu-closed)

lemma deterministic-ii-closed:
deterministic R  $\Longrightarrow$  deterministic S  $\Longrightarrow$  deterministic ( $R \cap\cap S$ )
by (simp add: deterministic-def univalent-ii-closed total-ii-closed)

lemma total-univalent-lower-implies-upper:
assumes total T
and univalent S
and  $T \sqsubseteq \sqdownarrow S$ 
shows  $T \sqsubseteq \squparrow S$ 
by (simp add: assms total-univalent-lower-iu upper-ii-upper-bound)

lemma total-univalent-lower-implies-convex:
assumes total T
and univalent S
and  $T \sqsubseteq \sqdownarrow S$ 
shows  $T \sqsubseteq \squparrow S$ 
by (simp add: assms total-univalent-lower-implies-upper)

lemma total-univalent-upper-implies-lower:
assumes total T
and univalent S
and  $S \sqsubseteq \squparrow T$ 
shows  $S \sqsubseteq \sqdownarrow T$ 
proof (clarify simp simp: mr-simp)
fix a B
assume 1:  $(a,B) \in S$ 
from this obtain C where 2:  $(a,C) \in T$ 

```

```

by (meson assms(1) total-set)
hence  $(a,C) \in S^\uparrow$ 
using assms(3) by auto
from this obtain D where 3:  $(a,D) \in S \wedge D \subseteq C$ 
using 2 by (meson assms(3) upper-less-eq)
hence  $D = B$ 
using 1 by (meson assms(2) univalent-set)
thus  $\exists C . (\exists D . B = C \cap D) \wedge (a,C) \in T$ 
using 2 3 by (metis Int-absorb1)
qed

lemma total-univalent-upper-implies-convex:
assumes total T
and univalent S
and  $S \sqsubseteq \uparrow T$ 
shows  $S \sqsubseteq \downarrow T$ 
by (simp add: assms total-univalent-upper-implies-lower)

lemma deterministic-lower-upper:
assumes deterministic T
and deterministic S
shows  $S \sqsubseteq \downarrow T \longleftrightarrow S \sqsubseteq \uparrow T$ 
by (meson assms deterministic-def total-univalent-lower-implies-convex
total-univalent-upper-implies-lower)

lemma deterministic-lower-convex:
assumes deterministic T
and deterministic S
shows  $S \sqsubseteq \downarrow T \longleftrightarrow S \sqsubseteq \uparrow T$ 
by (simp add: assms deterministic-lower-upper)

lemma deterministic-upper-convex:
assumes deterministic T
and deterministic S
shows  $S \sqsubseteq \uparrow T \longleftrightarrow S \sqsubseteq \downarrow T$ 
by (simp add: assms deterministic-lower-upper)

lemma total-down-sp-sp-down:
assumes total T
shows  $R_\downarrow * T \subseteq R * T_\downarrow$ 
proof -
have  $R_\downarrow * T \subseteq R * ((1_{\cup\cup} \cup 1) * T)$ 
by (simp add: down-sp s-prod-assoc1)
also have ... =  $R * (1_{\cup\cup} \cup T * 1)$ 
by (simp add: s-prod-distr)
also have ... =  $R * (T * 1_{\cup\cup} \cup T * 1)$ 
by (metis assms c6 order-antisym-conv total-4)
also have ...  $\subseteq R * (T * (1_{\cup\cup} \cup 1))$ 
by (metis down-sp le-supI s-prod-isor sp-iu-unit-lower sup-ge2)

```

```

also have ... =  $R * T \downarrow$ 
  by (simp add: down-sp)
finally show ?thesis
  by simp
qed

lemma total-down-sp-semi-commute:
  total  $T \implies R \downarrow * T \subseteq (R * T) \downarrow$ 
  by (simp add: down-dist-sp total-down-sp-sp-down)

lemma total-down-dist-sp:
  total  $T \implies (R * T) \downarrow = R \downarrow * T \downarrow$ 
  by (smt (verit, best) down-dist-sp equalityI ii-assoc ii-isotone lower-reflexive
s-prod-isol top-down total-down-sp-semi-commute)

lemma univalent-ic-closed:
  univalent  $R \longleftrightarrow$  univalent ( $\sim R$ )
  apply (unfold univalent-set)
  apply (clar simp simp: mr-simp)
  by (metis double-compl)

lemma total-ic-closed:
  total  $R \longleftrightarrow$  total ( $\sim R$ )
  by (metis total-dom d-def-expl domain-up-down-conjugate equalityI ic-down
ic-top ic-up ii-commute inf.orderE lower-ic-upper top-down top-lower-greatest
total-lower total-upper-ii)

lemma deterministic-ic-closed:
  deterministic  $R \longleftrightarrow$  deterministic ( $\sim R$ )
  by (meson deterministic-def total-ic-closed univalent-ic-closed)

lemma iu-unit-deterministic:
  deterministic ( $1_{\cup\cup}$ )
  by (metis Lambda-empty det-lambda)

lemma ii-unit-deterministic:
  deterministic ( $1_{\cap\cap}$ )
  using deterministic-ic-closed iu-unit-deterministic by force

lemma univalent-upper-iu:
  assumes univalent  $R$ 
  shows  $(R \sqsubseteq \uparrow S) \longleftrightarrow (R \cup\cup S = S)$ 
proof -
  have 1:  $R \cup\cup S = S \implies R \sqsubseteq \uparrow S$ 
    using upper-iu-increasing by blast
  have 2:  $R \sqsubseteq \uparrow S \implies S \subseteq R \cup\cup S$ 
    by (simp add: upper-ii-upper-bound)
  have  $R \sqsubseteq \uparrow S \implies R \cup\cup S \subseteq S$ 
    apply (clar simp simp: mr-simp)

```

```

by (smt (verit) Ball-Collect assms case-prodD le-iff-sup subset-refl
sup.bounded-iff univalent-set)
thus ?thesis
using 1 2 by blast
qed

lemma univalent-lower-ii:
assumes univalent S
shows ( $R \sqsubseteq \downarrow S$ ) = ( $R \cap S = R$ )
apply (clar simp simp: mr-simp)
apply safe
apply (smt (z3) CollectD CollectI Collect-cong Int-iff assms case-prodD
inf-set-def subsetD univalent-set)
apply blast
by (smt (verit, ccfv-threshold) CollectD Pair-inject case-prodE inf-commute)

```

6.2 Equivalences induced by powerdomain preorders

abbreviation lower-eq :: ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow bool (**infixl** $\sqsubseteq \downarrow$ 50)

where

$$R \sqsubseteq \downarrow S \equiv R \sqsubseteq \downarrow S \wedge S \sqsubseteq \downarrow R$$

abbreviation upper-eq :: ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow bool (**infixl** $\sqsubseteq \uparrow$ 50)

where

$$R \sqsubseteq \uparrow S \equiv R \sqsubseteq \uparrow S \wedge S \sqsubseteq \uparrow R$$

abbreviation convex-eq :: ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow bool (**infixl** $\sqsubseteq \leftrightarrow$ 50)

where

$$R \sqsubseteq \leftrightarrow S \equiv R \sqsubseteq \downarrow S \wedge S \sqsubseteq \uparrow R$$

lemma Convex-eq:

$$R \sqsubseteq \uparrow S \equiv R \sqsubseteq \leftrightarrow S \wedge S \sqsubseteq \leftrightarrow R$$

by (smt (z3) semilattice-inf-class.le-inf-iff)

lemma convex-lower-upper:

$$R \sqsubseteq \downarrow S \longleftrightarrow R \sqsubseteq \downarrow S \wedge R \sqsubseteq \uparrow S$$

by auto

lemma lower-eq-down:

$$R \sqsubseteq \downarrow S \longleftrightarrow R \downarrow = S \downarrow$$

using down-idempotent down-lower-isotone lower-reflexive **by** blast

lemma upper-eq-up:

$$R \sqsubseteq \uparrow S \longleftrightarrow R \uparrow = S \uparrow$$

by (metis p-prod-comm upclosed-ext upper-ii-up)

lemma convex-eq-convex:

$$R \sqsubseteq \uparrow S \longleftrightarrow R \uparrow = S \uparrow$$

by (metis Convex-lower-upper lower-eq-down upper-eq-up)

lemma *lower-eq*:
 $R =\downarrow S \longleftrightarrow (\forall a B . (\exists C . (a,C) \in R \wedge B \subseteq C) \longleftrightarrow (\exists C . (a,C) \in S \wedge B \subseteq C))$
by (*meson lower-less-eq order-refl order-trans*)

lemma *upper-eq*:
 $R =\uparrow S \longleftrightarrow (\forall a C . (\exists B . (a,B) \in R \wedge B \subseteq C) \longleftrightarrow (\exists B . (a,B) \in S \wedge B \subseteq C))$
by (*meson order-refl order-trans upper-less-eq*)

lemma *lower-eq-reflexive*:
 $R =\downarrow R$
by (*simp add: lower-reflexive*)

lemma *upper-eq-reflexive*:
 $R =\uparrow R$
by (*simp add: upper-reflexive*)

lemma *convex-eq-reflexive*:
 $R =\uparrow\downarrow R$
by (*simp add: lower-reflexive upper-reflexive*)

lemma *lower-eq-symmetric*:
 $R =\downarrow S \implies S =\downarrow R$
by *simp*

lemma *upper-eq-symmetric*:
 $R =\uparrow S \implies S =\uparrow R$
by *simp*

lemma *convex-eq-symmetric*:
 $R =\uparrow\downarrow S \implies S =\uparrow\downarrow R$
by *simp*

lemma *lower-eq-transitive*:
 $R =\downarrow S \implies S =\downarrow T \implies R =\downarrow T$
using *lower-transitive* **by** *auto*

lemma *upper-eq-transitive*:
 $R =\uparrow S \implies S =\uparrow T \implies R =\uparrow T$
using *upper-transitive* **by** *auto*

lemma *convex-eq-transitive*:
 $R =\uparrow\downarrow S \implies S =\uparrow\downarrow T \implies R =\uparrow\downarrow T$
by (*meson lower-transitive upper-transitive*)

lemma *ou-lower-eq-left-congruence*:
 $R =\downarrow S \implies R \cup T =\downarrow S \cup T$

using *ou-lower-left-isotone* **by** *blast*

lemma *ou-upper-eq-left-congruence*:

$$R = \uparrow S \implies R \cup T = \uparrow S \cup T$$

using *ou-upper-left-isotone* **by** *blast*

lemma *ou-convex-eq-left-congruence*:

$$R = \uparrow\downarrow S \implies R \cup T = \uparrow\downarrow S \cup T$$

by (*meson ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-lower-eq-right-congruence*:

$$R = \downarrow S \implies T \cup R = \downarrow T \cup S$$

using *ou-lower-right-isotone* **by** *blast*

lemma *ou-upper-eq-right-congruence*:

$$R = \uparrow S \implies T \cup R = \uparrow T \cup S$$

using *ou-upper-right-isotone* **by** *blast*

lemma *ou-convex-eq-right-congruence*:

$$R = \uparrow\downarrow S \implies T \cup R = \uparrow\downarrow T \cup S$$

by (*meson ou-lower-right-isotone ou-upper-right-isotone*)

lemma *ou-lower-eq-congruence*:

$$R = \downarrow S \implies P = \downarrow Q \implies R \cup P = \downarrow S \cup Q$$

using *ou-lower-isotone* **by** *blast*

lemma *ou-upper-eq-congruence*:

$$R = \uparrow S \implies P = \uparrow Q \implies R \cup P = \uparrow S \cup Q$$

using *ou-upper-isotone* **by** *blast*

lemma *ou-convex-eq-congruence*:

$$R = \uparrow\downarrow S \implies P = \uparrow\downarrow Q \implies R \cup P = \uparrow\downarrow S \cup Q$$

by (*meson ou-lower-isotone ou-upper-isotone*)

lemma *iu-lower-eq-left-congruence*:

$$R = \downarrow S \implies R \cup\cup T = \downarrow S \cup\cup T$$

using *iu-lower-left-isotone* **by** *blast*

lemma *iu-upper-eq-left-congruence*:

$$R = \uparrow S \implies R \cup\cup T = \uparrow S \cup\cup T$$

using *iu-upper-left-isotone* **by** *blast*

lemma *iu-convex-eq-left-congruence*:

$$R = \uparrow\downarrow S \implies R \cup\cup T = \uparrow\downarrow S \cup\cup T$$

by (*simp add: iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-lower-eq-right-congruence*:

$$R = \downarrow S \implies T \cup\cup R = \downarrow T \cup\cup S$$

using *iu-lower-right-isotone* **by** *blast*

lemma *iu-upper-eq-right-congruence*:

$R = \uparrow S \implies T \cup\cup R = \uparrow T \cup\cup S$

using *iu-upper-right-isotone* **by** *blast*

lemma *iu-convex-eq-right-congruence*:

$R = \uparrow\downarrow S \implies T \cup\cup R = \uparrow\downarrow T \cup\cup S$

by (*simp add: iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-lower-eq-congruence*:

$R = \downarrow S \implies P = \downarrow Q \implies R \cup\cup P = \downarrow S \cup\cup Q$

using *iu-lower-isotone* **by** *blast*

lemma *iu-upper-eq-congruence*:

$R = \uparrow S \implies P = \uparrow Q \implies R \cup\cup P = \uparrow S \cup\cup Q$

using *iu-upper-isotone* **by** *blast*

lemma *iu-convex-eq-congruence*:

$R = \uparrow\downarrow S \implies P = \uparrow\downarrow Q \implies R \cup\cup P = \uparrow\downarrow S \cup\cup Q$

by (*simp add: iu-lower-isotone iu-upper-isotone*)

lemma *ii-lower-eq-left-congruence*:

$R = \downarrow S \implies R \cap\cap T = \downarrow S \cap\cap T$

using *ii-lower-left-isotone* **by** *blast*

lemma *ii-upper-eq-left-congruence*:

$R = \uparrow S \implies R \cap\cap T = \uparrow S \cap\cap T$

using *ii-upper-left-isotone* **by** *blast*

lemma *ii-convex-eq-left-congruence*:

$R = \uparrow\downarrow S \implies R \cap\cap T = \uparrow\downarrow S \cap\cap T$

by (*simp add: ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-lower-eq-right-congruence*:

$R = \downarrow S \implies T \cap\cap R = \downarrow T \cap\cap S$

using *ii-lower-right-isotone* **by** *blast*

lemma *ii-upper-eq-right-congruence*:

$R = \uparrow S \implies T \cap\cap R = \uparrow T \cap\cap S$

using *ii-upper-right-isotone* **by** *blast*

lemma *ii-convex-eq-right-congruence*:

$R = \uparrow\downarrow S \implies T \cap\cap R = \uparrow\downarrow T \cap\cap S$

by (*simp add: ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-lower-eq-congruence*:

$R = \downarrow S \implies P = \downarrow Q \implies R \cap\cap P = \downarrow S \cap\cap Q$

using *ii-lower-isotone* **by** *blast*

lemma *ii-upper-eq-congruence*:
 $R =\uparrow S \implies P =\uparrow Q \implies R \cap P =\uparrow S \cap Q$
using *ii-upper-isotone* **by** *blast*

lemma *ii-convex-eq-congruence*:
 $R =\downarrow S \implies P =\downarrow Q \implies R \cap P =\downarrow S \cap Q$
by (*simp add: ii-lower-isotone ii-upper-isotone*)

lemma *sp-lower-eq-left-congruence*:
 $R =\downarrow S \implies T * R =\downarrow T * S$
by (*simp add: sp-lower-left-isotone*)

lemma *sp-upper-eq-left-congruence*:
 $R =\uparrow S \implies T * R =\uparrow T * S$
by (*simp add: sp-upper-left-isotone*)

lemma *sp-convex-eq-left-congruence*:
 $R =\downarrow S \implies T * R =\downarrow T * S$
by (*simp add: sp-lower-left-isotone sp-upper-left-isotone*)

lemma *cp-lower-eq-left-congruence*:
 $R =\downarrow S \implies T \odot R =\downarrow T \odot S$
by (*simp add: cp-lower-left-isotone*)

lemma *cp-upper-eq-left-congruence*:
 $R =\uparrow S \implies T \odot R =\uparrow T \odot S$
by (*simp add: cp-upper-left-isotone*)

lemma *cp-convex-eq-left-congruence*:
 $R =\downarrow S \implies T \odot R =\downarrow T \odot S$
by (*simp add: cp-lower-left-isotone cp-upper-left-isotone*)

lemma *lower-eq-ic-upper*:
 $R =\downarrow S \longleftrightarrow \sim R =\uparrow \sim S$
using *lower-ic-upper* **by** *auto*

lemma *upper-eq-ic-lower*:
 $R =\uparrow S \longleftrightarrow \sim R =\downarrow \sim S$
using *upper-ic-lower* **by** *auto*

lemma *convex-eq-ic-lower*:
 $R =\downarrow S \longleftrightarrow \sim R =\downarrow \sim S$
by (*meson lower-ic-upper upper-ic-lower*)

lemma *up-lower-eq-congruence*:
 $R =\downarrow S \implies R^\uparrow =\downarrow S^\uparrow$
by (*fact iu-lower-eq-left-congruence*)

lemma *up-upper-eq-congruence*:

$R = \uparrow S \implies R \uparrow = \uparrow S \uparrow$
by (fact *iu-upper-eq-left-congruence*)

lemma *up-convex-eq-congruence*:
 $R = \uparrow\downarrow S \implies R \uparrow = \uparrow\downarrow S \uparrow$
by (fact *iu-convex-eq-left-congruence*)

lemma *down-lower-eq-congruence*:
 $R = \downarrow S \implies R \downarrow = \downarrow S \downarrow$
by (fact *ii-lower-eq-left-congruence*)

lemma *down-upper-eq-congruence*:
 $R = \uparrow S \implies R \downarrow = \uparrow S \downarrow$
by (fact *ii-upper-eq-left-congruence*)

lemma *down-convex-eq-congruence*:
 $R = \uparrow\downarrow S \implies R \downarrow = \uparrow\downarrow S \downarrow$
by (fact *ii-convex-eq-left-congruence*)

lemma *convex-lower-eq-congruence*:
 $R = \downarrow S \implies R \uparrow\downarrow = \downarrow S \uparrow\downarrow$
by (simp add: *convex-lower-isotone*)

lemma *convex-upper-eq-congruence*:
 $R = \uparrow S \implies R \uparrow\downarrow = \uparrow S \uparrow\downarrow$
by (simp add: *convex-upper-isotone*)

lemma *convex-convex-eq-congruence*:
 $R = \uparrow\downarrow S \implies R \uparrow\downarrow = \uparrow\downarrow S \uparrow\downarrow$
by (simp add: *convex-lower-isotone convex-upper-isotone*)

lemma *univalent-lower-eq-subset*:
assumes *univalent S*
and $S = \downarrow R$
shows $S \subseteq R$
proof –
have 1: $\forall a B C. (a, B) \in S \wedge (a, C) \in S \longrightarrow B = C$
using *assms(1)* **by** (simp add: *univalent-set*)
have $\forall a B. (\exists A. (a, A) \in S \wedge B \subseteq A) = (\exists A. (a, A) \in R \wedge B \subseteq A)$
by (meson *assms(2)* *lower-eq*)
hence $\forall a B. (a, B) \in S \longrightarrow (a, B) \in R$
using 1 **by** (smt (*verit*, *del-insts*) *assms(2)* *lower-less-eq subset-antisym*)
thus *?thesis*
by (simp add: *subset-iff*)
qed

lemma *univalent-lower-eq*:
assumes *univalent R*
and *univalent S*

and $R =\downarrow S$
shows $R = S$
by (meson assms subset-antisym univalent-lower-eq-subset)

lemma univalent-lower-eq-iff:
assumes univalent R
and univalent S
shows $(R =\downarrow S) \longleftrightarrow (R = S)$
using assms lower-reflexive univalent-lower-eq by auto

lemma univalent-upper-eq-subset:
assumes univalent S
and $S =\uparrow R$
shows $S \subseteq R$
proof –
have 1: $\forall a B C. (a, B) \in S \wedge (a, C) \in S \longrightarrow B = C$
using assms(1) **by** (simp add: univalent-set)
have $\forall a B. (\exists A. (a, A) \in S \wedge A \subseteq B) = (\exists A. (a, A) \in R \wedge A \subseteq B)$
by (meson assms(2) upper-eq)
hence $\forall a B. (a, B) \in S \longrightarrow (a, B) \in R$
using 1 **by** (smt (verit) order-refl subset-antisym)
thus ?thesis
by (simp add: subset-iff)
qed

lemma univalent-upper-eq:
assumes univalent R
and univalent S
and $R =\uparrow S$
shows $R = S$
by (meson assms subset-antisym univalent-upper-eq-subset)

lemma univalent-upper-eq-iff:
assumes univalent R
and univalent S
shows $(R =\uparrow S) \longleftrightarrow (R = S)$
using assms univalent-upper-eq upclosed-ext **by** blast

lemma univalent-convex-eq-iff:
assumes univalent R
and univalent S
shows $(R =\uparrow\downarrow S) \longleftrightarrow (R = S)$
by (metis assms univalent-lower-eq-iff univalent-upper-eq-iff)

lemma total-univalent-upper-ii:
assumes total T
and univalent S
and $S \sqsubseteq\uparrow T$
shows $T \cap\cap S = S$

```

apply (rule antisym)
apply (metis assms(2,3) ii-left-isotone univalent-up-ii-idempotent)
by (metis assms ii-commute lower-ii-lower-bound
total-univalent-upper-implies-lower)

lemma lower-eq-down-closed:
 $R =_{\downarrow} R_{\downarrow}$ 
by (simp add: subset-lower)

lemma upper-eq-up-closed:
 $R =_{\uparrow} R^{\uparrow}$ 
by (simp add: subset-upper)

lemma convex-eq-up-closed:
 $R =_{\uparrow\downarrow} R^{\uparrow\downarrow}$ 
by (simp add: subset-lower subset-upper)

lemma lower-join:
 $(\forall P . Q \sqsubseteq_{\downarrow} P \longleftrightarrow R \sqsubseteq_{\downarrow} P \wedge S \sqsubseteq_{\downarrow} P) \longleftrightarrow Q =_{\downarrow} R \cup S$ 
by (meson Un-subset-iff lower-reflexive lower-transitive)

lemma lower-meet:
 $(\forall P . P \sqsubseteq_{\downarrow} Q \longleftrightarrow P \sqsubseteq_{\downarrow} R \wedge P \sqsubseteq_{\downarrow} S) \longleftrightarrow Q =_{\downarrow} R \cap S$ 
by (metis (no-types, lifting) down-dist-ii-oi le-inf-iff lower-eq-down
lower-reflexive)

lemma upper-join:
 $(\forall P . Q \sqsubseteq_{\uparrow} P \longleftrightarrow R \sqsubseteq_{\uparrow} P \wedge S \sqsubseteq_{\uparrow} P) \longleftrightarrow Q =_{\uparrow} R \cup\cup S$ 
by (metis (no-types, lifting) convex-increasing le-inf-iff up-dist-iu-oi
upper-eq-up)

lemma upper-meet:
 $(\forall P . P \sqsubseteq_{\uparrow} Q \longleftrightarrow P \sqsubseteq_{\uparrow} R \wedge P \sqsubseteq_{\uparrow} S) \longleftrightarrow Q =_{\uparrow} R \cap S$ 
by (meson Un-subset-iff upper-reflexive upper-transitive)

lemma lower-ii-idempotent:
 $R \cap\cap R =_{\downarrow} R$ 
using ii-down lower-reflexive by blast

lemma upper-iu-idempotent:
 $R \cup\cup R =_{\uparrow} R$ 
using iu-up upper-reflexive by auto

lemma lower-iI-idempotent:
 $I \neq \{\} \implies (\bigcap\bigcap (\lambda j . R)|I) =_{\downarrow} R$ 
by (metis iI-down lower-eq-down)

lemma upper-iU-idempotent:
 $I \neq \{\} \implies (\bigcup\bigcup (\lambda j . R)|I) =_{\uparrow} R$ 

```

by (*metis iU-up upper-eq-up*)

lemma *down-closed-intersection-closed*:

$$R = R \downarrow \implies \forall I . I \neq \{\} \longrightarrow (\bigcap \bigcap (\lambda j . R) | I) \subseteq R$$

by (*metis lower-iI-idempotent*)

lemma *up-closed-union-closed*:

$$R = R \uparrow \implies \forall I . I \neq \{\} \longrightarrow (\bigcup \bigcup (\lambda j . R) | I) \subseteq R$$

by (*metis upper-iU-idempotent*)

lemma *ou-down-lower-eq-ou*:

$$R \downarrow \cup S \downarrow = \downarrow R \cup S$$

using *down-dist-ou lower-eq-down-closed* **by** *blast*

lemma *oi-down-lower-eq-ii*:

$$R \downarrow \cap S \downarrow = \downarrow R \cap \cap S$$

by (*simp add: down-dist-ii-oi lower-reflexive*)

lemma *ou-up-upper-eq-ou*:

$$R \uparrow \cup S \uparrow = \uparrow R \cup S$$

by (*metis ou-upper-isotone up-idempotent upper-reflexive*)

lemma *oi-up-upper-eq-iu*:

$$R \uparrow \cap S \uparrow = \uparrow R \cup \cup S$$

by (*simp add: up-dist-iu-oi upper-reflexive*)

lemma *oU-down-lower-eq-oU*:

$$(\bigcup R \in X . R \downarrow) = \downarrow \bigcup X$$

by (*metis down-dist-oU lower-eq-down-closed*)

lemma *oI-down-lower-eq-iI*:

$$(\bigcap i \in I . X i \downarrow) = \downarrow \bigcap \bigcap X | I$$

apply *safe*
using *down-dist-ii-oI apply fastforce*
by (*metis (no-types, lifting) down-dist-ii-oI image-cong image-image lower-eq-down subsetD*)

lemma *oU-up-upper-eq-oU*:

$$(\bigcup R \in X . R \uparrow) = \uparrow \bigcup X$$

by (*metis up-dist-oU upper-eq-up-closed*)

lemma *oI-up-upper-eq-iI*:

$$(\bigcap i \in I . X i \uparrow) = \uparrow \bigcup \bigcup X | I$$

by (*smt (z3) INT-extend-simps(10) Sup.SUP-cong U-par-idem p-prod-assoc p-prod-comm top-upper-least up-dist-iU-oI upper-ii-upper-bound*)

lemma *down-order-lower*:

$$R \downarrow \subseteq S \downarrow \longleftrightarrow R \sqsubseteq \downarrow S$$

by (*meson lower-eq-down-closed lower-transitive*)

lemma *up-order-upper*:

$$R\uparrow \subseteq S\uparrow \longleftrightarrow S \sqsubseteq\uparrow R$$

by (*meson upper-eq-up-closed upper-transitive*)

lemma *convex-order-lower-upper*:

$$R\downarrow \subseteq S\downarrow \longleftrightarrow R \sqsubseteq\downarrow S \wedge S \sqsubseteq\uparrow R$$

by (*meson convex-eq-up-closed le-inf-iff lower-transitive upper-transitive*)

lemma *convex-order-Convex*:

$$R\downarrow\uparrow \subseteq S\downarrow\uparrow \longleftrightarrow R \sqsubseteq\uparrow\downarrow S$$

by (*meson Convex-lower-upper convex-order-lower-upper*)

6.3 Further results for convex-closure

lemma *convex-down*:

$$R\downarrow\downarrow = R\downarrow$$

by (*meson convex-eq-up-closed lower-eq-down*)

lemma *convex-up*:

$$R\uparrow\uparrow = R\uparrow$$

by (*meson convex-eq-up-closed upper-eq-up*)

lemma *iu-dist-oi-convex*:

assumes $R = R\uparrow$

and $S = S\downarrow$

and $T = T\uparrow$

shows $(R \cap S) \cup\cup T = (R \cup\cup T) \cap (S \cup\cup T)$

nitpick[*expect=genuine,card=1*]

oops

lemma *ii-dist-oi-convex*:

assumes $R = R\downarrow$

and $S = S\downarrow$

and $T = T\uparrow$

shows $(R \cap S) \cap\cap T = (R \cap\cap T) \cap (S \cap\cap T)$

nitpick[*expect=genuine,card=1*]

oops

lemma *oI-up-closed*:

assumes $\forall R \in X . R\uparrow = R$

shows $(\bigcap X)\uparrow = \bigcap X$

by (*meson Inter-iff assms ucl-iff*)

lemma *oI-down-closed*:

assumes $\forall R \in X . R\downarrow = R$

shows $(\bigcap X)\downarrow = \bigcap X$

by (*metis (no-types, opaque-lifting) antisym assms down-isotone dual-order.refl le-Inf-iff lower-reflexive*)

```

lemma oI-convex-closed:
  assumes  $\forall R \in X . R \uparrow\downarrow = R$ 
  shows  $(\bigcap X) \uparrow\downarrow = \bigcap X$ 
  by (metis (no-types, lifting) Inf-lower Inter-greatest antisym assms
convex-increasing convex-isotone)

lemma up-dist-Union:
   $(\bigcup X) \uparrow = \bigcup \{ R \uparrow \mid R . R \in X \}$ 
  by (simp add: Setcompr-eq-image up-dist-oU)

lemma down-dist-Union:
   $(\bigcup X) \downarrow = \bigcup \{ R \downarrow \mid R . R \in X \}$ 
  by (simp add: Setcompr-eq-image down-dist-oU)

lemma convex-dist-Union:
   $(\bigcup X) \uparrow\downarrow = \bigcup \{ R \uparrow\downarrow \mid R . R \in X \}$ 
  nitpick[expect=genuine,card=1,2]
  oops

lemma up-dist-Inter:
   $(\bigcap X) \uparrow = \bigcap \{ R \uparrow \mid R . R \in X \}$ 
  nitpick[expect=genuine,card=1]
  oops

lemma down-dist-Inter:
   $(\bigcap X) \downarrow = \bigcap \{ R \downarrow \mid R . R \in X \}$ 
  nitpick[expect=genuine,card=1]
  oops

lemma convex-dist-Inter:
   $(\bigcap X) \uparrow\downarrow = \bigcap \{ R \uparrow\downarrow \mid R . R \in X \}$ 
  nitpick[expect=genuine,card=1,2]
  oops

lemma Inter-convex-closed:
   $(\bigcap X) \uparrow\downarrow = \bigcap X$ 
  nitpick[expect=genuine,card=1,2]
  oops

abbreviation convex-iu (infixl ‹UUU› 70)
  where  $R \cup\cup\downarrow S \equiv (R \cup\cup S) \uparrow\downarrow$ 

lemma convex-iu:
   $R \cup\cup\downarrow S = (R \downarrow \cup\cup S \downarrow) \cap R \uparrow \cap S \uparrow$ 
  by (metis Int-assoc Int-commute down-dist-iu up-dist-iu-oi)

lemma convex-iu-sub:
   $R \uparrow\downarrow \cup\cup S \subseteq R \cup\cup\downarrow S$ 

```

by (meson equalityE iu-Convex-left-isotone)

lemma convex-iu-convex-left:

$$R \cup\cup\downarrow S = R\downarrow \cup\cup\downarrow S$$

by (simp add: convex-down convex-iu convex-up)

lemma convex-iu-convex-right:

$$R \cup\cup\downarrow S = R \cup\cup\downarrow S\downarrow$$

by (simp add: convex-down convex-iu convex-up)

lemma convex-iu-convex:

$$R \cup\cup\downarrow S = R\downarrow \cup\cup\downarrow S\downarrow$$

by (simp add: convex-down convex-iu convex-up)

lemma convex-iu-assoc:

$$(R \cup\cup\downarrow S) \cup\cup\downarrow T = R \cup\cup\downarrow (S \cup\cup\downarrow T)$$

by (smt convex-iu-convex-left convex-iu-convex-right p-prod-assoc)

lemma convex-iu-comm:

$$R \cup\cup\downarrow S = S \cup\cup\downarrow R$$

by (simp add: p-prod-comm)

lemma convex-iu-unit:

$$R = R\downarrow \implies R \cup\cup\downarrow 1_{\cup\cup} = R$$

by simp

abbreviation convex-ii (**infixl** $\langle\cap\cap\downarrow\rangle$ 70)

where $R \cap\cap\downarrow S \equiv (R \cap\cap S)\downarrow$

lemma convex-ii:

$$R \cap\cap\downarrow S = (R\uparrow \cap\cap S\uparrow) \cap R\downarrow \cap S\downarrow$$

by (simp add: down-dist-ii-oi inf-assoc up-dist-ii)

lemma convex-ii-sub:

$$R\downarrow \cap\cap S \subseteq R \cap\cap\downarrow S$$

by (meson equalityE ii-Convex-left-isotone)

lemma convex-ii-convex-left:

$$R \cap\cap\downarrow S = R\downarrow \cap\cap\downarrow S$$

by (simp add: convex-down convex-ii convex-up)

lemma convex-ii-convex-right:

$$R \cap\cap\downarrow S = R \cap\cap\downarrow S\downarrow$$

by (simp add: convex-down convex-ii convex-up)

lemma convex-ii-convex:

$$R \cap\cap\downarrow S = R\downarrow \cap\cap\downarrow S\downarrow$$

by (simp add: convex-down convex-ii convex-up)

lemma *convex-ii-assoc*:

$$(R \cap\downarrow S) \cap\downarrow T = R \cap\downarrow (S \cap\downarrow T)$$

by (*smt convex-ii-convex-left convex-ii-convex-right ii-assoc*)

lemma *convex-ii-comm*:

$$R \cap\downarrow S = S \cap\downarrow R$$

by (*simp add: ii-commute*)

lemma *convex-ii-unit*:

$$R = R\downarrow \implies R \cap\downarrow 1_{\cap\cap} = R$$

by (*simp*)

lemma *convex-iu-ic*:

$$\sim(R \cup\downarrow S) = \sim R \cap\downarrow \sim S$$

by (*simp add: ic-antidist-iu ic-convex*)

lemma *convex-ii-ic*:

$$\sim(R \cap\downarrow S) = \sim R \cup\downarrow \sim S$$

by (*simp add: ic-antidist-ii ic-convex*)

abbreviation *convex-sup* :: $('a, 'b) mrel set \Rightarrow ('a, 'b) mrel (\langle\cup\downarrow\rangle)$ **where**

$$\cup\downarrow X \equiv (\cup X)\downarrow$$

lemma *convex-sup-convex*:

$$\cup\downarrow X = (\cup\downarrow X)\downarrow$$

by (*simp*)

lemma *convex-sup-inter*:

$$\cup\downarrow X = \bigcap \{ Y . Y = Y\downarrow \wedge \cup X \subseteq Y \}$$

apply (*rule antisym*)

apply (*smt (verit) Inf-greatest convex-isotone mem-Collect-eq*)

by (*smt (verit, del-insts) Inter-lower convex-down convex-increasing convex-up mem-Collect-eq*)

lemma *convex-iu-dist-convex-sup*:

$$\cup\downarrow X \cup\downarrow S = \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$$

proof –

have $\cup\downarrow X \cup\downarrow S = \cup X \cup\downarrow S$

using *convex-iu-convex-left* **by** *blast*

also have $\dots = \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$

by (*simp add: Setcompr-eq-image iu-right-dist-oU*)

also have $\dots = \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$

proof (*rule antisym*)

show $\cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \} \subseteq \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$

by (*smt (verit, ccfv-SIG) Convex-transitive Sup-mono dual-order.refl mem-Collect-eq subset-Convex*)

have $\forall R \in X . R \cup\downarrow S \subseteq \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$

by (*smt (verit, ccfv-SIG) Union-upper convex-isotone mem-Collect-eq*)

thus $\cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \} \subseteq \cup\downarrow \{ R \cup\downarrow S \mid R . R \in X \}$

```

apply (subst convex-order-Convex)
apply (rule Union-least)
by blast
qed
finally show ?thesis
.

qed

lemma convex-ii-dist-convex-sup:
 $\bigcup\limits_{\uparrow\downarrow} X \cap\cap\uparrow\downarrow S = \bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap\uparrow\downarrow S \mid R . R \in X \}$ 
proof -
have  $\bigcup\limits_{\uparrow\downarrow} X \cap\cap\uparrow\downarrow S = \bigcup\limits_{\uparrow\downarrow} X \cap\cap\uparrow\downarrow S$ 
using convex-ii-convex-left by blast
also have ... =  $\bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap S \mid R . R \in X \}$ 
by (simp add: Setcompr-eq-image ii-right-dist-oU)
also have ... =  $\bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap\uparrow\downarrow S \mid R . R \in X \}$ 
proof (rule antisym)
show  $\bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap S \mid R . R \in X \} \subseteq \bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap\uparrow\downarrow S \mid R . R \in X \}$ 
by (smt (verit, ccfv-SIG) Convex-transitive Sup-mono dual-order.refl
mem-Collect-eq subset-Convex)
have  $\forall R \in X . R \cap\cap\uparrow\downarrow S \subseteq \bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap S \mid R . R \in X \}$ 
by (smt (verit, ccfv-SIG) Union-upper convex-isotone mem-Collect-eq)
thus  $\bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap\uparrow\downarrow S \mid R . R \in X \} \subseteq \bigcup\limits_{\uparrow\downarrow} \{ R \cap\cap S \mid R . R \in X \}$ 
apply (subst convex-order-Convex)
apply (rule Union-least)
by blast
qed
finally show ?thesis
.

qed

lemma convex-dist-sup:
 $(\bigcup X)\uparrow\downarrow = \bigcup\limits_{\uparrow\downarrow} \{ R\uparrow\downarrow \mid R . R \in X \}$ 
proof (rule antisym)
have  $\forall R \in X . R \subseteq \bigcup\limits_{\uparrow\downarrow} \{ R\uparrow\downarrow \mid R . R \in X \}$ 
by (metis (mono-tags, lifting) Sup-upper2 convex-increasing mem-Collect-eq
subset-Convex)
thus  $(\bigcup X)\uparrow\downarrow \subseteq \bigcup\limits_{\uparrow\downarrow} \{ R\uparrow\downarrow \mid R . R \in X \}$ 
by (meson Sup-least convex-order-Convex)
have  $\forall R \in X . R\uparrow\downarrow \subseteq (\bigcup X)\uparrow\downarrow$ 
by (meson Union-upper convex-isotone)
thus  $\bigcup\limits_{\uparrow\downarrow} \{ R\uparrow\downarrow \mid R . R \in X \} \subseteq (\bigcup X)\uparrow\downarrow$ 
apply (subst convex-order-Convex)
apply (rule Union-least)
by blast
qed

```

7 Fusion and Fission

7.1 Atoms and co-atoms

```
definition atoms :: ('a,'b) mrel ( $\langle A_{\cup\cup} \rangle$ )
  where  $A_{\cup\cup} \equiv \{ (a,\{b\}) \mid a \ b . \ True \}$ 
```

```
definition co-atoms :: ('a,'b) mrel ( $\langle A_{\cap\cap} \rangle$ )
  where  $A_{\cap\cap} \equiv \{ (a,UNIV - \{b\}) \mid a \ b . \ True \}$ 
```

```
declare atoms-def [mr-simp] co-atoms-def [mr-simp]
```

```
lemma atoms-solution:
```

```
   $A_{\cup\cup}\uparrow = -1_{\cup\cup}$ 
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  apply (clar simp simp: mr-simp)
  by (metis equals0I insert-is-Un mk-disjoint-insert)
```

```
lemma atoms-least-solution:
```

```
  assumes  $R\uparrow = -1_{\cup\cup}$ 
  shows  $A_{\cup\cup} \subseteq R$ 
proof
  fix  $x :: 'a \times 'b$  set
  assume  $1: x \in A_{\cup\cup}$ 
  from this obtain  $a \ b$  where  $2: x = (a,\{b\})$ 
    by (smt CollectD atoms-def)
  have  $3: x \in R\uparrow$ 
    using 1 assms atoms-solution upper-reflexive by fastforce
  have  $(a,\{\}) \notin R\uparrow$ 
    by (metis ComplD IntE U-c U-par-idem assms domain-pointwise p-prod-assoc
up-dist-iu-oi)
  thus  $x \in R$ 
    using 2 3 by (smt (verit) U-par-st subsetD subset-singletonD upclosed-ext)
qed
```

```
lemma ic-atoms:
```

```
   $\sim A_{\cup\cup} = A_{\cap\cap}$ 
  apply (clar simp simp: mr-simp)
  by fastforce
```

```
lemma ic-co-atoms:
```

```
   $\sim A_{\cap\cap} = A_{\cup\cup}$ 
  by (metis ic-atoms ic-involutive)
```

```
lemma co-atoms-solution:
```

```
   $A_{\cap\cap}\downarrow = -1_{\cap\cap}$ 
  by (metis atoms-solution ic-atoms ic-dist-oc ic-iu-unit ic-up)
```

```
lemma co-atoms-least-solution:
```

assumes $R \downarrow = -1_{\cap\cap}$
shows $A_{\cap\cap} \subseteq R$
by (metis assms atoms-least-solution ic-atoms ic-dist-oc ic-down ic-ii-unit ic-involutive ic-isotone)

lemma *iu-unit-atoms-disjoint*:

$1_{\cup\cup} \cap A_{\cup\cup} = \{\}$
by (metis Compl-disjoint atoms-solution iu-unit-down oi-down-up-iff)

lemma *ii-unit-co-atoms-disjoint*:

$1_{\cap\cap} \cap A_{\cap\cap} = \{\}$
using co-atoms-solution lower-reflexive **by** fastforce

lemma *atoms-sp-idempotent*:

$A_{\cup\cup} * A_{\cup\cup} = A_{\cup\cup}$
by (auto simp: mr-simp)

lemma *atoms-sp-cp*:

$(R \cap A_{\cup\cup}) * S = (R \cap A_{\cup\cup}) \odot S$
by (auto simp: mr-simp)

7.2 Inner-functional properties

abbreviation *inner-univalent* :: $('a, 'b) mrel \Rightarrow bool$ **where**
 $inner-univalent R \equiv R \subseteq 1_{\cup\cup} \cup A_{\cup\cup}$

abbreviation *inner-total* :: $('a, 'b) mrel \Rightarrow bool$ **where**
 $inner-total R \equiv R \subseteq -1_{\cup\cup}$

abbreviation *inner-deterministic* :: $('a, 'b) mrel \Rightarrow bool$ **where**
 $inner-deterministic R \equiv inner-total R \wedge inner-univalent R$

lemma *inner-deterministic-atoms*:

$inner-deterministic R \longleftrightarrow R \subseteq A_{\cup\cup}$
using atoms-solution upper-reflexive **by** fastforce

lemma *inner-univalent*:

$inner-univalent R \longleftrightarrow (\forall a b c B . (a, B) \in R \wedge b \in B \wedge c \in B \longrightarrow b = c)$
apply (clarify simp: mr-simp, safe)
apply blast
by (smt (z3) UNIV-I UN-iff equals0I insertI1 insert-absorb singleton-insert-inj-eq' subsetI)

lemma *inner-univalent-2*:

$inner-univalent R \longleftrightarrow (\forall a B . (a, B) \in R \longrightarrow finite B \wedge card B \leq one-class.one)$
apply (rule iffI)
apply (metis card-eq-0-iff finite.emptyI inner-univalent is-singletonI'
is-singleton-altdef linear nonempty-set-card)

by (*metis all-not-in-conv card-1-singletonE eq-iff inner-univalent nonempty-set-card singletonD*)

lemma *inner-total*:

inner-total R $\longleftrightarrow (\forall a B . (a,B) \in R \longrightarrow (\exists b . b \in B))$

apply (*rule iffI*)

apply (*smt (verit, del-insts) Collect-empty-eq all-not-in-conv disjoint-eq-subset-Compl p-id-zero-st*)

by (*smt (verit) Collect-empty-eq disjoint-eq-subset-Compl ex-in-conv p-id-zero-st*)

lemma *inner-total-2*:

inner-total R $\longleftrightarrow (\forall a B . (a,B) \in R \longrightarrow B \neq \{\})$

by (*meson all-not-in-conv inner-total*)

lemma *inner-total-3*:

inner-total R $\longleftrightarrow (\forall a B . (a,B) \in R \wedge \text{finite } B \longrightarrow \text{card } B \geq \text{one-class.one})$

by (*metis finite.emptyI inner-total-2 nonempty-set-card*)

lemma *inner-deterministic*:

inner-deterministic R $\longleftrightarrow (\forall a B . (a,B) \in R \longrightarrow (\exists !b . b \in B))$

by (*metis (no-types, lifting) inner-total inner-univalent*)

lemma *inner-deterministic-2*:

inner-deterministic R $\longleftrightarrow (\forall a B . (a,B) \in R \longrightarrow \text{card } B = \text{one-class.one})$

by (*metis card-1-singletonE eq-iff finite.emptyI finite-insert inner-total-3 inner-univalent-2*)

lemma *inner-deterministic-sp-unit*:

inner-deterministic 1

by (*simp add: inner-deterministic s-id-def*)

lemma *inner-univalent-down*:

assumes *inner-univalent S*

shows *S $\downarrow \subseteq S \cup 1_{\cup\cup}$*

using assms by (*auto simp: mr-simp*)

lemma *inner-deterministic-lower-eq*:

assumes *inner-deterministic V*

and *inner-deterministic W*

and *V = \downarrow W*

shows *V = W*

using assms inner-univalent-down by *blast*

lemma *inner-total-down-closed*:

inner-total T $\Longrightarrow R \subseteq T \Longrightarrow \text{inner-total } R$

by *simp*

lemma *inner-univalent-down-closed*:

inner-univalent $T \implies R \subseteq T \implies \text{inner-univalent } R$
by *simp*

lemma *inner-deterministic-down-closed*:
inner-deterministic $T \implies R \subseteq T \implies \text{inner-deterministic } R$
by *blast*

lemma *inner-univalent-convex*:
assumes *inner-univalent* R
shows $R = R^{\uparrow}$
apply (*rule antisym*)
using *convex-increasing* **apply** *blast*
apply (*clarsimp simp: mr-simp*)
by (*smt (verit) Un-Int-eq(2) Un-Int-eq(3) assms boolean-algebra-cancel.sup0 disjoint-iff inner-univalent semilattice-inf-class.inf.orderE subsetI*)

lemma *inner-deterministic-alt-closure*:
inner-deterministic $R = (R \ O \ \text{converse} \ 1 \ O \ 1 = R)$
apply (*clarsimp simp: mr-simp*)
apply *safe*
apply *force*
by *blast+*

lemma *inner-deterministic-s-id-conv-epsiloff*:
inner-deterministic $R \implies R \ O \ \text{converse} \ s\text{-id} = R \ O \ \text{epsiloff}$
apply (*clarsimp simp: mr-simp*)
unfolding *epsiloff-def*
by *blast*

lemma *inner-deterministic-lower-iff*:
assumes *inner-deterministic* R
and *inner-deterministic* S
shows $(R \sqsubseteq \downarrow S) \longleftrightarrow (R \subseteq S)$
apply *standard*
apply (*smt (verit, ccfv-threshold) Un-commute assms disjoint-eq-subset-Compl inf.orderE inf.orderI inf-commute inf-sup-distrib2 inner-univalent-down sup.orderE sup-bot-left*)
by (*simp add: subset-lower*)

lemma *inner-deterministic-upper-iff*:
assumes *inner-deterministic* R
and *inner-deterministic* S
shows $(R \sqsubseteq \uparrow S) \longleftrightarrow (S \subseteq R)$
apply *standard*
apply (*clarsimp simp: mr-simp*)
using *inner-deterministic* **apply** (*smt (verit, del-insts) Ball-Collect Un-subset-iff assms case-prodD subsetD subsetI subset-antisym*)
by (*simp add: subset-upper*)

```

lemma inner-deterministic-lower-eq-iff:
  assumes inner-deterministic R
  and inner-deterministic S
  shows (R =↓ S)  $\longleftrightarrow$  (R = S)
  by (meson assms inner-deterministic-lower-eq lower-reflexive)

lemma inner-deterministic-upper-eq-iff:
  assumes inner-deterministic R
  and inner-deterministic S
  shows (R =↑ S)  $\longleftrightarrow$  (R = S)
  by (simp add: antisym assms inner-deterministic-upper-iff)

lemma inner-deterministic-convex-eq-iff:
  assumes inner-deterministic R
  and inner-deterministic S
  shows (R =↑ S)  $\longleftrightarrow$  (R = S)
  by (metis assms inner-deterministic-lower-eq-iff
inner-deterministic-upper-eq-iff)

lemma
  inner-univalent R  $\implies$  inner-univalent S  $\implies$  inner-univalent (R  $\cup\cup$  S)
  nitpick[expect=genuine,card=1,2]
  oops

lemma inner-univalent-ii-closed:
  inner-univalent R  $\implies$  inner-univalent S  $\implies$  inner-univalent (R  $\cap\cap$  S)
  by (metis (no-types, lifting) Un-subset-iff convex-reflexive down-dist-ii-oi
inner-univalent-down inner-univalent-down-closed le-inf-iff subsetI)

lemma inner-total-iu-closed:
  inner-total R  $\implies$  inner-total S  $\implies$  inner-total (R  $\cup\cup$  S)
  by (metis U-par-idem U-par-p-id atoms-solution c-prod-idr iu-upper-isotone
s-prod-p-idl top-upper-least)

lemma
  inner-total R  $\implies$  inner-total S  $\implies$  inner-total (R  $\cap\cap$  S)
  nitpick[expect=genuine,card=1,2]
  oops

```

7.3 Fusion

```

lemma fusion-set:
  fus R  $\equiv$  { (a,B) . B =  $\bigcup$  { C . (a,C)  $\in$  R } }
  unfolding fus-set Image-singleton
  by (smt (verit) Collect-cong Pair-inject case-prodE case-prodI2)

declare fusion-set [mr-simp]

lemma fusion-lower-increasing:

```

```

 $R \sqsubseteq \downarrow \text{fus } R$ 
apply (clar simp simp: mr-simp)
by blast

lemma fusion-deterministic:
  deterministic (fus R)
  by (simp add: deterministic-set fusion-set)

lemma fusion-least:
  assumes  $R \sqsubseteq \downarrow S$ 
  and deterministic S
  shows fus R  $\sqsubseteq \downarrow S$ 
  proof (clar simp simp: mr-simp)
    fix a::'a
    from assms(2) obtain C::'b set where 1:  $(a, C) \in S$ 
    by (meson deterministic-set)
    hence  $\bigcup \{ B . (a, B) \in R \} \subseteq C$ 
    using assms deterministic-lower by (smt (verit, del-insts) Sup-le-iff
    mem-Collect-eq)
    thus  $\exists C . (\exists D . \bigcup \{ B . (a, B) \in R \} = C \cap D) \wedge (a, C) \in S$ 
    using 1 by blast
  qed

lemma fusion-unique:
  assumes  $\forall R . R \sqsubseteq \downarrow f R$ 
  and  $\forall R . \text{deterministic } (f R)$ 
  and  $\forall R S . R \sqsubseteq \downarrow S \wedge \text{deterministic } S \longrightarrow f R \sqsubseteq \downarrow S$ 
  shows  $f T = \text{fus } T$ 
  apply (rule univalent-lower-eq)
  using assms(2) deterministic-def apply blast
  using deterministic-def fusion-deterministic apply blast
  by (simp add: assms fusion-deterministic fusion-least fusion-lower-increasing)

lemma fusion-down-char:
   $(\text{fus } R) \downarrow = -((-(R \downarrow) \cap A_{\cup\cup}) \uparrow)$ 
proof
  show  $(\text{fus } R) \downarrow \subseteq -((-(R \downarrow) \cap A_{\cup\cup}) \uparrow)$ 
  apply (clar simp simp: mr-simp)
  by blast
next
  show  $-((-(R \downarrow) \cap A_{\cup\cup}) \uparrow) \subseteq (\text{fus } R) \downarrow$ 
  proof (clar simp simp: mr-simp)
    fix a A
    assume 1:  $\forall B . (\forall C . A \neq B \cup C) \vee (\exists C . (\exists D . B = C \cap D) \wedge (a, C) \in R) \vee (\forall b . B \neq \{b\})$ 
    have A  $\subseteq \bigcup \{ C . (a, C) \in R \}$ 
    proof
      fix x
      assume x  $\in A$ 

```

```

from this obtain C where x ∈ C ∧ (a,C) ∈ R
  using 1 by (metis IntD1 insert-Diff insert-is-Un singletonI)
  thus x ∈ ∪{ C . (a,C) ∈ R }
    by blast
qed
thus ∃ D . A = ∪{ C . (a,C) ∈ R } ∩ D
  by auto
qed
qed

lemma fusion-up-char:
(fus R)↑ = −((~(R↓) ∩ A∩∩)↓)
proof
show (fus R)↑ ⊆ −((~(R↓) ∩ A∩∩)↓)
apply (clar simp simp: mr-simp)
by blast
next
show −((~(R↓) ∩ A∩∩)↓) ⊆ (fus R)↑
proof (clar simp simp: mr-simp)
fix a A
assume 1: ∀ C . (∀ D . A ≠ C ∩ D) ∨ (∀ B . (∀ D . − C ≠ B ∩ D) ∨ (a,B)
∉ R) ∨ (∀ b . C ≠ UNIV − {b})
have ∪{ C . (a,C) ∈ R } ⊆ A
proof
fix x
assume x ∈ ∪{ C . (a,C) ∈ R }
from this obtain C where x ∈ C ∧ (a,C) ∈ R
  by blast
thus x ∈ A
using 1 by (metis Compl-eq-Diff-UNIV Diff-Diff-Int Diff-cancel Diff-eq
Diff-insert-absorb Int-commute double-complement insert-Diff insert-inter-insert)
qed
thus ∃ C . A = ∪{ C . (a,C) ∈ R } ∪ C
  by auto
qed
qed

lemma fusion-up-char-2:
(fus R)↑ = −(((R↓ ∩ A∪∪) * ~1)↓)
by (simp add: atoms-sp-cp co-prod fusion-up-char ic-atoms ic-dist-oi)

lemma fusion-char:
fus R = −((−(R↓) ∩ A∪∪)↑) ∩ −((~(R↓) ∩ A∩∩)↓)
by (metis deterministic-def fusion-deterministic fusion-down-char
fusion-up-char inf-commute univalent-convex)

lemma fusion-char-2:
fus R = −((−(R↓) ∩ A∪∪)↑) ∩ −(((R↓ ∩ A∪∪) * ~1)↓)
using fusion-char fusion-up-char fusion-up-char-2 by blast

```

lemma *fusion-lower-isotone*:
 $R \sqsubseteq \downarrow S \implies \text{fus } R \sqsubseteq \downarrow \text{fus } S$
by (*meson fusion-deterministic fusion-least fusion-lower-increasing lower-transitive*)

lemma *fusion-iu-idempotent*:
 $\text{fus } R \cup\cup \text{fus } R = \text{fus } R$
using *deterministic-def fusion-deterministic univalent-iu-idempotent* **by** *blast*

lemma *fusion-down*:
 $\text{fus } R = \text{fus } (R \downarrow)$
by (*simp add: fusion-char*)

lemma *fusion-iu-total*:
 $\text{total } T \implies T \cup\cup \text{fus } T = \text{fus } T$
by (*meson deterministic-def fusion-deterministic fusion-lower-increasing total-univalent-lower-iu*)

lemma *fusion-deterministic-fixpoint*:
 $\text{deterministic } R \longleftrightarrow R = \text{fus } R$
by (*metis deterministic-def fusion-deterministic fusion-iu-total fusion-least lower-reflexive p-prod-comm total-univalent-lower-iu*)

abbreviation *non-empty* :: $('a, 'b) \text{ mrel} \Rightarrow ('a, 'b) \text{ mrel} (\langle ne \rightarrow [100] 100)$
where $ne R \equiv R \cap -1 \cup\cup$

lemma *non-empty*:
 $ne R = \{ (a, B) \mid a \in B . (a, B) \in R \wedge B \neq \{ \} \}$
by (*auto simp: mr-simp*)

lemma *ne-equality*:
 $ne R = R \longleftrightarrow R \subseteq -1 \cup\cup$
by *blast*

lemma *ne-dist-ou*:
 $ne (R \cup S) = ne R \cup ne S$
by (*fact inf-sup-distrib2*)

lemma *ne-down-idempotent*:
 $ne ((ne (R \downarrow)) \downarrow) = ne (R \downarrow)$
by (*auto simp: mr-simp*)

lemma *ne-up*:
 $(ne R) \uparrow = ne R * 1 \uparrow$
proof
show $(ne R) \uparrow \subseteq ne R * 1 \uparrow$
apply (*clarsimp simp: mr-simp*)
by (*metis UN-constant Un-insert-left insert-absorb*)

```

next
  show ne R * 1↑ ⊆ (ne R)↑
  proof (clar simp simp: mr-simp)
    fix a B f
    assume 1: (a,B) ∈ R and 2: B ≠ {} and ∀ b∈B . ∃ C . f b = insert b C
    hence B ⊆ (∪ x∈B . f x)
      by blast
    thus ∃ D . (∃ C . (∪ x∈B . f x) = D ∪ C) ∧ (a,D) ∈ R ∧ D ≠ {}
      using 1 2 by blast
  qed
qed

```

```

lemma ne-dist-down-sp:
  ne (R↓ * S) = ne (R↓) * ne S
  proof (rule antisym)
    show ne (R↓ * S) ⊆ ne (R↓) * ne S
    proof (clar simp simp: mr-simp)
      fix a C f D x
      assume 1: (a,C) ∈ R and 2: ∀ b∈C∩D . (b,f b) ∈ S and 3: f x ≠ {} and 4:
      x ∈ C and 5: x ∈ D
      let ?B = { b∈C∩D . f b ≠ {} }
      have 6: ∃ C . (∃ D . ?B = C ∩ D) ∧ (a,C) ∈ R
        using 1 by auto
      have 7: ?B ≠ {}
        using 3 4 5 by auto
      have 8: ∀ b∈?B . (b,f b) ∈ S ∧ f b ≠ {}
        using 2 by auto
      have (∪ x∈C∩D . f x) = (∪ x∈?B . f x)
        by auto
      thus ∃ B . (∃ C . (∃ D . B = C ∩ D) ∧ (a,C) ∈ R) ∧ B ≠ {} ∧ (∃ g . (∀ b∈B
      . (b,g b) ∈ S ∧ g b ≠ {})) ∧ (∪ x∈C∩D . f x) = (∪ x∈B . g x))
        using 6 7 8 by blast
    qed
next
  show ne (R↓) * ne S ⊆ ne (R↓ * S)
    by (clar simp simp: mr-simp) blast
qed

```

```

lemma total-ne-down-dist-sp:
  total T ⇒ ne ((R * T)↓) = ne (R↓) * ne (T↓)
  by (simp add: ne-dist-down-sp total-down-dist-sp)

lemma inner-univalent-char:
  inner-univalent S ⇔ (∀ R . fus R = fus S ∧ R ⊑ S → ne R = ne S)
  proof
    assume 1: inner-univalent S
    show ∀ R . fus R = fus S ∧ R ⊑ S → ne R = ne S
    proof (rule allI, rule impI)
      fix R

```

```

assume 2: fus R = fus S  $\wedge$  R ⊑↓ S
show ne R = ne S
proof (rule antisym)
  show ne R ⊆ ne S
  proof
    fix x
    assume 3: x ∈ ne R
    from this obtain a B where 4: x = (a,B)  $\wedge$  x ∈ R  $\wedge$  B ≠ {}
      by (metis Int-iff Int-lower2 inner-total-2 surj-pair)
    from this obtain C where 5: B ⊆ C  $\wedge$  (a,C) ∈ S
      using 2 by (meson lower-less-eq)
    from this obtain b where C = {b}
      using 1 4 by (metis Un-empty inner-univalent is-singletonI'
is-singleton-the-elem subset-Un-eq)
      hence B = C
      using 4 5 by blast
      thus x ∈ ne S
      using 3 4 5 by blast
  qed
next
  show ne S ⊆ ne R
  proof
    fix x
    assume 6: x ∈ ne S
    from this obtain a B where 7: x = (a,B)  $\wedge$  x ∈ S  $\wedge$  B ≠ {}
      by (metis Int-absorb Int-iff inner-total-2
semilattice-inf-class.inf.absorb-iff2 surj-pair)
    from this obtain b where 8: B = {b}
      using 1 by (meson antisym card-1-singletonE inner-univalent-2
nonempty-set-card)
      from this obtain C where 9: b ∈ C  $\wedge$  (a,C) ∈ fus S
      using 7 by (meson fusion-lower-increasing insert-subset lower-less-eq)
      hence (a,C) ∈ fus R
      using 2 by simp
      hence C = ⋃{ D . (a,D) ∈ R }
        by (clar simp simp: mr-simp)
      from this obtain D where 10: b ∈ D  $\wedge$  (a,D) ∈ R
        using 9 by blast
      from this obtain E where 11: D ⊆ E  $\wedge$  (a,E) ∈ S
        using 2 by (meson lower-less-eq)
      from this obtain c where E = {c}
        using 1 10 by (metis antisym card-1-singletonE empty-iff
inner-univalent-2 nonempty-set-card subsetI)
        hence D = {b}
        using 10 11 by blast
        thus x ∈ ne R
        using 6 7 8 10 by blast
  qed
qed

```

```

qed
next
  assume 12:  $\forall R . \text{fus } R = \text{fus } S \wedge R \sqsubseteq \downarrow S \longrightarrow \text{ne } R = \text{ne } S$ 
  show inner-univalent  $S$ 
  proof (unfold inner-univalent, intro allI, rule impI)
    fix  $a b c B$ 
    assume 13:  $(a, B) \in S \wedge b \in B \wedge c \in B$ 
    let  $?R = \{ (f, \{e\}) \mid f e . \exists C . (f, C) \in S \wedge e \in C \}$ 
    have 14:  $\text{fus } ?R = \text{fus } S$ 
      apply (clar simp simp: mr-simp)
      by blast
    have  $?R \sqsubseteq \downarrow S$ 
      using lower-less-eq by fastforce
    hence  $\text{ne } ?R = \text{ne } S$ 
      using 12 14 by simp
    hence  $(a, B) \in ?R$ 
      using 13 by (smt (verit, del-insts) empty-iff mem-Collect-eq non-empty)
    thus  $b = c$ 
      using 13 by blast
  qed
qed

lemma ne-dist-oU:
  ne ( $\bigcup X$ ) =  $\bigcup (\text{non-empty} ` X)$ 
  by blast

```

7.4 Fission

```

lemma fission-set:
  fis  $R = \{ (a, \{b\}) \mid a b . \exists B . (a, B) \in R \wedge b \in B \}$ 
  unfolding fis-set Image-singleton
  by simp

```

```
declare fission-set [mr-simp]
```

```

lemma fission-var:
  fis  $R = R \downarrow \cap A_{\cup\cup}$ 
  apply (clar simp simp: mr-simp)
  by blast

```

```

lemma fission-lower-decreasing:
  fis  $R \sqsubseteq \downarrow R$ 
  by (simp add: fission-var)

```

```

lemma fission-inner-deterministic:
  inner-deterministic (fis  $R$ )
  by (simp add: fission-var inner-deterministic-atoms)

```

```
lemma fission-greatest:
```

```

assumes  $S \sqsubseteq \downarrow R$ 
and inner-deterministic  $S$ 
shows  $S \sqsubseteq \downarrow \text{fis } R$ 
proof (clar simp simp: mr-simp)
fix  $a B$ 
assume  $1: (a, B) \in S$ 
from this obtain  $b$  where  $2: B = \{b\}$ 
using assms(2) by (meson card-1-singletonE inner-deterministic-2)
from 1 obtain  $C$  where  $B \subseteq C \wedge (a, C) \in R$ 
using assms(1) by (meson lower-less-eq)
thus  $\exists C . (\exists D . B = C \cap D) \wedge (\exists b . C = \{b\} \wedge (\exists E . (a, E) \in R \wedge b \in E))$ 
using 2 by auto
qed

lemma fission-unique:
assumes  $\forall R . f R \sqsubseteq \downarrow R$ 
and  $\forall R . \text{inner-deterministic } (f R)$ 
and  $\forall R S . S \sqsubseteq \downarrow R \wedge \text{inner-deterministic } S \longrightarrow S \sqsubseteq \downarrow f R$ 
shows  $f T = \text{fis } T$ 
apply (rule inner-deterministic-lower-eq)
apply (simp add: assms(2))
apply (simp add: fission-inner-deterministic)
by (simp add: assms fission-greatest fission-inner-deterministic
fission-lower-decreasing)

lemma fission-lower-isotone:
 $R \sqsubseteq \downarrow S \implies \text{fis } R \sqsubseteq \downarrow \text{fis } S$ 
by (meson fission-greatest fission-inner-deterministic fission-lower-decreasing
lower-transitive)

lemma fission-idempotent:
 $\text{fis } (\text{fis } R) = \text{fis } R$ 
by (metis comp-apply fis-fis)

lemma fission-top:
 $\text{fis } U = A_{\cup \cup}$ 
using fission-var top-down top-upper-least by fastforce

lemma fission-down:
 $\text{fis } R = \text{fis } (R \downarrow)$ 
by (simp add: fission-var)

lemma fission-ne-fixpoint:
 $\text{fis } R = \text{ne } (\text{fis } R)$ 
using fission-inner-deterministic by blast

lemma fission-down-ne-fixpoint:
 $\text{fis } R = \text{ne } ((\text{fis } R) \downarrow)$ 
by (metis fission-inner-deterministic fission-ne-fixpoint fusion-down)

```

```

inner-univalent-char lower-ii-decreasing)

lemma fission-inner-deterministic-fixpoint:
  inner-deterministic R  $\longleftrightarrow$  R = fis R
  apply (rule iffI)
  apply (metis comp-eq-dest-lhs fission-lower-decreasing fission-ne-fixpoint fus-fis
inner-univalent-char le-iff-inf)
  using fission-inner-deterministic by auto

lemma fission-sp-subdist:
  fis (R * S)  $\subseteq$  fis R * fis S
proof
  fix x
  assume x ∈ fis (R * S)
  from this obtain a b B where 1: x = (a,{b})  $\wedge$  (a,B) ∈ R * S  $\wedge$  b ∈ B
    by (smt CollectD fission-set)
  from this obtain C f where 2: (a,C) ∈ R  $\wedge$  (∀ c ∈ C . (c,f c) ∈ S)  $\wedge$  B =
  ∪ {f c | c ∈ C}
    by (simp add: mr-simp) blast
  from this obtain c where 3: b ∈ f c  $\wedge$  c ∈ C
    using 1 by blast
  let ?B = {c}
  let ?f = λx . {b}
  have 4: (a,?B) ∈ fis R
    using 2 3 fission-set by blast
  have 5: ∀ b ∈ ?B . (b,?f b) ∈ fis S
    using 2 3 fission-set by blast
  have {b} = ∪ {?f b | b . b ∈ ?B}
    by simp
  hence ∃ f . (∀ b ∈ ?B . (b,f b) ∈ fis S)  $\wedge$  {b} = ∪ {f b | b . b ∈ ?B}
    using 5 by auto
  hence (a,{b}) ∈ fis R * fis S
    apply (unfold s-prod-def)
    using 4 by auto
  thus x ∈ fis R * fis S
    using 1 by simp
qed

lemma fission-sp-total-dist:
  assumes total T
  shows fis (R * T) = fis R * fis T
  by (metis assms atoms-sp-idempotent fis-lax fission-var sp-oi-subdist-2
subset-antisym total-down-dist-sp)

lemma fission-dist-ou:
  fis (R ∪ S) = fis R ∪ fis S
  by (simp add: down-dist-ou fission-var inf-sup-distrib2)

lemma fission-sp-iu-unit:

```

```

fis (R * 1UU) = {}
by (metis c-nc down-sp fission-lower-decreasing nu-def nu-fis nu-fis-var
s-prod-zerol subset-empty)

lemma fission-fusion-lower-decreasing:
  fis (fus R) ⊑↓ R
  apply (clar simp simp: mr-simp)
  by blast

lemma fusion-fission-lower-increasing:
  R ⊑↓ fus (fis R)
  apply (clar simp simp: mr-simp)
  by blast

lemma fission-fusion-galois:
  fis R ⊑↓ S ↔ R ⊑↓ fus S
  apply (rule iffI)
  apply (meson fission-fusion-lower-increasing fusion-lower-isotone
lower-transitive)
  by (meson fission-fusion-lower-decreasing fission-lower-isotone lower-transitive)

lemma fission-fusion:
  fis (fus R) = fis R
  by (metis fission-fusion-lower-decreasing fission-idempotent
fission-inner-deterministic fission-lower-isotone fusion-lower-increasing
inner-deterministic-lower-eq)

lemma fusion-fission:
  fus (fis R) = fus R
  by (metis comp-def fus-fis)

lemma same-fusion-fission-lower:
  fus R = fus S ==> fis R ⊑↓ S
  by (metis fission-fusion-galois fusion-lower-increasing)

lemma fission-below-ne-down-fusion:
  fis R ⊆ ne ((fus R)↓)
  using fission-fusion fission-inner-deterministic fission-lower-decreasing by blast

lemma ne-fusion-fission:
  (ne ((fus R)↓))↑ = (fis R)↑
  by (metis (mono-tags, lifting) atoms-solution fission-below-ne-down-fusion
fission-fusion oi-down-sub-up subset-trans upper-eq-up upper-reflexive fission-var)

lemma fission-up-ne-down-up:
  (fis R)↑ = (ne (R↓))↑
  by (metis (mono-tags, lifting) atoms-solution fission-ne-fixpoint fission-top
oi-down-sub-up semilattice-inf-class.inf-le2 semilattice-inf-class.inf-left-commute
subset-trans upper-eq-up fission-var)

```

```

lemma fusion-idempotent:
  fus (fus R) = fus R
  by (metis fission-fusion fusion-fission)

lemma fission-dist-oU:
  fis (( $\bigcup$  X)) =  $\bigcup$  (fis ' X)
  by (metis (no-types, lifting) SUP-cong UN-simps(4) fission-var ii-right-dist-oU)

7.5 Co-fusion and co-fission

definition co-fusion :: ('a,'b) mrel  $\Rightarrow$  ('a,'b) mrel ( $\langle \prod \prod \rightarrow [80] 80 \rangle$ ) where
   $\prod \prod R \equiv \{ (a,B) . B = \bigcap \{ C . (a,C) \in R \} \}$ 

declare co-fusion-def [mr-simp]

lemma co-fusion-upper-decreasing:
   $\prod \prod R \sqsubseteq \uparrow R$ 
  apply (clar simp simp: mr-simp)
  by blast

lemma co-fusion-deterministic:
  deterministic ( $\prod \prod R$ )
  by (simp add: deterministic-set co-fusion-def)

lemma co-fusion-greatest:
  assumes S  $\sqsubseteq \uparrow R$ 
  and deterministic S
  shows S  $\sqsubseteq \uparrow \prod \prod R$ 
  proof (clar simp simp: mr-simp)
    fix a
    from assms(2) obtain B where 1:  $(a,B) \in S$ 
      by (meson deterministic-set)
    hence B  $\subseteq \bigcap \{ C . (a,C) \in R \}$ 
      using assms deterministic-upper by (smt (verit, ccfv-threshold) Inter-greatest
      mem-Collect-eq)
    thus  $\exists B . (\exists D . \bigcap \{ C . (a,C) \in R \} = B \cup D) \wedge (a,B) \in S$ 
      using 1 by blast
  qed

lemma co-fusion-unique:
  assumes  $\forall R . f R \sqsubseteq \uparrow R$ 
  and  $\forall R . \text{deterministic } (f R)$ 
  and  $\forall R S . S \sqsubseteq \uparrow R \wedge \text{deterministic } S \longrightarrow S \sqsubseteq \uparrow f R$ 
  shows  $f T = \prod \prod T$ 
  apply (rule univalent-upper-eq)
  using assms(2) deterministic-def apply blast
  using co-fusion-deterministic deterministic-def apply blast
  by (simp add: assms co-fusion-deterministic co-fusion-greatest)

```

co-fusion-upper-decreasing)

```
lemma co-fusion-up-char:
  ( $\sqcap\sqcap R$ ) $\uparrow = -((-(R\uparrow) \cap A_{\sqcap\sqcap})\downarrow)$ 
proof
  show ( $\sqcap\sqcap R$ ) $\uparrow \subseteq -((-(R\uparrow) \cap A_{\sqcap\sqcap})\downarrow)$ 
  apply (clar simp simp: mr-simp)
  by blast
next
  show  $-((-(R\uparrow) \cap A_{\sqcap\sqcap})\downarrow) \subseteq (\sqcap\sqcap R)\uparrow$ 
  proof (clar simp simp: mr-simp)
    fix a A
    assume 1:  $\forall B . (\forall C . A \neq B \cap C) \vee (\exists C . (\exists D . B = C \cup D) \wedge (a, C) \in R) \vee (\forall b . B \neq \text{UNIV} - \{b\})$ 
    have  $\bigcap \{ C . (a, C) \in R \} \subseteq A$ 
    proof
      fix x
      assume  $x \in \bigcap \{ C . (a, C) \in R \}$ 
      hence  $\forall C . A \neq (\text{UNIV} - \{x\}) \cap C$ 
        using 1 by blast
      thus  $x \in A$ 
        by blast
    qed
    thus  $\exists D . A = \bigcap \{ C . (a, C) \in R \} \cup D$ 
      by auto
    qed
  qed
```

```
lemma co-fusion-down-char:
  ( $\sqcap\sqcap R$ ) $\downarrow = -((\sim(R\uparrow) \cap A_{\sqcup\sqcup})\uparrow)$ 
proof
  show ( $\sqcap\sqcap R$ ) $\downarrow \subseteq -((\sim(R\uparrow) \cap A_{\sqcup\sqcup})\uparrow)$ 
  apply (clar simp simp: mr-simp)
  by blast
next
  show  $-((\sim(R\uparrow) \cap A_{\sqcup\sqcup})\uparrow) \subseteq (\sqcap\sqcap R)\downarrow$ 
  proof (clar simp simp: mr-simp)
    fix a A
    assume 1:  $\forall C . (\forall D . A \neq C \cup D) \vee (\forall B . (\forall D . -C \neq B \cup D) \vee (a, B) \notin R) \vee (\forall b . C \neq \{b\})$ 
    have  $A \subseteq \bigcap \{ C . (a, C) \in R \}$ 
    proof
      fix x
      assume  $x \in A$ 
      hence  $\forall B . (\forall D . -\{x\} \neq B \cup D) \vee (a, B) \notin R$ 
        using 1 by blast
      thus  $x \in \bigcap \{ C . (a, C) \in R \}$ 
        by blast
    qed
```

thus $\exists C . A = \bigcap \{ C . (a, C) \in R \} \cap C$
by auto
qed
qed

lemma *co-fusion-down-char-2*:
 $(\prod \prod R) \downarrow = -(((R^\uparrow \cap A_{\cap\cap}) \odot \sim 1)^\uparrow)$
by (*metis co-fusion-down-char ic-co-atoms ic-cp-ic-unit ic-dist-oi*)

lemma *co-fusion-char*:
 $\prod \prod R = -((-(R^\uparrow) \cap A_{\cap\cap}) \downarrow) \cap -((\sim(R^\uparrow) \cap A_{\cup\cup})^\uparrow)$
by (*metis deterministic-def co-fusion-deterministic co-fusion-down-char co-fusion-up-char univalent-convex*)

lemma *co-fusion-char-2*:
 $\prod \prod R = -((-(R^\uparrow) \cap A_{\cap\cap}) \downarrow) \cap -(((R^\uparrow \cap A_{\cap\cap}) \odot \sim 1)^\uparrow)$
using *co-fusion-char co-fusion-down-char co-fusion-down-char-2* **by** *blast*

lemma *co-fusion-upper-isotone*:
 $R \sqsubseteq \uparrow S \implies \prod \prod R \sqsubseteq \uparrow \prod \prod S$
by (*meson co-fusion-deterministic co-fusion-greatest co-fusion-upper-decreasing upper-transitive*)

lemma *co-fusion-ii-idempotent*:
 $\prod \prod R \cap \prod \prod R = \prod \prod R$
by (*metis deterministic-def co-fusion-deterministic univalent-ii-idempotent*)

lemma *co-fusion-up*:
 $\prod \prod R = \prod \prod (R^\uparrow)$
by (*simp add: co-fusion-char*)

lemma *co-fusion-ii-total*:
 $total T \implies T \cap \prod \prod T = \prod \prod T$
by (*meson co-fusion-deterministic co-fusion-upper-decreasing deterministic-def total-univalent-upper-ii*)

lemma *co-fusion-deterministic-fixpoint*:
deterministic $R \longleftrightarrow R = \prod \prod R$
apply (*rule iffI*)
apply (*metis deterministic-def co-fusion-deterministic co-fusion-greatest co-fusion-ii-total ii-commute total-univalent-upper-ii upper-reflexive*)
by (*metis co-fusion-deterministic*)

abbreviation *co-fission* :: $('a, 'b) mrel \Rightarrow ('a, 'b) mrel (\langle at_{\cap\cap} \rightarrow [80] 80 \rangle) **where**
 $at_{\cap\cap} R \equiv R^\uparrow \cap A_{\cap\cap}$$

lemma *co-fission*:
 $at_{\cap\cap} R = \{ (a, B) \mid a \in B . (\exists b . -B = \{b\}) \wedge (\exists C . (a, C) \in R \wedge C \subseteq B) \}$
apply (*clar simp simp: mr-simp*)

by *blast*

declare co-fission [mr-simp]

lemma co-fission-upper-increasing:

$R \sqsubseteq \uparrow at_{\cap\cap} R$

by (fact semilattice-inf-class.inf-le1)

lemma co-fission-ic-inner-deterministic:

inner-deterministic ($\sim at_{\cap\cap} R$)

by (simp add: ic-co-atoms ic-dist-oi inner-deterministic-atoms)

lemma co-fission-least:

assumes $R \sqsubseteq \uparrow S$

and inner-deterministic ($\sim S$)

shows $at_{\cap\cap} R \sqsubseteq \uparrow S$

proof (clar simp simp: mr-simp)

fix $a B$

assume $1: (a, B) \in S$

hence $(a, -B) \in \sim S$

by (simp add: inner-complement-def)

from this obtain b **where** $2: -B = \{b\}$

using assms(2) **by** (meson card-1-singletonE inner-deterministic-2)

from 1 **obtain** C **where** $C \subseteq B \wedge (a, C) \in R$

using assms(1) **by** (meson upper-less-eq)

thus $\exists C . (\exists D . B = C \cup D) \wedge (\exists E . (\exists D . C = E \cup D) \wedge (a, E) \in R) \wedge (\exists b . C = UNIV - \{b\})$

using 2 **by** (metis Compl-eq-Diff-UNIV double-compl subset-Un-eq sup.idem)

qed

lemma co-fission-unique:

assumes $\forall R . R \sqsubseteq \uparrow f R$

and $\forall R . inner\text{-deterministic} (\sim f R)$

and $\forall R S . R \sqsubseteq \uparrow S \wedge inner\text{-deterministic} (\sim S) \longrightarrow f R \sqsubseteq \uparrow S$

shows $f T = at_{\cap\cap} T$

apply (rule ic-injective)

apply (rule inner-deterministic-lower-eq)

apply (simp add: assms(2))

apply (simp add: co-fission-ic-inner-deterministic)

by (meson assms co-fission-ic-inner-deterministic co-fission-least semilattice-inf-class.inf-le1 upper-ic-lower)

lemma co-fission-upper-isotone:

$R \sqsubseteq \uparrow S \implies at_{\cap\cap} R \sqsubseteq \uparrow at_{\cap\cap} S$

by (simp add: oi-subset-upper-left-antitone upper-transitive)

lemma co-fission-idempotent:

$at_{\cap\cap} (at_{\cap\cap} R) = at_{\cap\cap} R$

by (meson equalityI semilattice-inf-class.inf-le1 semilattice-inf-class.inf-le2)

semilattice-inf-class.le-inf-iff upper-reflexive upper-transitive)

lemma *co-fission-top*:

at_{nn} U = A_{nn}

using *top-lower-greatest U-par-idem top-down* **by** *blast*

lemma *co-fission-up*:

at_{nn} R = at_{nn} (R↑)

by *simp*

lemma *co-fission-ic-inner-deterministic-fixpoint*:

inner-deterministic (~R) ←→ R = at_{nn} R

apply (*rule iffI*)

apply (*simp add: fission-var fission-inner-deterministic-fixpoint ic-antidist-iu ic-co-atoms ic-dist-oi ic-injective ic-up*)

by (*metis co-fission-ic-inner-deterministic*)

lemma *co-fusion-co-fission-upper-decreasing*:

$\prod \prod (\text{at}_{nn} R) \sqsubseteq \uparrow R$

proof (*clarsimp simp: mr-simp*)

fix *a B*

assume *1: (a,B) ∈ R*

have $\bigcap \{ D . (\exists E . (\exists F . D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b . D = \text{UNIV} - \{b\}) \} \subseteq B$

proof

fix *x*

assume *2: x ∈ ∩ { D . (exists E . (exists F . D = E ∪ F) ∧ (a,E) ∈ R) ∧ (exists b . D = UNIV - {b}) }*

show *x ∈ B*

proof (*rule ccontr*)

let *?D = -{x}*

assume *3: x ∉ B*

hence *B ⊆ ?D*

by *simp*

hence $\bigcap \{ D . (\exists E . (\exists F . D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b . D = \text{UNIV} - \{b\}) \} \subseteq ?D$

using *1 by (smt CollectI Compl-eq-Diff-UNIV Inf-lower subset-Un-eq)*

thus *False*

using *2 by auto*

qed

qed

thus $\exists C . B = \bigcap \{ D . (\exists E . (\exists F . D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b . D = \text{UNIV} - \{b\}) \} \cup C$

by *auto*

qed

lemma *co-fission-co-fusion-upper-increasing*:

$R \sqsubseteq \uparrow \text{at}_{nn} (\prod \prod R)$

proof (*clarsimp simp: mr-simp*)

```

fix a b B
assume  $\bigcap \{ C . (a, C) \in R \} \cup B = UNIV - \{b\}$ 
hence  $b \notin \bigcap \{ C . (a, C) \in R \}$ 
  by blast
hence  $\exists C . b \notin C \wedge (a, C) \in R$ 
  by blast
thus  $\exists C . (\exists D . UNIV - \{b\} = C \cup D) \wedge (a, C) \in R$ 
  by blast
qed

```

```

lemma co-fusion-co-fission-galois:
 $\prod \prod R \sqsubseteq S \longleftrightarrow R \sqsubseteq_{at_{\cap\cap}} S$ 
apply (rule iffI)
apply (meson co-fission-co-fusion-upper-increasing co-fission-upper-isotone
upper-transitive)
by (meson co-fusion-co-fission-upper-decreasing co-fusion-upper-isotone
upper-transitive)

```

```

lemma co-fission-co-fusion:
 $at_{\cap\cap} (\prod \prod R) = at_{\cap\cap} R$ 
using co-fission-co-fusion-upper-increasing co-fission-idempotent
co-fission-upper-isotone co-fusion-upper-decreasing by blast

```

```

lemma co-fusion-co-fission:
 $\prod \prod (at_{\cap\cap} R) = \prod \prod R$ 
apply (rule antisym)
apply (metis deterministic-def co-fission-co-fusion
co-fission-co-fusion-upper-increasing co-fusion-co-fission-upper-decreasing
co-fusion-deterministic co-fusion-upper-isotone univalent-upper-eq-subset)
by (metis deterministic-def co-fission-co-fusion
co-fission-co-fusion-upper-increasing co-fusion-co-fission-upper-decreasing
co-fusion-deterministic co-fusion-upper-isotone univalent-upper-eq-subset)

```

```

lemma same-co-fusion-co-fission-upper:
 $\prod \prod R = \prod \prod S \implies S \sqsubseteq_{at_{\cap\cap}} R$ 
by (metis co-fusion-co-fission-galois co-fusion-upper-decreasing)

```

```

lemma co-fusion-idempotent:
 $\prod \prod (\prod \prod R) = \prod \prod R$ 
by (metis co-fission-co-fusion co-fusion-co-fission)

```

8 Modalities

8.1 Tests

```

abbreviation test :: ('a, 'a) mrel  $\Rightarrow$  bool where
  test R  $\equiv$  R  $\subseteq$  1

```

```

lemma test:

```

```

test R  $\longleftrightarrow$  ( $\forall a B . (a,B) \in R \longrightarrow B = \{a\}$ )
by (force simp: s-id-def)

lemma test-fix: test R  $\equiv$  R  $\cap$  1 $_{\sigma}$  = R
by (simp add: le-iff-inf)

lemma test-ou-closed:
test p  $\Longrightarrow$  test q  $\Longrightarrow$  test (p  $\cup$  q)
by (fact sup-least)

lemma test-oi-closed:
test p  $\Longrightarrow$  test (p  $\cap$  q)
by blast

abbreviation test-complement :: ('a,'a) mrel  $\Rightarrow$  ('a,'a) mrel ( $\wr \rightarrow [80]$  80)
where
 $\wr R \equiv -R \cap 1$ 

lemma test-complement-closed:
test ( $\wr p$ )
by simp

lemma test-double-complement:
test p  $\longleftrightarrow$  p =  $\wr \wr p$ 
by blast

lemma test-complement:
(a,{a})  $\in$   $\wr p \longleftrightarrow \neg (a,\{a\}) \in p$ 
by (simp add: s-id-def)

declare test-complement [mr-simp]

lemma test-complement-antitone:
assumes test p
shows p  $\subseteq$  q  $\longleftrightarrow$   $\wr q \subseteq \wr p$ 
using assms(1) by blast

lemma test-complement-huntington:
test p  $\Longrightarrow$  p =  $\wr (\wr p \cup \wr q) \cup \wr (\wr p \cup q)$ 
by blast

abbreviation test-implication :: ('a,'a) mrel  $\Rightarrow$  ('a,'a) mrel  $\Rightarrow$  ('a,'a) mrel
(infixl  $\leftrightarrow\!\!\!\rightarrow$  65) where
p  $\rightarrow$  q  $\equiv$   $\wr p \cup q$ 

lemma test-implication-closed:
test q  $\Longrightarrow$  test (p  $\rightarrow$  q)
by simp

```

```

lemma test-implication:
   $(a,\{a\}) \in p \rightarrow q \longleftrightarrow ((a,\{a\}) \in p \longrightarrow (a,\{a\}) \in q)$ 
  by (simp add: s-id-def)

declare test-implication [mr-simp]

lemma test-implication-left-antitone:
  assumes test p
  shows  $p \subseteq r \implies r \rightarrow q \subseteq p \rightarrow q$ 
  by blast

lemma test-implication-right-isotone:
  assumes test p
  shows  $q \subseteq r \implies p \rightarrow q \subseteq p \rightarrow r$ 
  by blast

lemma test-sp-idempotent:
  test p  $\implies p * p = p$ 
  by (metis d-rest-ax inf.order-iff s-subid-iff2)

lemma test-sp:
  assumes test p
  shows  $p * R = (p * U) \cap R$ 
  apply (clar simp simp: mr-simp)
  apply safe
    apply blast
  using assms subid-aux2 by fastforce+

lemma sp-test:
  test p  $\implies R * p = R \cap (U * p)$ 
  apply (rule antisym)
  apply (metis (no-types, lifting) U-par-idem inf.absorb-iff2 inf.idem le-inf-iff
  s-prod-idr sp-oi-subdist top-upper-least)
  using test-fix by (smt IntE s-prod-test-aux1 s-prod-test-aux2 subrell)

lemma sp-test-dist-oi:
  test p  $\implies (R \cap S) * p = (R * p) \cap (S * p)$ 
  by (smt Int-left-commute semilattice-inf-class.inf.assoc
  semilattice-inf-class.inf.right-idem sp-test)

lemma sp-test-dist-oi-left:
  test p  $\implies (R \cap S) * p = (R * p) \cap S$ 
  by (smt Int-commute semilattice-inf-class.inf.left-commute sp-test)

lemma sp-test-dist-oi-right:
  test p  $\implies (R \cap S) * p = R \cap (S * p)$ 
  by (metis semilattice-inf-class.inf.commute sp-test-dist-oi-left)

lemma sp-test-sp-oi-left:

```

```

test p  $\implies$   $(R \cap (U * p)) * T = R * p * T$ 
by (metis sp-test)

lemma sp-test-sp-oi-right:
test p  $\implies$   $R * ((p * U) \cap T) = R * p * T$ 
by (metis inf.orderE test-assoc1 test-sp)

lemma test-sp-ne:
test p  $\implies$   $p * ne R = ne (p * R)$ 
by (smt lattice-class.inf-sup-aci(1) lattice-class.inf-sup-aci(3) test-sp)

lemma ne-sp-test:
test p  $\implies$   $ne R * p = ne (R * p)$ 
by (fact sp-test-dist-oi-left)

lemma top-sp-test-down-closed:
assumes test p
shows  $U * p = (U * p) \downarrow$ 
proof -
have 1:  $p \cap 1_\sigma = p$ 
using assms by blast
hence  $(U * p) \downarrow = \{(a, A). (a, A) \in U \wedge (\forall a \in A. (a, \{a\}) \in p)\} \cap U$ 
by (smt (verit) Collect-cong case-prodI2 case-prod-conv s-prod-test-var)
also have ... =  $\{(a, A). \forall a \in A. (a, \{a\}) \in p\} \cap U$ 
by (smt (verit) Collect-cong ii-assoc lower-ii-down mem-Collect-eq split-cong
subsetD top-down)
also have ... =  $\{(a, A). (a, A) \in U \wedge (\forall a \in A. (a, \{a\}) \in p)\}$ 
by (auto simp: mr-simp)
also have ... =  $U * p$ 
using 1 by (smt (verit) Collect-cong case-prodI2 case-prod-conv
s-prod-test-var)
finally show ?thesis
by blast
qed

lemma oc-top-sp-test-up-closed:
test p  $\implies$   $-(U * p) = (-(U * p)) \uparrow$ 
by (metis antisym convex-reflexive disjoint-eq-subset-Compl inf-compl-bot
oi-down-up-iff semilattice-inf-class.inf.commute top-sp-test-down-closed)

lemma top-sp-test:
test p  $\implies$   $(a, B) \in U * p \longleftrightarrow (\forall b \in B. (b, \{b\}) \in p)$ 
using test-fix by (metis IntE UNIV-I s-prod-test sp-test)

lemma oc-top-sp-test:
test p  $\implies$   $(a, B) \in -(U * p) \longleftrightarrow (\exists b \in B. (b, \{b\}) \notin p)$ 
by (simp add: top-sp-test)

declare top-sp-test [mr-simp] oc-top-sp-test [mr-simp]

```

```

lemma oc-top-sp-test-0:
   $-1_{\cup\cup} * \ell p = ne(U * \ell p)$ 
  by (metis Int-lower1 semilattice-inf-class.inf.commute sp-test)

lemma oc-top-sp-test-1:
  assumes test p
  shows  $-(U * p) = (ne(U * \ell p))\uparrow$ 
  proof (rule antisym)
    show  $-(U * p) \subseteq (ne(U * \ell p))\uparrow$ 
    proof
      fix  $x::'c \times 'a$  set
      assume 1:  $x \in -(U * p)$ 
      from this obtain a B where 2:  $x = (a, B)$ 
      by force
      from this obtain c where 3:  $c \in B \wedge (c, \{c\}) \notin p$ 
      using 1 by (meson assms oc-top-sp-test)
      hence 4:  $(a, \{c\}) \in U * \ell p$ 
      by (metis singletonD test-complement test-complement-closed top-sp-test)
      have  $(a, \{c\}) \in -1_{\cup\cup}$ 
      using oc-top-sp-test by (smt (verit, del-insts) ComplI Int-iff assms
        boolean-algebra.conj-cancel-left inf.coboundedI2 p-id-zero s-prod-test-aux1
        singleton-iff)
      hence  $(a, \{c\}) \in ne(U * \ell p)$ 
      using 4 by simp
      thus  $x \in (ne(U * \ell p))\uparrow$ 
      using 2 3 by (metis (no-types, lifting) U-par-st singletonD subset-eq)
    qed
  next
    have  $(U * p)\downarrow = U * p$ 
    using assms top-sp-test-down-closed by auto
    also have ...  $\subseteq -(-1_{\cup\cup} * \ell p)$ 
    by (smt (verit) Compl-disjoint assms disjoint-eq-subset-Compl inf-commute
      oc-top-sp-test-0 p-id-zero s-prod-idl sp-test-dist-oi-right test-assoc1
      test-double-complement)
    also have ...  $= -ne(U * \ell p)$ 
    by (simp add: oc-top-sp-test-0)
    finally have  $U * p \subseteq -((ne(U * \ell p))\uparrow)$ 
    by (simp add: down-double-complement-up)
    thus  $(ne(U * \ell p))\uparrow \subseteq -(U * p)$ 
    by auto
  qed

lemma oc-top-sp-test-2:
  test p  $\implies -(U * p) = (-1_{\cup\cup} * \ell p)\uparrow$ 
  by (simp add: oc-top-sp-test-1 oc-top-sp-test-0)

lemma split-sp-test:
  assumes test p

```

```

shows  $R = (R * p) \cup (ne R \cap (ne (R\downarrow * \lambda p))\uparrow)$ 
proof (rule antisym)
show  $R \subseteq (R * p) \cup (ne R \cap (ne (R\downarrow * \lambda p))\uparrow)$ 
proof
fix x
assume 1:  $x \in R$ 
from this obtain a B where 2:  $x = (a, B)$ 
by force
show  $x \in (R * p) \cup (ne R \cap (ne (R\downarrow * \lambda p))\uparrow)$ 
proof (cases  $\forall b \in B . (b, \{b\}) \in p$ )
case True
hence  $(a, B) \in U * p$ 
by (simp add: assms top-sp-test)
thus ?thesis
using 1 2 by (metis Int-iff UnCI assms sp-test)
next
case False
from this obtain b where 3:  $\{b\} \subseteq B \wedge (b, \{b\}) \notin p$ 
by auto
hence  $(a, \{b\}) \in R\downarrow$ 
using 1 2 down by fastforce
hence  $(a, \{b\}) \in R\downarrow * \lambda p$ 
using 3 by (metis s-prod-test-aux2 singletonD test-complement)
hence  $(a, \{b\}) \in ne (R\downarrow * \lambda p)$ 
by (simp add: non-empty)
hence  $(a, B) \in (ne (R\downarrow * \lambda p))\uparrow$ 
using 3 by (meson U-par-st)
thus ?thesis
using 1 2 3 non-empty by auto
qed
qed
next
show  $(R * p) \cup (ne R \cap (ne (R\downarrow * \lambda p))\uparrow) \subseteq R$ 
using assms sp-test by auto
qed

lemma top-sp-test-down-iff-1:
assumes test p
shows  $R \subseteq U * p \longleftrightarrow R\downarrow \subseteq U * p$ 
by (smt (verit, del-insts) assms down-order-lower top-sp-test-down-closed)

lemma test-ne:
test p  $\implies$  ne p = p
using inner-deterministic-sp-unit by blast

lemma ne-test-up:
test p  $\implies$  ne ( $p\uparrow$ ) =  $p\uparrow$ 
by (metis atoms-solution ne-equality test-ne up-idempotent up-isotone)

```

```

lemma ne-sp-test-up:
  test p  $\implies$  (ne (R * p)) $\uparrow$  = ne R * p $\uparrow$ 
  using test-fix by (smt ne-up sp-test-dist-oi-left test-assoc1 test-ne)

lemma ne-down-sp-test-up:
  test p  $\implies$  ne (R $\downarrow$  * p $\uparrow$ ) = ne (R $\downarrow$ ) * p $\uparrow$ 
  by (simp add: ne-dist-down-sp ne-test-up)

lemma test-up-sp:
  test p  $\implies$  p $\uparrow$  = p * 1 $\uparrow$ 
  by (metis ne-up test-ne)

lemma top-test-oi-top-complement:
  test p  $\implies$  (U * p)  $\cap$  (U * l p) = 1 $_{\cup\cup}$ 
  by (smt (verit) Compl-disjoint U-par-idem inf.absorb-iff2 inf-commute p-id-zero
    s-prod-idl sp-test-dist-oi-right test-assoc1 top-upper-least)

lemma sp-test-oi-complement:
  test p  $\implies$  (R * p)  $\cap$  (R * l p) = R  $\cap$  1 $_{\cup\cup}$ 
  by (smt semilattice-inf-class.inf-idem sp-test sp-test-dist-oi-left
    sp-test-dist-oi-right test-complement-closed top-test-oi-top-complement)

lemma ne-top-sp-test-complement:
  assumes test p
  shows ne (U * p) * l p = {}
  by (metis Compl-disjoint Int-assoc assms oc-top-sp-test-0
    semilattice-inf-class.inf-le2 sp-test-dist-oi-right top-test-oi-top-complement)

lemma complement-test-sp-top:
  assumes test p
  shows -(p * U) = l p * U
  proof -
    have -(p * U) = -{(a,A). (a,{a})  $\in$  p  $\wedge$  (a,A)  $\in$  U}
      by (metis (no-types, lifting) Collect-cong assms inf.orderE split-cong
        test-s-prod-var)
    also have ... = -{(a,A). (a,{a})  $\in$  p}
      using top-upper-least by auto
    also have ... = {(a,A). (a,{a})  $\notin$  p}
      by force
    also have ... = {(a,A). (a,{a})  $\in$  l p}
      by (meson test-complement)
    also have ... = {(a,A). (a,{a})  $\in$  l p  $\wedge$  (a,A)  $\in$  U}
      using U-par-idem top-upper-least by auto
    also have ... = l p * U
      by (simp add: test-s-prod-var)
    finally show ?thesis
    .
  qed

```

```

lemma top-sp-test-shunt:
  assumes test p
  shows  $R \subseteq U * p \longrightarrow R * \lambda p \subseteq 1_{\cup\cup}$ 
  by (metis assms inf.absorb-iff1 sp-test sp-test-dist-oi test-complement-closed
top-test-oi-top-complement)

lemma top-sp-test-down-iff-2:
  assumes test p
  shows  $R \downarrow \subseteq U * p \longleftrightarrow R \downarrow * \lambda p \subseteq 1_{\cup\cup}$ 
proof
  assume  $R \downarrow \subseteq U * p$ 
  thus  $R \downarrow * \lambda p \subseteq 1_{\cup\cup}$ 
    using assms top-sp-test-shunt by blast
next
  assume 1:  $R \downarrow * \lambda p \subseteq 1_{\cup\cup}$ 
  have  $R \subseteq U * p$ 
  proof
    fix x
    assume  $x \in R$ 
    from this obtain a B where 2:  $x = (a, B) \wedge x \in R$ 
      by force
    have  $\forall b \in B . (b, \{b\}) \in p$ 
    proof
      fix b
      assume 3:  $b \in B$ 
      have  $(b, \{b\}) \notin \lambda p$ 
      proof
        assume 4:  $(b, \{b\}) \in \lambda p$ 
        have  $(a, \{b\}) \in R \downarrow$ 
          using 2 3 down by fastforce
        hence  $(a, \{b\}) \in R \downarrow * \lambda p$ 
          using 4 by (simp add: s-prod-test)
        thus False
        using 1 by (metis Pair-inject domain-pointwise insert-not-empty
p-subid-iff)
      qed
      thus  $(b, \{b\}) \in p$ 
        by (meson test-complement)
    qed
    thus  $x \in U * p$ 
      using 2 by (simp add: assms top-sp-test)
  qed
  thus  $R \downarrow \subseteq U * p$ 
    using assms top-sp-test-down-iff-1 by blast
qed

lemma top-sp-test-down-iff-3:
   $R \downarrow * \lambda p \subseteq 1_{\cup\cup} \longleftrightarrow \text{ne } (R \downarrow) * \lambda p \subseteq \{\}$ 
  by (simp add: disjoint-eq-subset-Compl ne-sp-test)

```

```

lemma top-sp-test-down-iff-4:
  assumes test p
  shows  $R \downarrow \cap (U * \{p\}) \subseteq 1_{\cup\cup} \longleftrightarrow R \downarrow \subseteq 1_{\cup\cup} \cup (U * p)$ 
  by (metis assms lattice-class.sup-inf-absorb semilattice-inf-class.inf-le2 sp-test
sup-commute top-sp-test-down-iff-2 top-test-oi-top-complement)

lemma top-sp-test-down-iff-5:
  assumes test p
  shows  $R \downarrow \subseteq U * p \longleftrightarrow R \downarrow \subseteq 1_{\cup\cup} \cup (U * p)$ 
  by (metis assms semilattice-inf-class.inf-le1 sup.absorb2
top-test-oi-top-complement)

lemma iu-test-sp-left-zero:
  assumes  $q \subseteq 1_{\cup\cup}$ 
  shows  $q * R = q$ 
  by (metis assms p-id-assoc2 p-subid-iff s-prod-p-idl)

lemma test-iu-test-split:
   $t \subseteq 1 \cup 1_{\cup\cup} \longleftrightarrow (\exists p \in q . p \subseteq 1 \wedge q \subseteq 1_{\cup\cup} \wedge t = p \cup q)$ 
  by (meson subset-UnE sup.mono)

lemma test-iu-test-sp-assoc-1:
   $t \subseteq 1 \cup 1_{\cup\cup} \implies t * (R * S) = (t * R) * S$ 
  unfolding test-iu-test-split
  by (smt (verit, ccfv-threshold) inf.orderE p-id-assoc2 p-subid-iff s-prod-distr
s-prod-p-idl test-assoc2)

lemma test-iu-test-sp-assoc-2:
   $t \subseteq 1_{\cup\cup} \implies R * (t * S) = (R * t) * S$ 
  proof -
    assume  $1: t \subseteq 1_{\cup\cup}$ 
    have  $R * (t * S) = R * (t * \{\})$ 
    using 1 by (metis iu-test-sp-left-zero p-id-assoc2 s-prod-p-idl)
    also have  $\dots = (R * t) * \{\}$ 
    by (metis cl5 s-prod-idl)
    also have  $\dots \subseteq (R * t) * S$ 
    by (simp add: s-prod-isor)
    finally have  $R * (t * S) \subseteq (R * t) * S$ 
    .
    thus ?thesis
    by (simp add: s-prod-assoc1 set-eq-subset)
  qed

lemma test-iu-test-sp-assoc-3:
  assumes  $t \subseteq 1 \cup 1_{\cup\cup}$ 
  shows  $R * (t * S) = (R * t) * S$ 
  proof
    let ?g =  $\lambda b . \text{if } (b, \{b\}) \in t \wedge (b, \{b\}) \notin t \text{ then } \{b\} \text{ else } \{\}$ 

```

```

show  $R * (t * S) \subseteq (R * t) * S$ 
proof
fix  $x$ 
assume  $x \in R * (t * S)$ 
from this obtain  $a B C f$  where  $1: x = (a,C) \wedge (a,B) \in R \wedge (\forall b \in B . (b,f$ 
 $b) \in t * S) \wedge C = \bigcup \{ f b \mid b . b \in B \}$ 
by (simp add: mr-simp) blast
hence  $\forall b \in B . \exists D . (b,D) \in t \wedge (\exists g . (\forall e \in D . (e,g) \in S) \wedge f b = \bigcup \{ g e \mid$ 
 $e . e \in D \})$ 
by (simp add: mr-simp Setcompr-eq-image)
hence  $\exists Db . \forall b \in B . (b,Db) \in t \wedge (\exists g . (\forall e \in Db . (e,g) \in S) \wedge f b =$ 
 $\bigcup \{ g e \mid e . e \in Db \})$ 
by (rule bchoice)
from this obtain  $Db$  where  $2: \forall b \in B . (b,Db) \in t \wedge (\exists g . (\forall e \in Db . (e,g$ 
 $e) \in S) \wedge f b = \bigcup \{ g e \mid e . e \in Db \})$ 
by auto
let ?D =  $\bigcup \{ Db b \mid b . b \in B \}$ 
have  $\forall b \in B . (b,Db) \in t$ 
using 2 by auto
hence 3:  $\forall b \in B . Db b = \{b\} \vee Db b = \{\}$ 
using assms by (metis Pair-inject Un-iff domain-pointwise inf.orderE
p-subid-iff subid-aux2 test-iu-test-split)
have 4:  $(a,?D) \in R * t$ 
apply (simp add: mr-simp)
apply (rule exI[where ?x=B])
apply (rule conjI)
using 1 apply simp
apply (rule exI[where ?x=Db])
using 2 by auto
have 5:  $\forall b \in ?D . (b,f b) \in S$ 
proof
fix  $b$ 
assume  $b \in ?D$ 
hence  $b \in B \wedge Db b = \{b\}$ 
using 3 by auto
thus  $(b,f b) \in S$ 
using 2 by force
qed
have 6:  $C = \bigcup \{ f b \mid b . b \in ?D \}$ 
proof
show  $C \subseteq \bigcup \{ f b \mid b . b \in ?D \}$ 
proof
fix  $y$ 
assume  $y \in C$ 
from this 1 obtain  $b$  where 7:  $b \in B \wedge y \in f b$ 
by auto
hence  $Db b = \{b\}$ 
using 2 3 by blast
thus  $y \in \bigcup \{ f b \mid b . b \in ?D \}$ 

```

```

        using 7 by blast
qed
next
show  $\bigcup\{ f b \mid b . b \in ?D \} \subseteq C$ 
proof
fix y
assume  $y \in \bigcup\{ f b \mid b . b \in ?D \}$ 
from this obtain b where 8:  $b \in ?D \wedge y \in f b$ 
by auto
hence  $b \in B$ 
using 3 by auto
thus  $y \in C$ 
using 1 8 by auto
qed
qed
have  $(a,C) \in (R * t) * S$ 
using 4 5 6 apply (clar simp simp: mr-simp) by blast
thus  $x \in (R * t) * S$ 
using 1 by simp
qed
next
show  $(R * t) * S \subseteq R * (t * S)$ 
using s-prod-assoc1 by blast
qed

lemma test-iu-test-sp-assoc-4:
 $t \subseteq 1_{\cup\cup} \implies R * (S * t) = (R * S) * t$ 
by (metis cl5 iu-test-sp-left-zero)

lemma test-iu-test-sp-assoc-5:
assumes  $t \subseteq 1 \cup 1_{\cup\cup}$ 
shows  $R * (S * t) = (R * S) * t$ 
proof
show  $R * (S * t) \subseteq (R * S) * t$ 
proof
fix x
assume  $x \in R * (S * t)$ 
from this obtain a B C f where 1:  $x = (a,C) \wedge (a,B) \in R \wedge (\forall b \in B . (b,f \in S * t) \wedge C = \bigcup\{ f b \mid b . b \in B \})$ 
by (clar simp simp: mr-simp) blast
hence  $\forall b \in B . \exists D . (b,D) \in S \wedge (\exists g . (\forall e \in D . (e,g e) \in t) \wedge f b = \bigcup\{ g e \mid e . e \in D \})$ 
by (clar simp simp: mr-simp Setcompr-eq-image)
hence  $\exists D b . \forall b \in B . (b,D b) \in S \wedge (\exists g . (\forall e \in D b . (e,g e) \in t) \wedge f b = \bigcup\{ g e \mid e . e \in D b \})$ 
by (rule bchoice)
from this obtain Db where 2:  $\forall b \in B . (b,Db b) \in S \wedge (\exists g . (\forall e \in D b . (e,g e) \in t) \wedge f b = \bigcup\{ g e \mid e . e \in D b \})$ 
by auto

```

```

hence  $\exists gb . \forall b \in B . (\forall e \in Db b . (e, gb b e) \in t) \wedge f b = \bigcup \{ gb b e \mid e . e \in Db b \}$ 
    by (auto intro: bchoice)
from this obtain gb where 3:  $\forall b \in B . (\forall e \in Db b . (e, gb b e) \in t) \wedge f b = \bigcup \{ gb b e \mid e . e \in Db b \}$ 
    by auto
let ?g =  $\lambda x . \bigcup \{ gb b x \mid b . b \in B \wedge x \in Db b \}$ 
have 4:  $\forall b \in B . \forall e \in Db b . gb b e = \{\} \vee gb b e = \{e\}$ 
proof (intro ballI)
    fix b e
    assume b ∈ B e ∈ Db b
    hence (e, gb b e) ∈ t
        using 3 by simp
    thus gb b e = {} ∨ gb b e = {e}
        by (metis Un-iff assms domain-pointwise inf.orderE p-subid-iff prod.inject
            subid-aux2 test-iu-test-split)
    qed
hence  $\forall e . ?g e \subseteq \{e\}$ 
    by auto
hence 5:  $\forall e . ?g e = \{\} \vee ?g e = \{e\}$ 
    by auto
let ?D =  $\bigcup \{ Db b \mid b . b \in B \}$ 
have 6: (a, ?D) ∈ R * S
    apply (unfold s-prod-def)
    using 1 2 by blast
have 7:  $\forall b \in ?D . (b, ?g b) \in t$ 
proof
    fix e
    assume e ∈ ?D
from this obtain b where 8:  $b \in B \wedge e \in Db b$ 
    by auto
    show (e, ?g e) ∈ t
proof (cases ?g e = {})
    case True
        hence gb b e = {}
        using 8 by auto
        thus ?thesis
            using 3 8 True by metis
    next
        case False
        hence 9: ?g e = {e}
            using 5 by auto
from this obtain bb where 10:  $bb \in B \wedge e \in Db bb \wedge e \in gb bb e$ 
            by auto
        hence gb bb e = {e}
            using 4 by auto
        thus ?thesis
            using 3 9 10 by force
qed

```

```

qed
have 11:  $C = \bigcup\{ ?g e \mid e . e \in ?D \}$ 
proof
show  $C \subseteq \bigcup\{ ?g e \mid e . e \in ?D \}$ 
proof
fix  $y$ 
assume  $y \in C$ 
from this obtain  $b$  where 12:  $b \in B \wedge y \in f b$ 
using 1 by auto
from this 3 obtain  $e$  where  $e \in Db b \wedge y \in gb b e$ 
by auto
thus  $y \in \bigcup\{ ?g e \mid e . e \in ?D \}$ 
using 4 12 by auto
qed
next
show  $\bigcup\{ ?g e \mid e . e \in ?D \} \subseteq C$ 
proof
fix  $y$ 
assume  $y \in \bigcup\{ ?g e \mid e . e \in ?D \}$ 
from this obtain  $b e$  where 13:  $b \in B \wedge e \in Db b \wedge y \in gb b e$ 
by auto
hence  $y \in f b$ 
using 3 by auto
thus  $y \in C$ 
using 1 13 by auto
qed
qed
have  $(a, C) \in (R * S) * t$ 
apply (simp add: mr-simp)
apply (rule exI[where ?x=?D])
apply (rule conjI)
using 6 apply (simp add: mr-simp)
apply (rule exI[where ?x=?g])
using 7 11 by auto
thus  $x \in (R * S) * t$ 
using 1 by simp
qed
next
show  $(R * S) * t \subseteq R * (S * t)$ 
using s-prod-assoc1 by blast
qed

lemma inner-deterministic-sp-assoc:
assumes inner-univalent  $t$ 
shows  $t * (R * S) = (t * R) * S$ 
proof (rule antisym)
show  $t * (R * S) \subseteq (t * R) * S$ 
proof
fix  $x$ 

```

```

assume  $x \in t * (R * S)$ 
from this obtain  $a B D f$  where  $1: x = (a,D) \wedge (a,B) \in t \wedge (\forall b \in B . (b,f b) \in R * S) \wedge D = \bigcup \{ f b \mid b \in B \}$ 
by (simp add: mr-simp) blast
have  $(a,D) \in (t * R) * S$ 
proof (cases  $(a,B) \in 1_{\cup\cup}$ )
  case True
  hence  $B = \{\}$ 
  by (metis Pair-inject domain-pointwise iu-test-sp-left-zero order-refl)
  hence  $D = \{\}$ 
  using 1 by fastforce
  hence  $(a,D) \in t * (R * \{\})$ 
  using 1 <(B::'b::type set) = \{\}> s-prod-def by fastforce
  hence  $(a,D) \in (t * R) * \{\}$ 
  by (metis cl5 s-prod-idl)
  thus ?thesis
  using s-prod-isor by auto
next
  case False
  hence  $(a,B) \in A_{\cup\cup}$ 
  using 1 assms by blast
from this obtain  $b$  where  $2: B = \{b\}$ 
  by (smt atoms-def Pair-inject mem-Collect-eq)
  hence  $D = f b \wedge (b,f b) \in R * S$ 
  using 1 by auto
from this obtain  $C g$  where  $3: (b,C) \in R \wedge (\forall c \in C . (c,g c) \in S) \wedge D = \bigcup \{ g c \mid c \in C \}$ 
  by (clarsimp simp: mr-simp) blast
  have  $(a,C) \in t * R$ 
  apply (simp add: mr-simp)
  apply (rule exI[where ?x=B])
  using 1 2 3 by auto
  thus ?thesis
  using 3 s-prod-def by blast
qed
thus  $x \in (t * R) * S$ 
  using 1 by auto
qed
next
show  $(t * R) * S \subseteq t * (R * S)$ 
  using s-prod-assoc1 by blast
qed

lemma iu-unit-below-top-sp-test:
   $1_{\cup\cup} \subseteq U * R$ 
  by (clarsimp simp: mr-simp) force

lemma ne-oi-complement-top-sp-test-1:
   $\text{ne } (R \cap -(U * S)) = R \cap -(U * S)$ 

```

```

using iu-unit-below-top-sp-test by auto

lemma ne-oi-complement-top-sp-test-2:
  ne R ∩ -(U * S) = R ∩ -(U * S)
  using iu-unit-below-top-sp-test by auto

lemma schroeder-test:
  assumes test p
  shows R * p ⊆ S  $\longleftrightarrow$  -S * p ⊆ -R
  by (metis assms disjoint-eq-subset-Compl double-compl
    semilattice-inf-class.inf-commute sp-test-dist-oi-right)

lemma complement-test-sp-test:
  test p  $\implies$  -p * p ⊆ -1
  by (simp add: schroeder-test)

lemma test-sp-commute:
  test p  $\implies$  test q  $\implies$  p * q = q * p
  by (metis s-prod-idl sp-test-dist-oi-left sp-test-dist-oi-right test-fix)

lemma test-shunting:
  assumes test p
  and test q
  and test r
  shows p * q ⊆ r  $\longleftrightarrow$  p ⊆ r ∪ l q
proof
  assume 1: p * q ⊆ r
  have p = p * q ∪ p * l q
  by (smt (verit, del-insts) Int-Un-distrib assms(1,2) inf-sup-aci(1)
    inf-sup-ord(2) le-iff-inf s-distl-test s-prod-idr sup-neg-inf)
  also have ... ⊆ r ∪ l q
  using 1 by (metis assms(1) inf-le2 sup.mono test-sp)
  finally show p ⊆ r ∪ l q
  .
next
  assume p ⊆ r ∪ l q
  hence p * q ⊆ p * r
  by (smt (verit) assms boolean-algebra-class.boolean-algebra.double-compl
    inf.orderE inf-le1 le-inf-iff schroeder-test sup-neg-inf test-double-complement
    test-s-prod-is-meet test-sp-commute)
  also have ... ⊆ r
  using assms(1) test-sp by auto
  finally show p * q ⊆ r
  .
qed

lemma test-sp-is-iu:
  test p  $\implies$  test q  $\implies$  p * q = p ∪ ∪ q
  by (metis Int-assoc inf.right-idem p-prod-comm s-prod-idr test-p-prod-is-meet

```

test-sp)

```
lemma test-set:
  test p ==> p = { (a,{a}) | a . (a,{a}) ∈ p }
  using s-id-st by blast

lemma test-sp-is-ii:
  assumes test p
  and test q
  shows p * q = p ∩ q
proof -
  have p ∩ q = { (a,{a}) | a . (a,{a}) ∈ p } ∩ { (a,{a}) | a . (a,{a}) ∈ q }
  using assms test-set by blast
  also have ... = { (a,{a}) | a . (a,{a}) ∈ p } ∩ { (a,{a}) | a . (a,{a}) ∈ q }
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  apply (rule Int-greatest)
  apply (clar simp simp: mr-simp)
  by (clar simp simp: mr-simp)
  also have ... = p ∩ q
  using test-set[symmetric, OF assms(1)] test-set[symmetric, OF assms(2)] by
  simp
  finally show p * q = p ∩ q
  by (metis assms inf.orderE test-oi-closed test-s-prod-is-meet)
qed

lemma test-galois-1:
  assumes test p
  and test q
  shows p * q ⊆ r ↔ q ⊆ p → r
  by (metis (no-types, lifting) Int-Un-eq(2) assms inf.orderE sup-neg-inf
  test-double-complement test-s-prod-is-meet test-sp-commute)

lemma test-sp-shunting:
  assumes test p
  shows ⋲ p * R ⊆ {} ↔ R ⊆ p * R
  apply (rule iffI)
  apply (metis (no-types, opaque-lifting) assms complement-test-sp-top
  disjoint-eq-subset-Compl double-compl semilattice-inf-class.inf.absorb-iff2
  semilattice-inf-class.inf-commute semilattice-inf-class.inf-right-idem test-sp)
  by (metis (no-types, opaque-lifting) assms complement-test-sp-top
  disjoint-eq-subset-Compl double-compl semilattice-inf-class.inf-commute
  semilattice-inf-class.inf-le1 subset-antisym test-sp)

lemma test-oU-closed:
  ∀ p ∈ X . test p ==> test (⋃ X)
  by blast

lemma test-oI-closed:
```

$\exists p \in X . \text{test } p \implies \text{test } (\bigcap X)$
by blast

```

lemma sp-test-dist-oI:
  assumes test p
  and X ≠ {}
  shows (bigcap X) * p = (bigcap R ∈ X . R * p)
  apply (rule antisym)
  apply (clarsimp simp: mr-simp)
  apply blast
  apply (clarsimp simp: mr-simp)
  subgoal for a C proof –
    assume 1: ∀ R ∈ X . ∃ B . (a, B) ∈ R ∧ (∃ f . (∀ b ∈ B . (b, f b) ∈ p) ∧ C = ⋃(f ` B))
    have 2: (∀ R ∈ X . (a, C) ∈ R ∧ (∀ b ∈ C . (b, {b}) ∈ p))
    proof
      fix R
      assume R ∈ X
      from this obtain B where 3: (a, B) ∈ R ∧ (∃ f . (∀ b ∈ B . (b, f b) ∈ p) ∧ C = ⋃(f ` B))
      using 1 by force
      from this obtain f where 4: ∀ b ∈ B . (b, f b) ∈ p and 5: C = ⋃(f ` B)
      by auto
      hence 6: ∀ b ∈ B . f b = {b}
      using assms(1) test by blast
      hence 7: C = B
      using 5 by auto
      hence (a, C) ∈ R
      using 3 by auto
      thus (a, C) ∈ R ∧ (∀ b ∈ C . (b, {b}) ∈ p)
      using 4 6 7 by fastforce
    qed
    show ?thesis
    apply (rule exI[of - C])
    apply (rule conjI)
    using 2 apply simp
    apply (rule exI[of - λx . {x}])
    apply (rule conjI)
    using 2 assms(2) apply blast
    by simp
  qed
  done

```

```

lemma test-iU-is-iI:
  assumes ∀ i ∈ I . test (X i)
  and I ≠ {}
  shows ⋃ X|I = ⋂ X|I
  apply (rule antisym)
  apply (clarsimp simp: mr-simp)

```

```

subgoal for a f proof -
  assume 1:  $\forall i \in I . (a, f i) \in X i$ 
  hence  $\forall i \in I . f i = \{a\}$ 
    using assms(1) by (meson test)
  hence  $\bigcup(f ` I) = \bigcap(f ` I) \wedge (\forall i \in I . (a, f i) \in X i)$ 
    using 1 assms(2) by auto
  thus ?thesis
    by meson
qed
apply (clar simp simp: mr-simp)
by (metis (no-types, opaque-lifting) INF-eq-const SUP-eq-const assms test)

```

```

lemma test-iU-is-oI:
  assumes  $\forall i \in I . \text{test} (X i)$ 
  and  $I \neq \{\}$ 
  shows  $\bigcup \bigcup X | I = \bigcap (X ` I)$ 
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  apply (metis (no-types, opaque-lifting) SUP-eq-const assms test)
  apply (clar simp simp: mr-simp)
  by (metis UN-constant assms(2))

```

8.2 Domain and antidomain

```
declare Dom-def [mr-simp]
```

```
abbreviation aDom :: ('a, 'b) mrel  $\Rightarrow$  ('a, 'a) mrel where
  aDom R  $\equiv$  l Dom R
```

```
lemma ad-set: aDom R =  $\{(a, \{a\}) \mid a. \neg(\exists A. (a, A) \in R)\}$ 
  by (clar simp simp: mr-simp) force
```

```
lemma d-test:
  test (Dom R)
  unfolding Dom-def using s-id-def by fastforce
```

```
lemma ad-test:
  test (aDom R)
  by simp
```

```
lemma ad-expl:
  aDom R =  $-((R * 1_{\cup\cup}) \cup\cup 1) \cap 1$ 
  by (simp add: d-def-expl)
```

```
lemma ad-expl-2:
  aDom (R :: ('a, 'b) mrel) =  $-((R * (1_{\cup\cup} :: ('b, 'a) mrel)) \uparrow) \cap (1 :: ('a, 'a) mrel)$ 
proof
  have  $-((R * 1_{\cup\cup}) \uparrow) \cap 1 = -((R * 1_{\cup\cup}) \cup\cup U) \cap 1$ 
  by simp
```

```

also have ...  $\subseteq -((R * 1_{\cup\cup}) \cup\cup 1) \cap 1$ 
  by (metis c6 convex-increasing iu-commute iu-isotone iu-unit sp-unit-convex
test-complement-antitone upper-iu-increasing)
also have ... = aDom R
  by (simp add: d-def-expl)
finally show  $-((R * 1_{\cup\cup})^{\uparrow}) \cap 1 \subseteq aDom R$ 
  using l ((R::('a::type  $\times$  'b::type set) set) * 1_{\cup\cup})^{\uparrow}  $\subseteq l(R * 1_{\cup\cup} \cup\cup 1)$  by
blast
have aDom R = {(a,{a}) | a.  $\neg(\exists B. (a,B) \in R)$ }
  by (simp add: ad-set)
also have ...  $\subseteq \{(a,{a}) | a. (a,{a}) \notin (R * (p-id::('b,'a) mrel))^{\uparrow}\}$ 
  by (simp add: U-par-st domain-pointwise)
also have ... = {(a,{a}) | a. (a,{a})  $\in -(R * (p-id::('b,'a) mrel))^{\uparrow} \cap 1\}$ 
  using test-complement by fastforce
finally show aDom R  $\subseteq -((R * (1_{\cup\cup}::('b,'a) mrel))^{\uparrow}) \cap (1::('a,'a) mrel)$ 
  by blast
qed

lemma aDom:
aDom R = { (a,{a}) | a .  $\neg(\exists B . (a,B) \in R)$  }
by (simp add: ad-set)

declare aDom [mr-simp]

lemma d-down-oi-up-1:
Dom (R $\downarrow$   $\cap$  S) = Dom (R  $\cap$  S $\uparrow$ )
by (metis Int-commute d-def-expl domain-up-down-conjugate)

lemma d-down-oi-up-2:
Dom (R $\downarrow$   $\cap$  S) = Dom (R $\downarrow$   $\cap$  S $\uparrow$ )
by (simp add: d-down-oi-up-1)

lemma d-ne-down-dp-complement-test:
assumes test p
shows Dom (R  $\cap - (U * p)) = Dom (ne(R\downarrow) * l p)$ 
by (simp add: assms d-down-oi-up-1 oc-top-sp-test-0 oc-top-sp-test-1
sp-test-dist-oi-right)

lemma d-strict:
R = {}  $\longleftrightarrow$  Dom R = {}
using Dom-def by fastforce

lemma d-sp-strict:
R * S = {}  $\longleftrightarrow$  R * Dom S = {}
apply (clar simp simp: mr-simp)
apply safe
apply metis
by (metis UN-singleton)

```

```

lemma d-complement-ad:
  Dom R = $\wr$  aDom R
  using d-test by blast

lemma down-sp-below-iu-unit:
   $R \downarrow * S \subseteq 1_{\cup\cup} \longleftrightarrow R \subseteq U * aDom (ne S)$ 
proof –
  have  $R \downarrow * S \subseteq 1_{\cup\cup} \longleftrightarrow ne(R \downarrow * S) = \{\}$ 
    by (simp add: disjoint-eq-subset-Compl)
  also have ...  $\longleftrightarrow ne(R \downarrow) * ne S = \{\}$ 
    by (simp add: ne-dist-down-sp)
  also have ...  $\longleftrightarrow ne(R \downarrow) * Dom(ne S) = \{\}$ 
    using d-sp-strict by auto
  also have ...  $\longleftrightarrow ne(R \downarrow) * \wr aDom(ne S) = \{\}$ 
    by (metis d-complement-ad)
  also have ...  $\longleftrightarrow R \subseteq U * aDom(ne S)$ 
    by (metis top-sp-test-down-iff-1 top-sp-test-down-iff-2 top-sp-test-down-iff-3
ad-test subset-empty)
  finally show ?thesis
  .
qed

lemma ad-sp-bot:
  aDom R * R =  $\{\}$ 
  by (simp add: d-s-id-ax d-sp-strict inf-sup-aci(1) sp-test-dist-oi-left)

lemma sp-top-d:
   $R * U \subseteq Dom R * U$ 
  by (simp add: cl8-var iu-unit-up sp-upper-left-isotone)

lemma d-sp-top:
   $Dom(R * U) = Dom R$ 
  by (clar simp simp: mr-simp) blast

lemma d-down:
   $Dom(R \downarrow) = Dom R$ 
  by (metis U-par-idem d-down-oi-up-1 inf.orderE top-down top-lower-greatest)

lemma d-up:
   $Dom(R \uparrow) = Dom R$ 
  by (metis Int-absorb1 U-par-idem d-down-oi-up-1 top-down top-upper-least)

lemma d-isotone:
   $R \subseteq S \implies Dom R \subseteq Dom S$ 
  unfolding Dom-def by blast

lemma ad-antitone:
   $R \subseteq S \implies aDom S \subseteq aDom R$ 
  by (simp add: Int-commute d-isotone semilattice-inf-class.le-infl2)

```

lemma *d-dist-ou*:

Dom ($R \cup S$) = *Dom* $R \cup$ *Dom* S
unfolding *Dom-def* **by** *blast*

lemma *d-dist-iu*:

Dom ($R \cup\cup S$) = *Dom* $R *$ *Dom* S
by (*clar simp* *simp*: *mr-simp*) *auto*

lemma *d-dist-ii*:

Dom ($R \cap\cap S$) = *Dom* $R *$ *Dom* S
by (*metis* *antisym-conv* *d-U* *d-dist-iu* *d-down* *d-isotone* *ii-convex-iu* *s-prod-idr*)

lemma *d-loc*:

Dom ($R * \text{Dom } S$) = *Dom* ($R * S$)
apply (*clar simp* *simp*: *mr-simp*)
apply *safe*
apply *metis*
by (*metis* *UN-singleton*)

lemma *ad-loc*:

aDom ($R * \text{Dom } S$) = *aDom* ($R * S$)
by *simp*

lemma *d-ne-down*:

Dom (*ne* ($R \downarrow$)) = *Dom* (*ne* R)
by (*metis* *atoms-solution* *d-down-oi-up-1* *d-down-oi-up-2*)

lemma *ne-sp-ji-unit-up*:

ne R = $R \implies (R * 1_{\cup\cup})^\uparrow = R * U$
apply (*clar simp* *simp*: *mr-simp*)
apply *safe*
apply (*metis* (*no-types*, *lifting*) *Compl-iff* *IntE* *Inter-iff* *UNIV-I* *UN-simps(2)*
image-eqI *singletonI*)
apply *clar simp*
by (*metis* *SUP-bot* *sup-bot-left*)

lemma *ne-d-expl*:

ne R = $R \implies \text{Dom } R = R * U \cap 1$
by (*metis* *cl8-var* *d-def-expl* *d-test* *ne-sp-ji-unit-up* *test-sp*)

lemma *ne-a-expl*:

ne R = $R \implies \text{aDom } R = -(R * U) \cap 1$
by (*simp* *add*: *ad-expl-2* *ne-sp-ji-unit-up*)

lemma *d-dist-oU*:

Dom ($\bigcup X$) = $\bigcup (\text{Dom} ' X)$
apply (*clar simp* *simp*: *mr-simp*)
by *blast*

```

lemma d-dist-iU-iI:
  Dom (UU X|I) = Dom (INN X|I)
  by (clar simp simp: mr-simp)

```

```

lemma d-dist-iU-oI:
  assumes I ≠ {}
  shows Dom (UU X|I) = INN (Dom ' X ' I)
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  apply blast
  apply (clar simp simp: mr-simp)
  by (meson all-not-in-conv assms)

```

8.3 Left residual

```

definition sp-lres :: ('a,'c) mrel ⇒ ('b,'c) mrel ⇒ ('a,'b) mrel (infixl ⟨∅⟩ 65)
where
  Q ∅ R ≡ { (a,B) . ∀f . (∀ b ∈ B . (b,f b) ∈ R) → (a, ∪ { f b | b . b ∈ B }) ∈
  Q }

```

```
declare sp-lres-def [mr-simp]
```

```

lemma sp-lres-galois:
  S ∗ R ⊆ Q ↔ S ⊆ Q ∅ R
proof
  assume 1: S ∗ R ⊆ Q
  show S ⊆ Q ∅ R
  proof
    fix x
    assume x ∈ S
    from this obtain a B where 2: x = (a,B) ∧ (a,B) ∈ S
    by (metis surj-pair)
    have ∀f . (∀ b ∈ B . (b,f b) ∈ R) → (a, ∪ { f b | b . b ∈ B }) ∈ Q
    proof (rule allI, rule impI)
      fix f
      assume ∀ b ∈ B . (b,f b) ∈ R
      hence (a, ∪ { f b | b . b ∈ B }) ∈ S ∗ R
      apply (unfold s-prod-def)
      using 2 by auto
      thus (a, ∪ { f b | b . b ∈ B }) ∈ Q
      using 1 by auto
    qed
    thus x ∈ Q ∅ R
    using 2 sp-lres-def by auto
  qed
next
  assume 3: S ⊆ Q ∅ R
  have (Q ∅ R) ∗ R ⊆ Q

```

```

proof
  fix  $x$ 
  assume  $x \in (Q \oslash R) * R$ 
  from this obtain  $a D$  where  $4: x = (a,D) \wedge (a,D) \in (Q \oslash R) * R$ 
    by (metis surj-pair)
  from this obtain  $C$  where  $(a,C) \in Q \oslash R \wedge (\exists g . (\forall c \in C . (c,g) \in R) \wedge$ 
 $D = \bigcup \{ g c \mid c \in C \})$ 
    by (simp add: mr-simp) blast
  thus  $x \in Q$ 
    apply (unfold sp-lres-def)
    using 4 by auto
  qed
  thus  $S * R \subseteq Q$ 
    using 3 by (meson dual-order.trans s-prod-isol)
  qed

lemma sp-lres-expl:

$$Q \oslash R = \bigcup \{ S . S * R \subseteq Q \}$$

using sp-lres-galois by blast

lemma bot-sp-lres-d:

$$\{\} \oslash R = \{\} \oslash \text{Dom } R$$

by (metis d-sp-strict order-refl sp-lres-galois subset-antisym subset-empty)

lemma bot-sp-lres-expl:

$$\{\} \oslash R = -(U * \text{Dom } R)$$

apply (rule antisym)
apply (metis d-sp-strict d-test disjoint-eq-subset-Compl order-refl sp-lres-galois
sp-test subset-empty)
by (metis Compl-disjoint2 bot-sp-lres-d d-test sp-lres-galois sp-test subset-empty)

lemma sp-lres-sp-below:

$$(Q \oslash R) * R \subseteq Q$$

by (simp add: sp-lres-galois)

lemma sp-lres-left-isotone:

$$Q \subseteq S \implies Q \oslash R \subseteq S \oslash R$$

by (meson dual-order.refl sp-lres-galois subset-trans)

lemma sp-lres-right-antitone:

$$S \subseteq R \implies Q \oslash R \subseteq Q \oslash S$$

by (meson dual-order.trans s-prod-isor sp-lres-galois sp-lres-sp-below)

lemma sp-lres-down-closed-1:

$$Q \downarrow \oslash R = Q \downarrow \oslash R \downarrow$$

proof
  show  $Q \downarrow \oslash R \subseteq Q \downarrow \oslash R \downarrow$ 
    by (metis down-dist-sp down-idempotent down-isotone sp-lres-galois
sp-lres-sp-below)

```

```

next
show  $Q \downarrow \oslash R \downarrow \subseteq Q \downarrow \oslash R$ 
by (simp add: lower-reflexive sp-lres-right-antitone)
qed

lemma sp-lres-down-closed-2:
assumes  $R \downarrow = R$ 
and total  $T$ 
shows  $(R \oslash T) \downarrow = R \oslash T$ 
proof -
have  $(R \oslash T) \downarrow \subseteq R \oslash T$ 
by (metis assms lower-transitive sp-lres-galois sp-lres-sp-below
total-down-sp-semi-commute)
thus ?thesis
by (simp add: lower-reflexive subset-antisym)
qed

lemma down-sp-sp:
 $R \downarrow * S = R * (1_{\cup\cup} \cup S)$ 
proof -
have  $R \downarrow * S = R * (1_{\cup\cup} \cup 1) * S$ 
by (simp add: down-sp)
also have ... =  $R * ((1_{\cup\cup} \cup 1) * S)$ 
by (simp add: test-iu-test-sp-assoc-3)
also have ... =  $R * (1_{\cup\cup} \cup S)$ 
apply (clar simp simp: mr-simp)
apply safe
apply (smt (z3) Sup-empty ccpo-Sup-singleton image-empty image-insert
singletonI)
by (smt (verit, del-insts) Sup-empty all-not-in-conv ccpo-Sup-singleton
image-insert image-is-empty singletonD)
finally show ?thesis
.

qed

lemma iu-unit-sp-lres-iu-unit-ou:
 $U * aDom (ne R) = 1_{\cup\cup} \oslash (1_{\cup\cup} \cup R)$ 
apply (rule antisym)
apply (metis down-sp-sp sp-lres-galois down-sp-below-iu-unit order-refl)
by (metis down-sp-sp sp-lres-galois down-sp-below-iu-unit order-refl)

lemma bot-sl-below-complement-d:
 $\{\} \oslash R \subseteq - Dom R$ 
by (metis Compl-anti-mono bot-sp-lres-expl d-test dual-order.refl inf.order-iff
sp-test test-s-prod-is-meet)

lemma sp-unit-oi-bot-sp-lres:
 $1 \cap - Dom R = 1 \cap (\{\} \oslash R)$ 
by (smt (verit, ccfv-SIG) ad-sp-bot boolean-algebra-cancel.inf1

```

bot-sl-below-complement-d inf.orderE inf-bot-right inf-commute inf-le2 sp-lres-galois)

lemma *ad-explicit-d*:

*aDom R = -(U * Dom R) ∩ 1*

by (*simp add: bot-sp-lres-expl lattice-class.inf-sup-aci(1) sp-unit-oi-bot-sp-lres*)

lemma *top-test-sp-lres-total-expl-1*:

assumes *test p*

shows $\forall S . S \downarrow \subseteq (U * p) \oslash R \longleftrightarrow S \subseteq U * aDom (R \cap -(U * p))$

proof

fix *S::('b,'c) mrel*

have $S \subseteq U * aDom (R \cap -(U * p)) \longleftrightarrow ne(S \downarrow) * Dom (R \cap -(U * p)) = \{\}$

by (*metis (no-types, lifting) d-complement-ad inf-le2 subset-empty top-sp-test-down-iff-1 top-sp-test-down-iff-2 top-sp-test-down-iff-3*)

also have $\dots \longleftrightarrow ne(S \downarrow) * Dom (ne(R \downarrow) * \lambda p) = \{\}$

by (*simp add: assms d-ne-down-dp-complement-test*)

also have $\dots \longleftrightarrow ne(S \downarrow) * (ne(R \downarrow) * \lambda p) = \{\}$

using *d-sp-strict* **by** *auto*

also have $\dots \longleftrightarrow ne(S \downarrow) * ne(R \downarrow) * \lambda p = \{\}$

by (*simp add: test-assoc3*)

also have $\dots \longleftrightarrow ne((S \downarrow * R) \downarrow) * \lambda p = \{\}$

by (*simp add: down-dist-sp ne-dist-down-sp*)

also have $\dots \longleftrightarrow S \downarrow * R \subseteq U * p$

by (*metis assms top-sp-test-down-iff-1 top-sp-test-down-iff-2*)

top-sp-test-down-iff-3 subset-empty)

also have $\dots \longleftrightarrow S \downarrow \subseteq (U * p) \oslash R$

by (*simp add: sp-lres-galois*)

finally show $S \downarrow \subseteq (U * p) \oslash R \longleftrightarrow S \subseteq U * aDom (R \cap -(U * p))$

by *simp*

qed

lemma *top-test-sp-lres-total-expl-2*:

assumes *test p*

and *total T*

shows $(U * p) \oslash T = U * aDom (T \cap -(U * p))$

proof –

have $\forall S . S \subseteq (U * p) \oslash T \longleftrightarrow S \subseteq U * aDom (T \cap -(U * p))$

by (*smt assms lower-reflexive sp-lres-down-closed-2 subset-trans*)

top-sp-test-down-closed top-test-sp-lres-total-expl-1)

thus *?thesis*

by *blast*

qed

lemma *top-test-sp-lres-total-expl-3*:

assumes *test p*

shows $((U * p) \oslash R) \cap 1 = aDom (R \cap -(U * p))$

proof

```

have (((U * p) ∘ R) ∩ 1)↓ * R = (((U * p) ∘ R) ∩ 1) * (1UU ∪ R)
  using down-sp-sp by blast
also have ... = (((U * p) ∘ R) ∩ 1) * 1UU ∪ (((U * p) ∘ R) ∩ 1) * R
  by (simp add: s-distl-test)
also have ... ⊆ 1UU ∪ (((U * p) ∘ R) ∩ 1) * R
  using c6 by auto
also have ... ⊆ 1UU ∪ ((U * p) ∘ R) * R
  by (metis Un-Int-eq(4) inf-le2 iu-unit-convex iu-unit-down sp-oi-subdist
sup.mono)
also have ... ⊆ 1UU ∪ U * p
  using sp-lres-sp-below by auto
also have ... = U * p
  by (simp add: iu-unit-below-top-sp-test sup.absorb2)
finally have (((U * p) ∘ R) ∩ 1)↓ ⊆ (U * p) ∘ R
  by (simp add: sp-lres-galois)
hence ((U * p) ∘ R) ∩ 1 ⊆ U * aDom (R ∩ -(U * p))
  by (metis assms top-test-sp-lres-total-expl-1)
thus ((U * p) ∘ R) ∩ 1 ⊆ aDom (R ∩ -(U * p))
  by (metis (no-types, lifting) inf.idem inf.orderE inf-commute sp-oi-subdist
sp-test test-s-prod-is-meet)
next
have aDom (R ∩ -(U * p)) = U * aDom (R ∩ -(U * p)) ∩ 1
  by (metis (no-types, lifting) inf.absorb-iff2 inf.idem inf-commute inf-le2
sp-test test-s-prod-is-meet)
thus aDom (R ∩ -(U * p)) ⊆ ((U * p) ∘ R) ∩ 1
  by (metis Int-mono ad-test assms order-refl top-sp-test-down-closed
top-test-sp-lres-total-expl-1)
qed

lemma top-test-sp-lres-total-expl-4:
assumes test p
shows aDom (ne (R↓) * l p) = ((U * p) ∘ R) ∩ 1
by (simp add: assms d-ne-down-dp-complement-test top-test-sp-lres-total-expl-3)

lemma oi-complement-top-sp-test-top-1:
assumes test p
shows (R ∩ -(U * p)) * U = (R↓ ∩ -(U * p)) * U
proof (rule antisym)
show (R ∩ -(U * p)) * U ⊆ (R↓ ∩ -(U * p)) * U
  by (metis (no-types, lifting) assms cl8-var d-down-oi-up-1 equalityD2
ne-oi-complement-top-sp-test-1 ne-sp-iu-unit-up oc-top-sp-test-up-closed)
next
have R↓ ∩ -(U * p) ⊆ (R ∩ -(U * p))↓
  by (metis oc-top-sp-test-up-closed down-oi-up-closed assms)
also have ... ⊆ (R ∩ -(U * p)) * U
  by (simp add: down-below-sp-top)
finally show (R↓ ∩ -(U * p)) * U ⊆ (R ∩ -(U * p)) * U
  by (metis assms domain-up-down-conjugate inf-commute
ne-oi-complement-top-sp-test-1 ne-sp-iu-unit-up oc-top-sp-test-up-closed)

```

```

set-eq-subset)
qed

lemma oi-complement-top-sp-test-top-2:
assumes test p
shows  $(R \downarrow \cap -(U * p)) * U = ne(R \downarrow) * \lhd p * U$ 
proof (rule antisym)
have  $R \downarrow \cap -(U * p) * U \subseteq Dom(R \downarrow \cap -(U * p)) * U$ 
using sp-top-d by blast
also have ... =  $Dom(ne(R \downarrow) * \lhd p) * U$ 
by (simp add: assms d-ne-down-dp-complement-test)
also have ... =  $Dom(ne(R \downarrow * \lhd p)) * U$ 
by (simp add: ne-sp-test)
also have ... =  $ne(R \downarrow * \lhd p) * U$ 
by (simp add: cl8-var ne-sp-iu-unit-up)
also have ... =  $ne(R \downarrow) * \lhd p * U$ 
by (simp add: ne-sp-test)
finally show  $(R \downarrow \cap -(U * p)) * U \subseteq ne(R \downarrow) * \lhd p * U$ 
.

next
have  $ne(R \downarrow) * \lhd p \subseteq -(U * p)$ 
by (metis assms disjoint-eq-subset-Compl double-complement
inf-compl-bot-right schroeder-test sp-test test-complement-closed
top-test-oi-top-complement)
thus  $ne(R \downarrow) * \lhd p * U \subseteq (R \downarrow \cap -(U * p)) * U$ 
by (metis (no-types) assms cl8-var d-down-oi-up-1
d-ne-down-dp-complement-test ne-oi-complement-top-sp-test-1 ne-sp-iu-unit-up
oc-top-sp-test-up-closed sp-top-d)
qed

lemma oi-complement-top-sp-test-top-3:
assumes test p
shows  $(R \downarrow \cap -(U * p)) * U = ne(R \downarrow) * -(p * U)$ 
by (simp add: assms complement-test-sp-top oi-complement-top-sp-test-top-2
test-assoc1)

lemma split-sp-test-2:
test p  $\implies R \subseteq R * p \cup ne(R \downarrow) * (\lhd p)^\uparrow$ 
proof -
assume test p
hence  $R \subseteq R * p \cup (ne(R \downarrow * \lhd p))^\uparrow$ 
by (smt (verit, best) IntE UnCI UnE split-sp-test subsetI)
thus  $R \subseteq R * p \cup ne(R \downarrow) * (\lhd p)^\uparrow$ 
by (simp add: ne-sp-test-up)
qed

lemma split-sp-test-3:
test p  $\implies R \subseteq R * p \cup R \downarrow * (\lhd p)^\uparrow$ 
by (smt IntE UnCI UnE ne-dist-down-sp ne-sp-test-up ne-test-up split-sp-test

```

subsetI)

```
lemma split-sp-test-4:
  assumes test p
    and test q
  shows R * (p ∪ q) ⊆ R * p ∪ ne (R↓) * q↑
proof -
  have 1: (p ∪ q) * p = p
    by (metis Un-Int-eq(1) assms sp-test-dist-oi-left test-ou-closed
      test-sp-commute test-sp-idempotent)
  have (R * (p ∪ q))↓ * l p ⊆ R↓ * q
  proof -
    have (R * (p ∪ q))↓ * l p = R * (p ∪ q) * (1_{UU} ∪ l p)
      using down-sp-sp by blast
    also have ... = R * ((p ∪ q) * 1_{UU} ∪ (p ∪ q) * l p)
      by (smt (verit) assms inf.orderE s-distl-test test-assoc1 test-ou-closed)
    also have ... ⊆ R * (1_{UU} ∪ (p ∪ q) * l p)
      by (meson c6 s-prod-isor subset-refl sup.mono)
    also have ... = R↓ * ((p ∪ q) * l p)
      using down-sp-sp by blast
    also have ... = R↓ * (p * l p ∪ q * l p)
      by (metis assms ii-right-dist-ou inf-commute inf-le1 test-ou-closed
        test-sp-is-ii)
    also have ... = R↓ * (q * l p)
      by (metis Compl-disjoint assms(1) inf-commute inf-le1 s-prod-idl
        sp-test-dist-oi-left subset-Un-eq test-sp-commute)
    also have ... ⊆ R↓ * q
      by (metis inf-le1 inf-le2 s-prod-isor sp-test)
    finally show ?thesis
  .
qed
hence 2: ne ((R * (p ∪ q))↓) * l p ⊆ ne (R↓) * q
  by (metis assms(2) inf-le2 ne-dist-ou ne-sp-test subset-Un-eq)
have R * (p ∪ q) ⊆ R * (p ∪ q) * p ∪ ne ((R * (p ∪ q))↓) * (l p)↑
  by (simp add: assms split-sp-test-2)
also have ... = R * p ∪ ne ((R * (p ∪ q))↓) * (l p)↑
  using 1 by (metis assms(1) inf.orderE test-assoc3)
also have ... ⊆ R * p ∪ ne (R↓) * q↑
  using 2 by (metis (no-types, lifting) Un-Int-eq(1) assms(2) inf-le2 iu-isotone
    ne-sp-test ne-sp-test-up sup.mono)
finally show ?thesis
.
qed

lemma split-sp-test-5:
  assumes test p
    and test q
  shows R * (p ∪ q) ⊆ R * p ∪ R↓ * q↑
proof -
```

```

have  $R * (p \cup q) \subseteq R * p \cup ne(R\downarrow) * q\uparrow$ 
  by (simp add: assms split-sp-test-4)
thus ?thesis
  by (metis (no-types, lifting) assms(2) le-inf-iff ne-dist-down-sp ne-test-up
sup-neg-inf)
qed

lemma split-sp-test-6:
assumes test p
and test q
shows Dom (R * (p ∪ q)) ⊆ Dom (R * p ∪ ne(R↓) * q)
proof -
  have Dom (R * (p ∪ q)) ⊆ Dom (R * p ∪ ne(R↓) * q↑)
    by (simp add: assms d-isotone split-sp-test-4)
  also have ... = Dom (R * p ∪ (ne(R↓) * q)↑)
    by (simp add: assms(2) ne-sp-test ne-sp-test-up)
  also have ... ⊆ Dom (R * p ∪ ne(R↓) * q)
    by (metis d-up subsetI up-dist-ou up-idempotent)
  finally show ?thesis
.

qed

lemma split-sp-test-7:
assumes test p
and test q
shows Dom (ne(R↓) * (p ∪ q)) = Dom (ne(R↓) * p ∪ ne(R↓) * q)
apply (rule antisym)
apply (metis assms ne-down-idempotent split-sp-test-6)
by (smt (verit, ccfv-SIG) Un-Int-eq(1) Un-subset-iff assms(2) d-isotone
inf.orderE inf-le1 le-inf-iff lower-eq-down ne-dist-ou sp-oi-subdist-2 subset-Un-eq
sup.mono)

lemma test-sp-left-dist-iu-1:
test p ==> p * (R ∪ S) = p * R ∪ S
by (metis cl8-var inf.orderE p-prod-assoc s-subid-iff2)

lemma test-sp-left-dist-iu-2:
test p ==> p * (R ∪ S) = R ∪ p * S
by (metis iu-commute test-sp-left-dist-iu-1)

lemma d-sp-below-iu-down:
Dom R * S ⊆ (R ∪ S)↓
by (simp add: cl8-var iu-lower-left-isotone sp-ji-unit-lower)

lemma d-sp-ne-down-below-ne-iu-down:
Dom R * ne(S↓) ⊆ ne((R ∪ S)↓)
proof -
  have Dom R * S↓ ⊆ (R ∪ S)↓
    by (simp add: cl8-var iu-lower-isotone sp-ji-unit-lower)

```

```

hence ne (Dom R * S↓) ⊆ ne ((R ∪∪ S)↓)
  by blast
thus ?thesis
  by (smt d-test test-sp-ne)
qed

lemma top-test:
  test p ==> U * p = { (a,B) . (∀ b ∈ B . (b,{b}) ∈ p) }
  apply (unfold test)
  apply (clar simp simp: mr-simp)
  by fastforce

lemma iu-oi-complement-top-test-ou-up:
  test p ==> (R ∪∪ S) ∩ -(U * p) ⊆ ((R ∪ S) ∩ -(U * p))↑
  apply (unfold top-test)
  apply (clar simp simp: mr-simp)
  by blast

lemma d-ne-iu-down-sp-test-ou:
  assumes test p
  shows Dom (ne ((R ∪∪ S)↓) * p) ⊆ Dom ((ne (R↓) ∪ ne (S↓)) * p)
  proof –
    have Dom (ne ((R ∪∪ S)↓) * p) = Dom ((R ∪∪ S) ∩ -(U * l p))
      by (metis assms d-ne-down-dp-complement-test test-double-complement)
    also have ... ⊆ Dom ((R ∪ S) ∩ -(U * l p))
      by (metis iu-oi-complement-top-test-ou-up d-isotone d-up
semilattice-inf-class.inf-le2)
    also have ... = Dom (ne ((R ∪ S)↓) * p)
      by (metis assms d-ne-down-dp-complement-test test-double-complement)
    finally show ?thesis
      by (simp add: down-dist-ou ne-dist-ou)
  qed

lemma test-sp-left-dist-iU:
  assumes test p
  and I ≠ {}
  shows p * (⋃ X|I) = ⋃ X(λi . p * X i)|I
  apply (rule antisym)
  apply (clar simp simp: mr-simp)
  subgoal for a B f proof –
    assume 1: (a,B) ∈ p
    hence 2: B = {a}
      by (metis assms(1) test)
    assume ∀ b ∈ B . ∃ g . f b = ⋃(g ` I) ∧ (∀ i ∈ I . (b,g i) ∈ X i)
    from this obtain g where 3: f a = ⋃(g ` I) ∧ (∀ i ∈ I . (a,g i) ∈ X i)
      using 2 by auto
    have ⋃(f ` B) = ⋃(g ` I) ∧ (∀ i ∈ I . ∃ B . (a,B) ∈ p ∧ (∃ f . (∀ b ∈ B . (b,f b)
    ∈ X i) ∧ g i = ⋃(f ` B)))
      apply (rule conjI)

```

```

using 2 3 apply blast
apply (rule ballI)
apply (rule exI[of - B])
apply (rule conjI)
using 1 apply simp
subgoal for i
  apply (rule exI[of - λb . g i])
  using 2 3 by blast
done
thus ?thesis
  by auto
qed
apply (clarify simp: mr-simp)
subgoal for a f proof -
  assume 4: ∀ i∈I . ∃ B . (a,B) ∈ p ∧ (∃ g . (∀ b∈B . (b,g b) ∈ X i) ∧ f i = ∪(g ` B))
  have (a,{a}) ∈ p ∧ (∃ g . (∀ b∈{a} . ∃ f . g b = ∪(f ` I) ∧ (∀ i∈I . (b,f i) ∈ X i)) ∧ ∪(f ` I) = ∪(g ` {a}))
    apply (rule conjI)
    using 4 apply (metis assms equals0I test)
    apply (rule exI[of - λa . ∪(f ` I)])
    apply clarify
    apply (rule exI[of - f])
    using 4 assms(1) test by fastforce
  thus ?thesis
    by auto
qed
done

```

8.4 Modal operations

```

definition adia :: ('a,'b) mrel ⇒ ('b,'b) mrel ⇒ ('a,'a) mrel (⟨| - ⟩ → [50,90] 95)
where
|R⟩p ≡ { (a,{a}) | a . ∃ B . (a,B) ∈ R ∧ (∀ b∈B . (b,{b}) ∈ p) }

```

```

definition abox :: ('a,'b) mrel ⇒ ('b,'b) mrel ⇒ ('a,'a) mrel (⟨| - ] → [50,90] 95)
where
|R]p ≡ { (a,{a}) | a . ∀ B . (a,B) ∈ R → (∀ b∈B . (b,{b}) ∈ p) }

```

```

definition edia :: ('a,'b) mrel ⇒ ('b,'b) mrel ⇒ ('a,'a) mrel (⟨| - ⟧ → [50,90]
95) where
|R⟩⟩p ≡ { (a,{a}) | a . ∃ B . (a,B) ∈ R ∧ (∃ b∈B . (b,{b}) ∈ p) }

```

```

definition ebox :: ('a,'b) mrel ⇒ ('b,'b) mrel ⇒ ('a,'a) mrel (⟨| - ]] → [50,90] 95)
where
|R]]p ≡ { (a,{a}) | a . ∀ B . (a,B) ∈ R → (∃ b∈B . (b,{b}) ∈ p) }

```

```

declare adia-def [mr-simp] abox-def [mr-simp] edia-def [mr-simp] ebox-def
[mr-simp]

```

```

lemma adia:
  assumes test p
  shows |R>p = Dom (R * p)
proof
  show |R>p ⊆ Dom (R * p)
  proof
    fix x
    assume x ∈ |R>p
    from this obtain a B where 1: x = (a,{a}) ∧ (a,B) ∈ R ∧ (∀ b∈B . (b,{b}) ∈ p)
      by (smt adia-def surj-pair mem-Collect-eq)
    have (a,B) ∈ R * p
    apply (clar simp simp: s-prod-def)
    apply (rule exI[where ?x=B])
    apply (rule conjI)
    using 1 apply simp
    apply (rule exI[where ?x=λx . {x}])
    using 1 by auto
    thus x ∈ Dom (R * p)
      using 1 Dom-def by auto
  qed
next
show Dom (R * p) ⊆ |R>p
proof
  fix x
  assume x ∈ Dom (R * p)
  from this obtain a A where 2: x = (a,{a}) ∧ (a,A) ∈ R * p
    by (smt Dom-def surj-pair mem-Collect-eq)
  from this obtain B f where 3: (a,B) ∈ R ∧ (∀ b∈B . (b,f b) ∈ p) ∧ A = ∪ {f b | b . b ∈ B }
    by (simp add: mr-simp) blast
  hence ∀ b∈B . (b,{b}) ∈ p
    using assms subid-aux2 by fastforce
  thus x ∈ |R>p
    using 2 3 adia-def by blast
  qed
qed

lemma abox-1:
  assumes test p
  shows |R]p = aDom (R ∩ -(U * p))
proof
  show |R]p ⊆ aDom (R ∩ -(U * p))
  proof
    fix x
    assume x ∈ |R]p
    from this obtain a where 1: x = (a,{a}) ∧ (∀ B . (a,B) ∈ R → (∀ b∈B . (b,{b}) ∈ p))

```

```

by (smt abox-def surj-pair mem-Collect-eq)
have  $\neg(\exists B . (a,B) \in R \cap -(U * p))$ 
proof
  assume  $\exists B . (a,B) \in R \cap -(U * p)$ 
  from this obtain B where  $(a,B) \in R \wedge (a,B) \notin U * p$ 
    by auto
  thus False
    using 1 by (metis (no-types, lifting) assms top-sp-test)
qed
thus  $x \in aDom (R \cap -(U * p))$ 
  using 1 aDom by blast
qed
next
show  $aDom (R \cap -(U * p)) \subseteq |R|p$ 
proof
  fix x
  assume  $x \in aDom (R \cap -(U * p))$ 
  from this obtain a where 2:  $x = (a, \{a\}) \wedge \neg(\exists B . (a,B) \in R \cap -(U * p))$ 
    by (smt aDom surj-pair mem-Collect-eq)
  hence  $\forall B . (a,B) \in R \longrightarrow (\forall b \in B . (b, \{b\}) \in p)$ 
    using assms by (metis (no-types, lifting) IntI oc-top-sp-test)
  thus  $x \in |R|p$ 
    using 2 abox-def by blast
qed
qed

lemma abox:
assumes test p
shows  $|R|p = aDom (ne (R\downarrow) * \lambda p)$ 
by (simp add: abox-1 assms d-ne-down-dp-complement-test)

lemma edia-1:
assumes test p
shows  $|R\rangle\rangle p = Dom (R \cap -(U * \lambda p))$ 
proof
show  $|R\rangle\rangle p \subseteq Dom (R \cap -(U * \lambda p))$ 
proof
  fix x
  assume  $x \in |R\rangle\rangle p$ 
  from this obtain a b B where 1:  $x = (a, \{a\}) \wedge (a,B) \in R \wedge b \in B \wedge (b, \{b\}) \in p$ 
    by (smt edia-def surj-pair mem-Collect-eq)
  hence  $(a,B) \in -(U * \lambda p)$ 
    by (metis (no-types, lifting) lattice-class.inf-sup-ord(2) oc-top-sp-test test-complement)
  thus  $x \in Dom (R \cap -(U * \lambda p))$ 
    using 1 Dom-def by auto
qed
next

```

```

show Dom (R ∩ −(U * ↴ p)) ⊆ |R⟩⟨p
proof
  fix x
  assume x ∈ Dom (R ∩ −(U * ↴ p))
  from this obtain a B where 2: x = (a,{a}) ∧ (a,B) ∈ R ∧ (a,B) ∈ −(U * ↴ p)
    by (smt Dom-def surj-pair mem-Collect-eq IntE)
  hence ∃ b∈B . (b,{b}) ∈ p
    by (meson oc-top-sp-test test-complement test-complement-closed)
  thus x ∈ |R⟩⟨p
    using 2 edia-def by blast
  qed
qed

lemma edia:
assumes test p
shows |R⟩⟨p = Dom (ne (R↓) * p)
by (metis assms d-ne-down-dp-complement-test edia-1 test-double-complement)

lemma ebox:
assumes test p
shows |R]⟩p = aDom (R * ↴ p)
proof
  show |R]⟩p ⊆ aDom (R * ↴ p)
  proof
    fix x
    assume x ∈ |R]⟩p
    from this obtain a where 1: x = (a,{a}) ∧ (∀ B . (a,B) ∈ R → (∃ b∈B . (b,{b}) ∈ p))
      by (smt ebox-def surj-pair mem-Collect-eq)
    hence ¬(∃ B . (a,B) ∈ R * ↴ p)
      by (metis (no-types, lifting) s-prod-test test-complement)
    thus x ∈ aDom (R * ↴ p)
      using 1 aDom by blast
  qed
next
  show aDom (R * ↴ p) ⊆ |R]⟩p
  proof
    fix x
    assume x ∈ aDom (R * ↴ p)
    from this obtain a where 2: x = (a,{a}) ∧ ¬(∃ B . (a,B) ∈ R * ↴ p)
      by (smt aDom surj-pair mem-Collect-eq)
    have ∀ B . (a,B) ∈ R → (∃ b∈B . (b,{b}) ∈ p)
    proof (rule allI, rule impI)
      fix B
      assume (a,B) ∈ R
      hence (a,B) ∈ U * ↴ p
        using 2 by (metis Int-iff Int-lower2 sp-test)
      thus ∃ b∈B . (b,{b}) ∈ p

```

```

    by (meson test-complement test-complement-closed top-sp-test)
qed
thus  $x \in |R|]p$ 
      using 2 ebox-def by blast
qed
qed

lemma abox-2:
assumes test p
shows  $|R|p = -((R \cap -(U * p)) * U) \cap 1$ 
by (simp add: abox-1 assms ne-a-expl ne-oi-complement-top-sp-test-1)

lemma abox-3:
assumes test p
shows  $|R|p = -(ne(R\downarrow) * \lambda p * U) \cap 1$ 
by (simp add: abox assms ne-a-expl ne-sp-test)

lemma abox-4:
assumes test p
shows  $|R|p = ((U * p) \oslash R) \cap 1$ 
by (simp add: abox-1 assms top-test-sp-lres-total-expl-3)

lemma abox-ebox:
assumes test p
shows  $|R|p = |ne(R\downarrow)]]p$ 
by (simp add: abox assms ebox)

lemma abox-edia:
assumes test p
shows  $|R|p = \lambda |R\rangle(\lambda p)$ 
by (simp add: abox assms edia)

lemma abox-adia:
assumes test p
shows  $|R|p = \lambda |ne(R\downarrow)\rangle(\lambda p)$ 
by (simp add: abox adia assms)

lemma edia-adia:
assumes test p
shows  $|R\rangle\rangle p = |ne(R\downarrow)\rangle p$ 
by (simp add: adia assms edia)

lemma edia-abox:
assumes test p
shows  $|R\rangle\rangle p = \lambda |R](\lambda p)$ 
by (metis abox-1 assms d-complement-ad edia-1
semilattice-inf-class.inf.cobounded2)

lemma edia-ebox:

```

```

assumes test p
shows |R⟩p = ↵[ne(R↓)](p)
by (simp add: abox assms ebox edia-abox)

lemma abox-ne-down:
assumes test p
shows |R]p = |ne(R↓)]p
by (simp add: abox assms ne-down-idempotent)

lemma edia-ne-down:
assumes test p
shows |R⟩p = |ne(R↓)⟩p
by (simp add: assms edia ne-down-idempotent)

lemma adia-up:
assumes test p
shows |R⟩p = |R↑⟩p
proof –
  have |R↑⟩p = Dom (R↑ ∩ U * p)
  by (metis adia assms iu-assoc iu-unit-up up-dist-iu-oi)
  also have ... = Dom (R ∩ U * p)
  by (metis assms d-def-expl domain-up-down-conjugate sp-test-dist-oi-right
    top-sp-test-down-closed)
  also have ... = |R⟩p
  by (metis adia assms inf.absorb-iff2 inf-commute top-down top-lower-greatest)
  finally show ?thesis
  by simp
qed

lemma ebox-up:
assumes test p
shows |R]]p = |R↑]]p
by (metis Int-commute adia adia-up assms ebox semilattice-inf-class.inf-le1)

lemma adia-ebox:
assumes test p
shows |R⟩p = ↵[R]](p)
by (metis (no-types, lifting) adia assms d-complement-ad ebox
    test-double-complement)

lemma ebox-adia:
assumes test p
shows |R]]p = ↵[R](p)
by (simp add: adia assms ebox)

lemma abox-down:
assumes test p
shows |R]p = |R↓]p
by (simp add: abox assms)

```

```

lemma edia-down:
  assumes test p
  shows  $|R\rangle\langle p = |R\downarrow\rangle\langle p$ 
  by (simp add: assms edia)

lemma fusion-oi-complement-top-test-up:
  test p  $\implies$  fus R  $\cap$   $-(U * p) \subseteq (R \cap -(U * p))^\uparrow$ 
  apply (unfold top-test)
  apply (clar simp simp: mr-simp)
  by blast

lemma adia-left-isotone:
  test p  $\implies$  R  $\subseteq$  S  $\implies$   $|R\rangle p \subseteq |S\rangle p$ 
  by (metis adia d-isotone inf.absorb-iff1 sp-test-dist-oi)

lemma adia-right-isotone:
  test p  $\implies$  test q  $\implies$  p  $\subseteq$  q  $\implies$   $|R\rangle p \subseteq |R\rangle q$ 
  by (metis (no-types, opaque-lifting) adia d-isotone inf.orderE inf-commute
inf-le1 sp-test test-assoc3 test-s-prod-is-meet)

lemma abox-left-antitone:
  test p  $\implies$  R  $\subseteq$  S  $\implies$   $|S]p \subseteq |R]p$ 
  apply (clar simp simp: mr-simp) by force

lemma abox-right-isotone:
  test p  $\implies$  test q  $\implies$  p  $\subseteq$  q  $\implies$   $|R]p \subseteq |R]q$ 
  by (smt (verit, ccfv-threshold) IntE abox-def inf.orderE mem-Collect-eq subsetI)

lemma edia-left-isotone:
  test p  $\implies$  R  $\subseteq$  S  $\implies$   $|R\rangle\langle p \subseteq |S\rangle\langle p$ 
  by (metis Int-mono adia-left-isotone down-isotone edia-adia order-refl)

lemma edia-right-isotone:
  test p  $\implies$  test q  $\implies$  p  $\subseteq$  q  $\implies$   $|R\rangle\langle p \subseteq |R\rangle\langle q$ 
  by (simp add: adia-right-isotone edia-adia)

lemma ebox-left-antitone:
  test p  $\implies$  R  $\subseteq$  S  $\implies$   $|S]]p \subseteq |R]]p$ 
  by (metis (no-types, lifting) adia-ebox adia-left-isotone ebox-adia
test-complement-antitone test-double-complement)

lemma ebox-right-isotone:
  test p  $\implies$  test q  $\implies$  p  $\subseteq$  q  $\implies$   $|R]]p \subseteq |R]]q$ 
  by (smt (verit, ccfv-SIG) adia-ebox adia-right-isotone ebox inf-le2
test-complement-antitone test-double-complement)

lemma edia-fusion:
  assumes test p

```

```

shows  $|R\rangle\rangle p = |fus R\rangle\rangle p$ 
proof
  have  $|fus R\rangle\rangle p = Dom(fus R \cap -(U * \lambda p))$ 
    using assms edia-1 by blast
  also have ...  $\subseteq Dom(R \cap -(U * \lambda p))$ 
    by (metis fusion-oi-complement-top-test-up d-isotone d-up semilattice-inf-class.inf-le2)
  also have ...  $= |R\rangle\rangle p$ 
    using assms edia-1 by blast
  finally show  $|fus R\rangle\rangle p \subseteq |R\rangle\rangle p$ 
.

next
  have  $|R\rangle\rangle p \subseteq |(fus R)\downarrow\rangle\rangle p$ 
    by (simp add: assms edia-left-isotone fusion-lower-increasing)
  thus  $|R\rangle\rangle p \subseteq |fus R\rangle\rangle p$ 
    using assms edia-down by blast
qed

lemma abox-fusion:
  assumes test p
  shows  $|R]p = |fus R]p$ 
  by (metis Int-lower2 abox-edia assms edia-fusion)

lemma abox-fission:
  assumes test p
  shows  $|R]p = |fis R]p$ 
  by (metis assms abox-fusion fusion-fission)

lemma edia-fission:
  assumes test p
  shows  $|R\rangle\rangle p = |fis R\rangle\rangle p$ 
  by (metis assms edia-fusion fusion-fission)

lemma fission-below:
   $fis R \subseteq S \longleftrightarrow (\forall a b B . (a,B) \in R \wedge b \in B \longrightarrow (a,\{b\}) \in S)$ 
  apply standard
  apply (simp add: basic-trans-rules(31) fission-set)
  apply (clar simp: mr-simp)
  by blast

lemma below-fission-up:
   $S \subseteq (fis R)\uparrow \longleftrightarrow (\forall a B . (a,B) \in S \longrightarrow (\exists C . (a,C) \in R \wedge C \cap B \neq \{\}))$ 
proof
  assume  $S \subseteq (fis R)\uparrow$ 
  thus  $\forall a B . (a,B) \in S \longrightarrow (\exists C . (a,C) \in R \wedge C \cap B \neq \{\})$ 
    apply (clar simp: mr-simp)
    by fastforce
next
  assume 1:  $\forall a B . (a,B) \in S \longrightarrow (\exists C . (a,C) \in R \wedge C \cap B \neq \{\})$ 

```

```

show  $S \subseteq (\text{fis } R)^\uparrow$ 
proof
  fix  $x$ 
  assume  $x \in S$ 
  from this obtain  $a B$  where  $\exists: x = (a, B) \wedge (a, B) \in S$ 
    by (metis surj-pair)
  hence  $\exists C . (a, C) \in R \wedge C \cap B \neq \{\}$ 
    using 1 by simp
  from this obtain  $C b$  where  $\exists: (a, C) \in R \wedge b \in C \wedge b \in B$ 
    by auto
  hence  $(a, \{b\}) \in \text{fis } R$ 
    using fission-set by blast
  thus  $x \in (\text{fis } R)^\uparrow$ 
    using 2 3 U-par-st by fastforce
  qed
qed

```

```

lemma ebox-below-abox:
  assumes test  $p$ 
  and fis  $R \subseteq S$ 
  shows  $|S|p \subseteq |R|p$ 
  by (metis abox-ebox abox-fission assms ebox-left-antitone
    fission-down-ne-fixpoint)

lemma abox-below-ebox:
  assumes test  $p$ 
  and  $S \subseteq (\text{fis } R)^\uparrow$ 
  shows  $|R|p \subseteq |S|p$ 
  by (metis abox-ebox abox-fission assms ebox-left-antitone ebox-up
    fission-down-ne-fixpoint)

lemma abox-eq-ebox:
  assumes test  $p$ 
  and fis  $R \subseteq S$ 
  and  $S \subseteq (\text{fis } R)^\uparrow$ 
  shows  $|R|p = |S|p$ 
  by (simp add: abox-below-ebox assms ebox-below-abox subset-antisym)

lemma abox-eq-ebox-sufficient:
   $S = \text{fis } R \vee S = \text{ne } (R^\downarrow) \vee S = (\text{ne } (R^\downarrow))^\uparrow \longrightarrow \text{fis } R \subseteq S \wedge S \subseteq (\text{fis } R)^\uparrow$ 
  apply (unfold imp-disjL)
  apply (intro conjI)
  apply (simp add: convex-reflexive)
  apply (simp add: fission-inner-deterministic fission-up-ne-down-up
    oi-subset-upper-right-antitone same-fusion-fission-lower)
  by (metis convex-reflexive fission-up-ne-down-up order-refl)

lemma ebox-fission-abox:
  test  $p \implies |R|p = |\text{fis } R|p$ 

```

```

by (metis abox abox-fission ebox fission-down-ne-fixpoint)

lemma ebox-down-ne-up-abox:
  test p ==> |R]p = |(ne (R↓))↑]p
  using abox-ebox ebox-up by blast

lemma same-fusion:
  assumes fis R ⊑↓ S
  and S ⊑↓ fus R
  shows fis R = fis S
  by (metis assms fission-down fission-fusion-galois subset-antisym)

lemma same-abox:
  assumes fis R ⊑↓ S
  and S ⊑↓ fus R
  and test p
  shows |R]p = |S]p
  by (metis assms ebox-fission-abox same-fusion)

lemma abox-ebox-inner-deterministic:
  assumes test p
  and inner-deterministic R
  shows |R]p = |R]]p
  apply (rule abox-eq-ebox)
  apply (simp add: assms(1))
  using assms(2) fission-inner-deterministic-fixpoint apply blast
  by (metis assms(2) convex-reflexive fission-inner-deterministic-fixpoint)

lemma adia-edia-inner-deterministic:
  assumes test p
  and inner-deterministic R
  shows |R>p = |R>]]p
  by (metis assms edia-adia fission-down-ne-fixpoint
fission-inner-deterministic-fixpoint)

lemma abox-adia-deterministic:
  assumes test p
  and deterministic R
  shows |R]p = |R>p
proof
  show |R]p ⊆ |R>p
  proof
    fix x
    assume x ∈ |R]p
    from this obtain a where 1: x = (a,{a}) ∧ (∀ B . (a,B) ∈ R → (∀ b∈B .
(b,{b}) ∈ p))
    using abox-def by force
    from assms(2) obtain B where (a,B) ∈ R
    by (meson deterministic-set)

```

```

thus  $x \in |R\rangle p$ 
  using 1 adia-def by fastforce
qed
next
show  $|R\rangle p \subseteq |R]p$ 
proof
fix  $x$ 
assume  $x \in |R\rangle p$ 
from this obtain  $a B$  where  $\exists: x = (a,\{a\}) \wedge (a,B) \in R \wedge (\forall b \in B . (b,\{b\}) \in p)$ 
by (smt adia-def mem-Collect-eq)
have  $\forall C . (a,C) \in R \longrightarrow (\forall b \in C . (b,\{b\}) \in p)$ 
proof (rule allI, rule impI)
fix  $C$ 
assume  $(a,C) \in R$ 
hence  $B = C$ 
using 2 by (metis assms(2) deterministic-set)
thus  $\forall b \in C . (b,\{b\}) \in p$ 
  using 2 by simp
qed
thus  $x \in |R]p$ 
  using 2 abox-def by blast
qed
qed

lemma ebox-edia-deterministic:
assumes test  $p$ 
and deterministic  $R$ 
shows  $|R]]p = |R\rangle\rangle p$ 
by (simp add: assms abox-adia-deterministic ebox-adia edia-abox)

lemma abox-ebox-fusion:
assumes test  $p$ 
shows  $|fis R]p = |fis R]]p$ 
by (metis abox-fission assms ebox-fission-abox)

lemma abox-fission-edia-fusion:
assumes test  $p$ 
shows  $|fis R]p = |fus R\rangle p$ 
by (simp add: abox-adia-deterministic abox-fusion assms fusion-deterministic fusion-fission)

lemma abox-adia-fusion:
assumes test  $p$ 
shows  $|fus R]p = |fus R\rangle p$ 
by (simp add: abox-adia-deterministic assms fusion-deterministic)

```

8.5 Goldblatt's axioms without star

```

lemma abox-sp-unit:
  |R]1 = 1
  apply (clar simp simp: mr-simp) by force

lemma ou-unit-abox:
  test p  $\implies$  |{}|p = 1
  by (metis abox-1 abox-sp-unit disjoint-eq-subset-Compl empty-subsetI
    inf.absorb-iff2 test-complement-closed)

lemma ou-unit-test-implication:
  test p  $\implies$  {}  $\rightarrow$  p = 1
  by blast

lemma sp-unit-abox:
  test p  $\implies$  |1|p = p
  by (smt (verit) Int-left-commute abox-1 c1 cl8-var convex-reflexive
    d-ne-down-dp-complement-test fission-down-ne-fixpoint
    fission-inner-deterministic-fixpoint inf.absorb-iff2 inf-commute
    inner-deterministic-sp-unit s-subid-iff2 test-double-complement test-sp)

lemma sp-unit-test-implication:
  test p  $\implies$  1  $\rightarrow$  p = p
  by simp

lemma test-abox-ebox:
  test p  $\implies$  test q  $\implies$  |q|p = |q]|p
  apply (rule antisym)
  apply (metis abox-ebox-inner-deterministic dual-order.trans
    inner-deterministic-sp-unit subset-refl)
  by (metis abox-ebox-inner-deterministic dual-order.eq-iff
    inner-deterministic-sp-unit inner-univalent-down-closed ne-equality test-ne)

lemma test-abox:
  test p  $\implies$  test q  $\implies$  |q|p = q  $\rightarrow$  p
  by (smt Int-commute Int-lower2 abox cl9-var compl-sup d-complement-ad
    d-ne-down-dp-complement-test lattice-class.inf-sup-aci(2) sp-unit-abox
    test-ou-closed)

lemma abox-ou-adia-sp-unit:
  assumes test p
  shows |R]p  $\cup$  |R>1 = 1
  apply (rule antisym)
  apply (simp add: assms abox adia-ebox)
  by (clar simp simp: mr-simp)

lemma d-test-sp:
  test p  $\implies$  Dom (p * R) = p * Dom R
  by (simp add: c4 d-def-expl test-sp-left-dist-iu-1)

```

```

lemma ad-test-sp:
  test p ==> aDom (p * R) = l p ∪ aDom R
  by (metis (no-types, opaque-lifting) Int-commute boolean-algebra.conj-disj-distrib
    boolean-algebra.de-Morgan-conj d-s-id-inter d-test-sp s-subid-iff2 test-fix)

lemma adia-test-sp:
  test p ==> test q ==> |p * R]q = p * |R]q
  by (metis (no-types, lifting) adia d-test-sp test-assoc3 test-double-complement)

lemma ebox-test-sp:
  test p ==> test q ==> |p * R]]q = l p ∪ |R]]q
  by (simp add: ad-test-sp ebox test-assoc3)

lemma abox-test-sp:
  assumes test p
  and test q
  shows |p * R]q = l p ∪ |R]q
proof -
  have |p * R]q = aDom ((p * R) ∩ -(U * q))
  by (simp add: abox-1 assms(2))
  also have ... = aDom (p * (R ∩ -(U * q)))
  by (metis Int-assoc assms(1) test-sp)
  also have ... = l p ∪ |R]q
  by (simp add: abox-1 ad-test-sp assms)
  finally show ?thesis
  .
qed

lemma abox-test-sp-2:
  test p ==> test q ==> p ∪ |R]q = |l p * R]q
  by (simp add: abox-test-sp test-double-complement)

lemma abox-test-sp-3:
  test p ==> test q ==> p → |R]q = |p * R]q
  by (simp add: abox-test-sp)

lemma fission-sp-dist:
  fis (R * S) = fis (R * Dom S) * fis S
proof -
  have S = Dom S * (S ∪ aDom S * 1UU)
  by (auto simp: mr-simp)
  hence fis (R * S) = fis (R * Dom S * (S ∪ aDom S * 1UU))
  by (metis d-s-id-ax sp-test-sp-oi-right test-sp)
  also have ... = fis (R * Dom S) * fis (S ∪ aDom S * 1UU)
  apply (rule fission-sp-total-dist)
  by (smt (verit) total-dom Compl-disjoint ad-sp-bot ad-test-sp c6 compl-inf-bot
    d-complement-ad d-dist-ou inf-le2 iu-unit-down subset-Un-eq sup-ge2
    sup-inf-absorb total-lower)

```

```

also have ... = fis (R * Dom S) * (fis S ∪ fis (aDom S * 1UU))
  by (simp add: fission-dist-ou)
also have ... = fis (R * Dom S) * fis S
  by (simp add: fission-sp-iu-unit)
finally show ?thesis
.

qed

```

lemma abox-test:

```

test p ==> test ( |R]p )
  by (simp add: abox)

```

lemma adia-test:

```

test p ==> test ( |R⟩p )
  by (simp add: adia d-test)

```

lemma ebox-test:

```

test p ==> test ( |R]]p )
  by (simp add: ebox)

```

lemma edia-test:

```

test p ==> test ( |R⟩⟩p )
  by (simp add: edia d-test)

```

lemma abox-sp:

```

assumes test p
  and test q
shows |R](p * q) = |R]p * |R]q
proof -
  have |R](p * q) = aDom (ne (R↓) * (l p ∪ l q))
    by (metis (no-types, lifting) abox-1 ad-test-sp assms cl9-var
      d-ne-down-dp-complement-test sp-test test-double-complement test-oi-closed)
  also have ... = aDom (ne (R↓) * l p) * aDom (ne (R↓) * l q)
    by (smt ad-test-sp cl9-var d-complement-ad d-dist-ou d-test-sp
      semilattice-inf-class.inf-le2 split-sp-test-7)
  also have ... = |R]p * |R]q
    by (simp add: abox assms)
finally show ?thesis
.
```

qed

lemma adia-ou-below-ne-down:

```

assumes test p
shows |R⟩(p ∪ l q) ⊆ |R⟩p ∪ |ne (R↓)⟩(l q)
  by (metis adia assms d-dist-ou split-sp-test-6 test-complement-closed
    test-ou-closed)

```

lemma abox-adia-mp:

```

assumes test p

```

and test q
shows $|R\rangle(p \rightarrow q) * |R\rangle p \subseteq |R\rangle q$
by (smt adia-ou-below-ne-down test-shunting abox adia assms d-complement-ad
 sup-commute test-complement-closed test-implication-closed)

lemma adia-abox-mp:
assumes test p
and test q
shows $|R\rangle p * |R\rangle(p \rightarrow q) \subseteq |R\rangle q$
proof –
have $p \subseteq p \rightarrow q \rightarrow q$
using assms(1) **by** blast
hence $|R\rangle p \subseteq |R\rangle((p \rightarrow q) \rightarrow q)$
by (simp add: adia-right-isotone assms)
thus ?thesis
by (smt abox-adia-mp abox-test adia-test assms(2)
 semilattice-inf-class.inf.orderE semilattice-inf-class.le-infI2 test-implication-closed
 test-shunting)
qed

lemma abox-implication-adia:
assumes test p
and test q
shows $|R\rangle(p \rightarrow q) \subseteq |R\rangle p \rightarrow |R\rangle q$
by (metis adia-abox-mp test-shunting test-sp-commute Int-lower2 Un-commute
 abox-test adia-test assms test-ou-closed)

lemma abox-adia-implication:
assumes test p
and test q
shows $|R\rangle p \subseteq |R\rangle q \rightarrow |R\rangle(p * q)$
proof –
have $p \subseteq q \rightarrow p * q$
by (metis assms subset-refl test-galois-1 test-sp-commute)
hence $|R\rangle p \subseteq |R\rangle(q \rightarrow p * q)$
by (simp add: abox-right-isotone assms test-galois-1)
thus ?thesis
by (metis (no-types, lifting) Int-Un-eq(2) abox-implication-adia assms
 le-sup-iff subset-Un-eq test-galois-1)
qed

lemma abox-mp:
assumes test p
and test q
shows $|R\rangle p * |R\rangle(p \rightarrow q) \subseteq |R\rangle q$
by (metis (no-types, lifting) abox-right-isotone abox-sp assms
 semilattice-inf-class.inf.absorb-iff1 sp-test-dist-oi-left subset-refl sup-commute
 test-implication-closed test-shunting test-sp-commute)

```

lemma abox-implication:
  assumes test p
  and test q
  shows |R](p → q) ⊆ |R]p → |R]q
  by (metis abox-mp test-shunting test-sp-commute abox-test assms sup-commute
    test-implication-closed)

lemma ebox-left-dist-ou:
  assumes test p
  shows |R ∪ S]]p = |R]]p * |S]]p
  by (auto simp: mr-simp)

lemma abox-left-dist-ou:
  assumes test p
  shows |R ∪ S]p = |R]p * |S]p
  by (simp add: abox-ebox assms ebox-left-dist-ou ii-right-dist-ou ne-dist-ou)

lemma adia-left-dist-ou:
  assumes test p
  shows |R ∪ S⟩p = |R⟩p ∪ |S⟩p
  by (auto simp: mr-simp)

lemma edia-left-dist-ou:
  assumes test p
  shows |R ∪ S⟩p = |R⟩p ∪ |S⟩p
  by (simp add: assms boolean-algebra.conj-disj-distrib2 d-dist-ou edia-1)

lemma abox-dist-iu-1:
  assumes test p
  shows |R ∪ S]p = |Dom R * ne (S↓)]]p * |Dom S * ne (R↓)]]p
  proof
    have 1: |R ∪ S]p ⊆ |Dom R * ne (S↓)]]p
    by (metis abox-ebox assms d-sp-ne-down-below-ne-iu-down ebox-left-antitone)
    have |R ∪ S]p ⊆ |Dom S * ne (R↓)]]p
    by (metis abox-ebox assms d-sp-ne-down-below-ne-iu-down ebox-left-antitone
      iu-commute)
    thus |R ∪ S]p ⊆ |Dom R * ne (S↓)]]p * |Dom S * ne (R↓)]]p
      using 1 by (simp add: assms ebox)
  next
    have |Dom R * ne (S↓)]]p * |Dom S * ne (R↓)]]p ⊆ |Dom R * Dom S * ne
      (S↓)]]p * |Dom S * ne (R↓)]]p
      apply (clarify simp: mr-simp)
      by (metis UN-I singletonI)
    also have ... ⊆ |Dom R * Dom S * ne (S↓)]]p * |Dom R * Dom S * ne (R↓)]]p
      by (simp add: assms d-lb2 ebox-left-antitone s-prod-isol s-prod-isor)
    also have ... = |Dom R * Dom S * ne (S↓) ∪ Dom R * Dom S * ne (R↓)]]p
      using assms ebox-left-dist-ou by blast
    also have ... = |ne (Dom R * Dom S * S↓) ∪ ne (Dom R * Dom S * R↓)]]p
      by (metis d-dist-ii d-test test-sp-ne)

```

```

also have ... = |ne ((Dom R * Dom S * S)↓) ∪ ne ((Dom R * Dom S * R)↓)]|p
  by (simp add: down-dist-sp)
also have ... = aDom ((ne ((Dom R * Dom S * S)↓) ∪ ne ((Dom R * Dom S
* R)↓)) * l p)
  using assms ebox by blast
also have ... ⊆ aDom ((ne ((Dom R * Dom S * S ∪ Dom R * Dom S *
R)↓)) * l p)
  using d-ne-iu-down-sp-test-ou by blast
also have ... = |Dom R * Dom S * S ∪ Dom R * Dom S * R|p
  using abox assms by blast
also have ... = |Dom R * Dom S * (R ∪ S)|p
  by (metis d-assoc1 d-inter-r p-prod-comm)
also have ... = |R ∪ S|p
  by (metis c1 cl8-var d-dist-iu)
finally show |Dom R * ne (S↓)]|p * |Dom S * ne (R↓)]|p ⊆ |R ∪ S|p
.
qed

```

```

lemma abox-dist-iu-2:
assumes test p
shows |R ∪ S|p = |Dom R * S|p * |Dom S * R|p
proof -
  have |Dom R * ne (S↓)]|p * |Dom S * ne (R↓)]|p = |ne ((Dom R * S)↓)]|p *
|ne ((Dom S * R)↓)]|p
  by (simp add: d-test down-dist-sp test-sp-ne)
also have ... = |Dom R * S|p * |Dom S * R|p
  by (simp add: abox-ebox assms)
finally show ?thesis
  using assms abox-dist-iu-1 by blast
qed

```

```

lemma abox-dist-iu-3:
assumes test p
shows |R ∪ S|p = ( |R⟩1 → |S|p ) * ( |S⟩1 → |R|p )
by (metis abox-dist-iu-2 adia assms abox-test-sp d-test s-prod-idr subset-refl)

```

```

lemma abox-adia-sp-one-set:
|R⟩|S⟩1 = { (a,{a}) | a . ∀ B . (a,B) ∈ R → (∀ b∈B . ∃ D . (b,D) ∈ S) }
by (auto simp: abox-def Dom-def adia)

```

```

lemma abox-abox-set:
|R⟩|S⟩p = { (a,{a}) | a . ∀ B . (a,B) ∈ R → (∀ C . (∃ b∈B . (b,C) ∈ S) →
(∀ c∈C . (c,{c}) ∈ p)) }
by (auto simp: abox-def)

```

```

lemma sp-abox-set:
|R * S|p = { (a,{a}) | a . ∀ B . (a,B) ∈ R → (∀ C . (∃ f . (∀ b∈B . (b,f b) ∈
S) ∧ C = ∪ { f b | b . b ∈ B }) → (∀ c∈C . (c,{c}) ∈ p)) }
apply (unfold abox-def s-prod-def)

```

by *blast*

```
lemma abox-sp-1:
  assumes test p
  shows |R||S>1 * |R * S]p ⊆ |R||S]p
proof -
  have |R||S>1 * |R * S]p = |R||S>1 ∩ |R * S]p
    by (smt (verit, ccfv-SIG) abox-test adia-test assms convex-increasing
         inf.orderE inf-assoc sp-unit-convex test-s-prod-is-meet)
  also have ... ⊆ |R||S]p
  proof
    fix x
    assume x ∈ |R||S>1 ∩ |R * S]p
    from this obtain a where 1: x = (a,{a}) ∧ x ∈ |R||S>1 ∧ x ∈ |R * S]p
      by (metis Int-iff abox-test adia-test order-refl subid-aux2 subsetD surj-pair)
    hence 2: ∀ B . (a,B) ∈ R → (∀ b∈B . ∃ D . (b,D) ∈ S)
      by (smt abox-adia-sp-one-set mem-Collect-eq old.prod.inject)
    have 3: ∀ B . (a,B) ∈ R → (∀ C . (∃ f . (∀ b∈B . (b,f b) ∈ S) ∧ C = ∪ { f b
      | b . b ∈ B }) → (∀ c∈C . (c,{c}) ∈ p))
      using 1 by (smt sp-abox-set mem-Collect-eq old.prod.inject)
    have ∀ B . (a,B) ∈ R → (∀ C . (∃ b∈B . (b,C) ∈ S) → (∀ c∈C . (c,{c}) ∈
      p))
  proof (rule allI, rule impI)
    fix B
    assume 4: (a,B) ∈ R
    hence ∃ DD . ∀ b∈B . (b,DD b) ∈ S
      using 2 by (auto intro: bchoice)
    from this obtain DD where 5: ∀ b∈B . (b,DD b) ∈ S
      by auto
    show ∀ C . (∃ b∈B . (b,C) ∈ S) → (∀ c∈C . (c,{c}) ∈ p)
    proof (rule allI, rule impI)
      fix C
      assume ∃ b∈B . (b,C) ∈ S
      from this obtain b where 6: b ∈ B ∧ (b,C) ∈ S
        by auto
      let ?f = λx . if x = b then C else DD x
      let ?C = C ∪ ∪ { ?f x | x . x ∈ B ∧ x ≠ b }
      have ∃ f . (∀ b∈B . (b,f b) ∈ S) ∧ ?C = ∪ { f b | b . b ∈ B }
        apply (rule exI[where ?x=?f])
        using 5 6 by auto
      hence ∀ c∈?C . (c,{c}) ∈ p
        using 3 4 by auto
      thus ∀ c∈C . (c,{c}) ∈ p
        by blast
    qed
    qed
    thus x ∈ |R||S]p
      using 1 abox-abox-set by blast
  qed
```

```

finally show ?thesis
.

qed

lemma abox-sp-2:
assumes test p
shows |R||S|p = |R↓ * S|p
proof -
  have |R||S|p = aDom (ne (R↓) * Dom (ne (S↓) * l p))
    by (metis abox abox-test assms d-complement-ad)
  also have ... = aDom (ne (R↓) * ne (S↓) * l p)
    by (simp add: test-assoc3)
  also have ... = aDom (ne ((R↓ * S)↓) * l p)
    by (simp add: down-dist-sp ne-dist-down-sp)
  also have ... = |R↓ * S|p
    by (simp add: abox assms)
  finally show ?thesis
.

qed

lemma abox-sp-3:
assumes test p
shows |R||S|p ⊆ |R * S|p
by (clar simp simp: mr-simp) auto

lemma abox-sp-4:
assumes test p
shows |R * S|p ⊆ |R||S⟩1 → |R||S|p
proof -
  have |R||S⟩1 * |R * S|p ⊆ |R||S|p
    by (auto simp: assms abox-sp-1)
  hence |R||S⟩1 ∩ |R * S|p ⊆ |R||S|p
    by (smt (verit) abox-test abox-test assms convex-increasing inf.orderE
      sp-unit-convex test-oi-closed test-s-prod-is-meet)
  thus ?thesis
    using abox-test assms by blast
qed

lemma abox-sp-5:
assumes test p
shows |R||S⟩1 * |R * S|p = |R||S⟩1 * |R||S|p
proof (rule antisym)
  have |R * S|p ⊆ |R||S⟩1 → |R||S|p
    by (simp add: abox-sp-4 assms)
  hence |R||S⟩1 ∩ |R * S|p ⊆ |R||S|p
    by blast
  hence |R||S⟩1 ∩ |R * S|p ⊆ |R||S⟩1 ∩ |R||S|p
    by blast
  thus |R||S⟩1 * |R * S|p ⊆ |R||S⟩1 * |R||S|p

```

```

by (smt (verit, del-insts) abox-test adia-test assms convex-increasing
inf.orderE sp-unit-convex test-oi-closed test-s-prod-is-meet)
show  $|R||S\rangle 1 * |R||S\rangle p \subseteq |R||S\rangle 1 * |R * S|p$ 
by (simp add: abox-sp-3 assms s-prod-isor)
qed

lemma abox-sp-6:
assumes test p
shows  $|R||S\rangle 1 \rightarrow |R * S|p = |R||S\rangle 1 \rightarrow |R||S\rangle p$ 
by (smt Int-commute abox-sp-3 abox-sp-4 assms inf-sup-distrib2
lattice-class.inf-sup-absorb semilattice-inf-class.inf.absorb-iff2 sup-commute)

lemma abox-sp-7:
assumes test p
and total S
shows  $|R * S|p = |R||S\rangle p$ 
by (metis (no-types, lifting) abox-ebox abox-sp-2 assms down-dist-sp
total-down-dist-sp)

lemma adia-sp-associative:
assumes test p
shows  $|Q * (R * S)\rangle p = |(Q * R) * S\rangle p$ 
proof -
have  $|Q * (R * S)\rangle p = |Q\rangle (|R\rangle (|S\rangle p))$ 
by (metis (no-types, lifting) adia adia-test assms d-loc-ax inf.orderE
test-assoc3)
also have ... =  $|(Q * R) * S\rangle p$ 
by (smt (verit, best) adia adia-test assms d-loc test-assoc3
test-double-complement)
finally show  $|Q * (R * S)\rangle p = |(Q * R) * S\rangle p$ 
.

qed

lemma ebox-sp-associative:
assumes test p
shows  $|Q * (R * S)]\rangle p = |(Q * R) * S]\rangle p$ 
by (simp add: adia-sp-associative assms ebox-adia)

lemma edia-sp-associative:
assumes test p
shows  $|Q * (R * S))\rangle p = |(Q * R) * S)\rangle p$ 
proof -
have  $|fis (Q * (R * S)))\rangle p = |fis (Q * Dom (R * S)) * (fis (R * Dom S) * fis S))\rangle p$ 
by (metis fission-sp-dist)
also have ... =  $|(fis (Q * Dom (R * S)) * fis (R * Dom S)) * fis S)\rangle p$ 
by (simp add: inner-deterministic-sp-assoc semilattice-inf-class.inf-commute
semilattice-inf-class.le-infI1 fission-var)
also have ... =  $|fis (Q * Dom (R * Dom S)) * fis (R * Dom S) * fis S)\rangle p$ 

```

```

    by simp
also have ... = |fis (Q * (R * Dom S)) * fis S>>p
    by (metis fission-sp-dist)
also have ... = |fis ((Q * R) * Dom S) * fis S>>p
    by (metis d-complement-ad test-assoc3)
also have ... = |fis ((Q * R) * S)>>p
    by (metis fission-sp-dist)
finally show ?thesis
    using assms edia-fission by blast
qed

lemma abox-sp-associative:
assumes test p
shows |Q * (R * S)]p = |(Q * R) * S]p
by (simp add: edia-sp-associative assms abox-edia)

lemma abox-oI:
assumes X ≠ {}
shows |R]∩ X = (∩ p∈X . |R]p)
apply (rule antisym)
apply (clarsimp simp: mr-simp)
apply (clarsimp simp: mr-simp)
using assms by blast

lemma ebox-left-dist-oU:
assumes X ≠ {}
shows |∪ X]]p = (∩ R∈X . |R]]p)
apply (rule antisym)
apply (clarsimp simp: mr-simp)
apply blast
apply (clarsimp simp: mr-simp)
using assms by blast

lemma abox-left-dist-oU:
assumes X ≠ {}
shows |∪ X]p = (∩ R∈X . |R]p)
apply (rule antisym)
apply (clarsimp simp: mr-simp)
apply blast
apply (clarsimp simp: mr-simp)
using assms by blast

lemma adia-left-dist-oU:
|∪ X>p = (∪ R∈X . |R>p)
apply (clarsimp simp: mr-simp)
by blast

lemma edia-left-dist-oU:
|∪ X>>p = (∪ R∈X . |R>>p)

```

```

apply (clar simp simp: mr-simp)
by blast

```

8.6 Goldblatt's axioms with star

```

unbundle no rtrancl-syntax
notation star ( $\langle\cdot\rangle^*$ ) [1000] 999

lemma star-induct-1:
  assumes  $I \subseteq X$ 
  and  $R * X \subseteq X$ 
  shows  $R^* \subseteq X$ 
  apply (unfold star-def)
  apply (rule lfp-lowerbound)
  by (simp add: assms)

lemma star-induct:
  assumes  $S \subseteq I \cup I_{\cup\cup}$ 
  and  $S \subseteq X$ 
  and  $R * X \subseteq X$ 
  shows  $R^* * S \subseteq X$ 
proof -
  have  $R^* \subseteq X \oslash S$ 
  proof (rule star-induct-1)
    show  $I \subseteq X \oslash S$ 
    by (metis (no-types, opaque-lifting) Int-subset-iff assms(2) dual-order.eq-iff
      sp-lres-galois test-sp)
  next
    have  $(X \oslash S) * S \subseteq X$ 
    by (simp add: sp-lres-sp-below)
    hence  $R * (X \oslash S) * S \subseteq R * X$ 
    by (metis assms(1) s-prod-isor test-iu-test-sp-assoc-5)
    also have ...  $\subseteq X$ 
    by (simp add: assms(3))
    finally show  $R * (X \oslash S) \subseteq X \oslash S$ 
    by (simp add: sp-lres-galois)
  qed
  thus ?thesis
  by (simp add: sp-lres-galois)
qed

lemma star-total:
  total ( $R^*$ )
  by (metis s-prod-idl s-prod-isol star-refl total-4)

lemma star-down:
   $R^* \downarrow = (R \downarrow)^* \cup I_{\cup\cup}$ 
proof
  have  $R^* * (I \cup I_{\cup\cup}) \subseteq (R \downarrow)^* \cup I_{\cup\cup}$ 

```

```

proof (rule star-induct)
  show  $1 \cup 1_{\cup\cup} \subseteq 1 \cup 1_{\cup\cup}$ 
    by simp
next
  show  $1 \cup 1_{\cup\cup} \subseteq (R\downarrow)^* \cup 1_{\cup\cup}$ 
    using star-refl by auto
next
  have  $ne(R * ((R\downarrow)^* \cup 1_{\cup\cup})) \subseteq ne(R\downarrow * ((R\downarrow)^* \cup 1_{\cup\cup}))$ 
    by (simp add: down-sp-sp sup-commute)
  also have ... =  $ne(R\downarrow) * ne((R\downarrow)^* \cup 1_{\cup\cup})$ 
    by (simp add: ne-dist-down-sp)
  also have ... =  $ne(R\downarrow) * ne((R\downarrow)^*)$ 
    by (metis down-idempotent down-sp-sp ne-dist-down-sp sup-commute)
  also have ...  $\subseteq R\downarrow * (R\downarrow)^*$ 
    using sp-oi-subdist by blast
  also have ...  $\subseteq (R\downarrow)^*$ 
    using star-unfold-eq by blast
  finally show  $R * ((R\downarrow)^* \cup 1_{\cup\cup}) \subseteq (R\downarrow)^* \cup 1_{\cup\cup}$ 
    by blast
qed
thus  $R^*\downarrow \subseteq (R\downarrow)^* \cup 1_{\cup\cup}$ 
  by (simp add: down-sp sup-commute)
next
  have  $(R\downarrow)^* \subseteq R^*\downarrow$ 
  proof (rule star-induct-1)
    show  $1 \subseteq R^*\downarrow$ 
      by (simp add: star-refl subset-lower)
next
  show  $R\downarrow * R^*\downarrow \subseteq R^*\downarrow$ 
    by (metis total-dom Un-Int-eq(1) d-isotone d-test ii-right-dist-ou inf-le2
le-sup-iff s-subid-iff2 star-unfold-eq subset-antisym total-down-dist-sp)
  qed
  thus  $(R\downarrow)^* \cup 1_{\cup\cup} \subseteq R^*\downarrow$ 
    using star-total total-lower by blast
qed

lemma ne-star-down:
  ne  $(R^*\downarrow) = ne((R\downarrow)^*)$ 
  by (simp add: ne-dist-ou star-down)

lemma ne-down-star:
  ne  $((R\downarrow)^*) = (ne(R\downarrow))^*$ 
proof
  have  $(R\downarrow)^* \subseteq (ne(R\downarrow))^* \cup 1_{\cup\cup}$ 
  proof (rule star-induct-1)
    show  $1 \subseteq (ne(R\downarrow))^* \cup 1_{\cup\cup}$ 
      by (simp add: le-supI1 star-refl)
next
  have  $ne(R\downarrow * ((ne(R\downarrow))^* \cup 1_{\cup\cup})) = ne(R\downarrow) * ne((ne(R\downarrow))^*)$ 

```

```

    by (metis down-idempotent down-sp-sp ne-dist-down-sp sup-commute)
  also have ... ⊆ (ne (R↓))*
    by (metis (no-types, lifting) IntE UnCI inf.absorb-iff2 sp-oi-subdist
star-unfold-eq subsetI)
  finally show R↓ * ((ne (R↓))*)* ⊆ ((ne (R↓))*)* ∪ 1_{UU}
    by blast
qed
thus ne ((R↓))* ⊆ (ne (R↓))*
  by (smt Compl-disjoint2 Int-commute Int-left-commute ne-dist-ou
semilattice-inf-class.le-iff-inf sup-bot.right-neutral)
next
show (ne (R↓))* ⊆ ne ((R↓))*
proof (rule star-induct-1)
show 1 ⊆ ne ((R↓))*
  using star-refl test-ne by auto
next
show ne (R↓) * ne ((R↓))* ⊆ ne ((R↓))*
  by (metis IntE IntI UnCI ne-dist-down-sp star-unfold-eq subsetI)
qed
qed

lemma abox-star-unfold:
test p ==> |R*]p = p * |R| |R*]p
by (metis abox-left-dist-ou abox-sp-7 sp-unit-abox star-total star-unfold-eq)

lemma star-sp-test-commute:
assumes S ⊆ 1 ∪ 1_{UU}
and Q * S ⊆ S * R
shows Q* * S ⊆ S * R*
proof (rule star-induct)
show S ⊆ 1 ∪ 1_{UU}
  by (simp add: assms(1))
next
show S ⊆ S * R*
  by (metis s-prod-idr s-prod-isor star-refl)
next
have Q * (S * R*) ⊆ S * R * R*
  by (metis (no-types, lifting) assms s-prod-distr subset-Un-eq
test-iu-test-sp-assoc-3)
thus Q * (S * R*) ⊆ S * R*
  by (metis (no-types, lifting) UnCI dual-order.trans s-prod-assoc1 s-prod-isor
star-unfold subset-eq)
qed

lemma adia-star-induct:
assumes test p
shows |R]p ⊆ p ↔ |R*]p ⊆ p
proof
assume |R]p ⊆ p

```

```

hence  $\lambda p * \text{Dom } (R * p) = \{\}$ 
  by (metis adia assms d-idem2 s-prod-isol subset-empty test-sp-shunting)
hence  $R * p \subseteq p * (R * p)$ 
  by (metis assms d-sp-strict subset-refl test-sp-shunting)
hence  $R * p \subseteq p * R$ 
  by (metis assms inf.absorb-iff2 inf-commute sp-test-dist-oi-right test-assoc3
test-sp-idempotent)
hence  $R^* * p \subseteq p * R^*$ 
  by (simp add: assms le-supI1 star-sp-test-commute)
hence  $R^* * p \subseteq p * (R^* * p)$ 
  by (metis assms inf.absorb-iff2 inf.orderE sp-oi-subdist test-assoc3
test-sp-idempotent)
hence  $\lambda p * (R^* * p) = \{\}$ 
  by (meson assms subset-empty test-sp-shunting)
hence  $\lambda p * \text{Dom } (R^* * p) = \{\}$ 
  using d-sp-strict by blast
thus  $|R^* \rangle p \subseteq p$ 
  by (metis adia assms d-test empty-subsetI semilattice-inf-class.le-inf-iff sp-test
test-sp-shunting)
next
assume  $|R^* \rangle p \subseteq p$ 
thus  $|R \rangle p \subseteq p$ 
  by (metis adia-left-isotone assms dual-order.trans s-prod-idr s-prod-isor
star-refl star-unfold sup.coboundedI2)
qed

lemma ebox-star-induct:
assumes test p
shows  $p \subseteq |R|]p \longleftrightarrow p \subseteq |R^*|]p$ 
by (smt (verit, best) adia adia-star-induct assms d-complement-ad ebox-adia
test-complement-antitone test-double-complement)

lemma abox-star-induct:
assumes test p
shows  $p \subseteq |R]p \longleftrightarrow p \subseteq |R^*]p$ 
proof -
  have  $p \subseteq |ne (R\downarrow)]]p \longleftrightarrow p \subseteq |ne (R^*\downarrow)]]p$ 
    by (metis assms ebox-star-induct ne-down-star ne-star-down)
  thus ?thesis
    by (metis abox-ebox assms)
qed

lemma edia-star-induct:
assumes test p
shows  $|R\rangle\rangle p \subseteq p \longleftrightarrow |R^*\rangle\rangle p \subseteq p$ 
by (metis adia-star-induct assms edia-adia ne-down-star ne-star-down)

lemma abox-star-induct-1:
assumes test p

```

```

and test q
and  $q \subseteq p * |R|q$ 
shows  $q \subseteq |R^*|p$ 
proof -
  have  $q \subseteq p \wedge q \subseteq |R^*|q$ 
  by (metis Int-subset-iff abox-star-induct abox-test assms test-sp
    test-sp-commute)
  thus ?thesis
  using abox-right-isotone assms(1,2) by blast
qed

lemma adia-star-induct-1:
  assumes test p
  and test q
  and  $p \cup |R\rangle q \subseteq q$ 
  shows  $|R^*|p \subseteq q$ 
  by (meson adia-right-isotone adia-star-induct assms order.trans sup.bounded-iff)

lemma abox-segerberg:
  assumes test p
  shows  $|R^*|(p \rightarrow |R|p) \subseteq p \rightarrow |R^*|p$ 
proof -
  have  $p * |R^*|(p \rightarrow |R|p) \subseteq |R^*|p$ 
  proof (rule abox-star-induct-1)
    show test p
    by (simp add: assms)
  next
    show test ( $p * |R^*|(p \rightarrow |R|p)$ )
    by (simp add: abox-test assms test-galois-1)
  next
    have  $p * |R^*|(p \rightarrow |R|p) = p * (p \rightarrow |R|p) * |R||R^*|(p \rightarrow |R|p)$ 
    by (metis abox-star-unfold abox-test assms inf-le2 le-infE sp-unit-convex
      sp-unit-down test-iu-test-sp-assoc-1 test-ou-closed)
    also have ... =  $p * |R|p * |R||R^*|(p \rightarrow |R|p)$ 
    by (smt (verit, best) Un-Int-eq(4) abox-left-dist-ou abox-test assms
      equalityD1 le-infE s-prod-isol sp-unit-abox sp-unit-convex sp-unit-down
      subset-Un-eq subset-antisym test-galois-1 test-iu-test-sp-assoc-1 test-ou-closed
      test-sp-commute)
    also have ... =  $p * |R|(p * |R^*|(p \rightarrow |R|p))$ 
    by (metis (no-types, lifting) abox-sp abox-test abox-test-sp-3 assms
      test-assoc2 test-double-complement)
    finally show  $p * |R^*|(p \rightarrow |R|p) \subseteq p * |R|(p * |R^*|(p \rightarrow |R|p))$ 
    by simp
  qed
  thus ?thesis
  by (meson abox-test assms test-galois-1 test-implication-closed)
qed

lemma abox-segerberg-adia:

```

```

assumes test p
shows  $|R^*|(|R\rangle p \rightarrow p) \subseteq |R^*|p \rightarrow p$ 
proof -
let ?q =  $|R^*|(|R\rangle p \rightarrow p)$ 
have  $|R^*|p \subseteq ?q \rightarrow p$ 
proof (rule adia-star-induct-1)
show test p
by (simp add: assms)
next
show test (?q \rightarrow p)
by (simp add: assms)
next
have  $|R\rangle(?q \rightarrow p) * |R|?q * (|R\rangle p \rightarrow p) \subseteq |R\rangle p * (|R\rangle p \rightarrow p)$ 
by (metis (no-types, lifting) abox-adia-mp abox-test assms inf.absorb-iff2
sp-test-dist-oi test-implication-closed)
also have ... \subseteq p
by (meson adia-test assms equalityD2 test-galois-1 test-implication-closed)
finally have  $|R\rangle(?q \rightarrow p) \subseteq (|R\rangle p \rightarrow p) * |R|?q \rightarrow p$ 
by (smt (verit) abox-star-unfold abox-test adia assms d-complement-ad
test-assoc3 test-double-complement test-galois-1 test-implication-closed
test-sp-commute)
also have ... = ?q \rightarrow p
by (metis abox-star-unfold assms test-implication-closed)
finally show p \cup |R\rangle(?q \rightarrow p) \subseteq ?q \rightarrow p
by (metis le-sup-iff order-refl)
qed
thus ?thesis
by (smt abox-test adia-test assms sup-commute test-galois-1
test-implication-closed test-shunting)
qed

```

lemma s-p-id-sp:
 $(s\text{-}id \cup p\text{-}id) * R = R \cup p\text{-}id$
by (simp add: s-prod-distr)

8.7 Propositional Hoare logic

abbreviation hoare :: ('a,'a) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('b,'b) mrel \Rightarrow bool ($\langle \cdot \rangle$ -{})-
[50,60,50] 95
where $p\{R\}q \equiv p \subseteq |R|q$

abbreviation if-then-else :: ('a,'a) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('a,'b)
mrel
where if-then-else p R S \equiv $p * R \cup \lambda p * S$

abbreviation while-do :: ('a,'a) mrel \Rightarrow ('a,'a) mrel \Rightarrow ('a,'a) mrel
where while-do p R \equiv $(p * R)^* * \lambda p$

lemma hoare-skip:

```

assumes test p
  shows  $p\{1\}p$ 
  by (simp add: assms sp-unit-abox)

lemma hoare-cons:
  assumes test s
    and  $r \subseteq p$ 
    and  $q \subseteq s$ 
    and  $p\{R\}q$ 
  shows  $r\{R\}s$ 
  by (meson abox-right-isotone assms order-trans)

lemma hoare-seq:
  assumes test q
    and test r
    and  $p\{R\}q$ 
    and  $q\{S\}r$ 
  shows  $p\{R*S\}r$ 
proof -
  have  $p \subseteq |R|q$ 
    by (simp add: assms(3))
  also have ...  $\subseteq |R||S|r$ 
    by (simp add: abox-right-isotone abox-test assms(1,2,4))
  also have ...  $\subseteq |R*S|r$ 
    by (simp add: abox-sp-3 assms(2))
  finally show ?thesis
    by simp
qed

lemma hoare-if:
  assumes test p
    and test q
    and test r
    and  $(p*q)\{R\}r$ 
    and  $((\lambda p)*q)\{S\}r$ 
  shows  $q\{\text{if-then-else } p \ R \ S\}r$ 
  by (smt (verit, ccfv-threshold) abox-left-dist-ou abox-test-sp assms inf.absorb-iff2 inf-le2 sp-oi-subdist test-galois-1 test-sp-idempotent)

lemma hoare-while:
  assumes test p
    and test q
    and  $(p*q)\{R\}q$ 
  shows  $q\{\text{while-do } p \ R\}(q*(\lambda p))$ 
proof -
  have  $q \subseteq p \rightarrow |R|q$ 
    by (meson assms test-galois-1)
  also have ...  $= |p||R|q$ 
    by (simp add: abox-test assms(1,2) test-abox)

```

```

also have ...  $\subseteq |p*R]q$ 
  by (simp add: abox-sp-3 assms(2))
finally have q  $\subseteq |(p*R)^*]q$ 
  by (meson abox-star-induct assms(2))
also have ...  $\subseteq |(p*R)^*](q \cup p)$ 
  by (simp add: abox-right-isotone assms(1,2))
also have ... =  $|(p*R)^*](\lambda p \rightarrow (q*(\lambda p)))$ 
  by (smt (verit, best) Un-assoc Un-upper1 assms(1,2) inf-commute
sup.commute sup.order-iff test-double-complement test-galois-1 test-s-prod-is-meet)
also have ... =  $|(p*R)^*][\lambda p](q*(\lambda p))$ 
  by (simp add: assms(2) test-abox test-shunting)
also have ...  $\subseteq |\text{while-do } p R](q*(\lambda p))$ 
  by (simp add: abox-sp-3 assms(2) test-galois-1)
finally show ?thesis
  by simp
qed

```

```

lemma hoare-par:
assumes test q
  and  $p\{R\}q$ 
  and  $p\{S\}q$ 
shows  $p\{R \cup S\}q$ 
proof -
  have 1:  $p \subseteq |R\rangle 1 \rightarrow |S]q$ 
  by (simp add: assms(3) le-supI2)
  have  $p \subseteq |S\rangle 1 \rightarrow |R]q$ 
  by (simp add: assms(2) le-supI2)
  hence  $p \subseteq (|R\rangle 1 \rightarrow |S]q) \cap (|S\rangle 1 \rightarrow |R]q)$ 
  using 1 by simp
  also have ... =  $|R \cup S]q$ 
  by (smt (verit) abox-dist-iu-3 abox-test assms(1) inf.orderE
test-implication-closed test-oi-closed test-s-prod-is-meet)
finally show ?thesis
  by simp
qed

```

9 Counterexamples

```

locale counterexamples
begin

lemma counter-01:
   $\neg ((U::('a,'b) mrel) * -((U::('b,'c) mrel) * (R::('c,'d) mrel)) \subseteq -(U * R))$ 
  by (metis UNIV-I U-par-zero disjoint-eq-subset-Compl emptyE
iu-unit-below-top-sp-test iu-unit-up le-inf-iff s-prod-zerol subset-empty
top-upper-least)

```

abbreviation a-1 ≡ finite-1.a1

lemma counter-02:

$$\exists R::(\text{Enum.finite-1}, \text{Enum.finite-1}) \text{ mrel} . \exists p . \neg (\text{test } p \longrightarrow (R \cap -(U * p)) * U = R * -(p * U))$$

apply (rule exI[where ?x={(a-1,{})}])
 apply (rule exI[where ?x={{} }])
 apply (clar simp simp: s-id-def)
 by (smt (verit, ccfv-SIG) Compl-empty-eq Diff-eq Int-insert-left-if0 U-par-p-id cl8-var complement-test-sp-top d-U d-sp-strict dc empty-subsetI inf-le2 inf-top-left inner-total-2 insert-not-empty s-prod-zerol x-split-var x-y-split zero-nc)

lemma counter-03:

$$\exists R::(\text{Enum.finite-1}, \text{Enum.finite-1}) \text{ mrel} . \exists p . \neg (\text{test } p \longrightarrow (R \cap -(U * p)) * 1_{\cup\cup} = R * -(p * U) \cap 1_{\cup\cup})$$

apply (rule exI[where ?x={(a-1,{})}])
 apply (rule exI[where ?x={{} }])
 apply (clar simp simp: s-id-def)
 by (smt (z3) Int-Un-eq(3) Int-absorb2 U-c ad-sp-bot cd-iso dc-prop1 disjoint-eq-subset-Compl inf-compl-bot-right inner-total-2 insertI1 p-id-zero singleton-Un-iff sp-oi-subdist)

abbreviation b-1 ≡ finite-2.a₁
abbreviation b-2 ≡ finite-2.a₂
abbreviation b-1-0 ≡ (b-1,{ })
abbreviation b-1-1 ≡ (b-1,{b-1})
abbreviation b-1-2 ≡ (b-1,{b-2})
abbreviation b-1-3 ≡ (b-1,{b-1,b-2})
abbreviation b-2-0 ≡ (b-2,{ })
abbreviation b-2-1 ≡ (b-2,{b-1})
abbreviation b-2-2 ≡ (b-2,{b-2})
abbreviation b-2-3 ≡ (b-2,{b-1,b-2})

lemma counter-04:

$$\exists R::(\text{Enum.finite-2}, \text{Enum.finite-2}) \text{ mrel} . \exists p q . \neg (\text{test } p \longrightarrow \text{test } q \longrightarrow |R * p]q = |R|p]q)$$

apply (rule exI[where ?x={b-1-3}])
 apply (rule exI[where ?x={b-1-1}])
 apply (rule exI[where ?x={{} }])
 apply (subst sp-test)
 apply (clar simp simp: s-id-def)
 apply (subst top-test)
 apply (clar simp simp: s-id-def)
 apply (unfold abox-def)
 apply (clar simp simp: s-id-def)
 by blast

lemma counter-05:

$$\neg (\exists f . \forall R p . \text{test } p \longrightarrow |R\rangle p = |f R]p)$$

by (smt (verit, ccfv-threshold) Int-lower1 Int-lower2 abox-sp-unit adia-test-sp counter-01 iu-test-sp-left-zero s-prod-idl subset-refl)

lemma counter-06:
 $\neg (\exists f . \forall R p . test p \longrightarrow |R|p = |f R|p)$
by (metis abox-adia-fusion abox-fusion abox-sp-unit adia counter-05
d-complement-ad disjoint-eq-subset-Compl ebox-adia empty-subsetI s-prod-idr
order-refl)

lemma counter-07:
 $\neg (\exists f . mono f \wedge (\forall R . fus R = lfp (\lambda X . f R X)))$
proof
assume $\exists f::('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('a,'b) mrel . mono f \wedge (\forall R . fus R = lfp (\lambda X . f R X))$
from this obtain $f :: ('a,'b) mrel \Rightarrow ('a,'b) mrel \Rightarrow ('a,'b) mrel$ **where** $mono f \wedge (\forall R . fus R = lfp (\lambda X . f R X))$
by auto
hence $fus \{\} \subseteq fus (U::('a,'b) mrel)$
by (simp add: le-fun-def mono-def lfp-mono)
thus False
by (auto simp: mr-simp)
qed

abbreviation c-1 \equiv finite-3.a1
abbreviation c-2 \equiv finite-3.a2
abbreviation c-3 \equiv finite-3.a3

lemma counter-08:
 $\neg (\sim(1::(Enum.finite-3,Enum.finite-3) mrel) * \sim 1 \in \{1, \sim 1\})$
proof –
let ?c = (c-1,{c-1,c-2,c-3})
have 1: ?c $\in \sim 1 * \sim 1$
apply (clar simp simp: mr-simp)
apply (rule exI[where ?x={c-2,c-3}])
using UNIV-finite-3 **by** auto
have ?c $\notin 1 \wedge ?c \notin \sim 1$
by (auto simp: mr-simp)
thus ?thesis
using 1 **by** auto
qed

lemma counter-09:
 $\neg (\sim(1::(Enum.finite-3,Enum.finite-3) mrel) \odot 1 \in \{1, \sim 1\})$
by (metis counter-08 co-prod empty-iff ic-involutive insert-iff)

lemma ex-2-cases:
 $\exists b. b = b-1 \vee b = b-2$
by auto

lemma all-2-cases:
 $(\forall b. b = b-2 \wedge b = b-1) = False$

```

by auto

lemma impl-2-cases:
   $\bigcup \{ X . \exists b. (b = b\text{-}1 \longrightarrow X = Y) \wedge (b = b\text{-}2 \longrightarrow X = Z) \} = Y \cup Z$ 
  by auto

lemma ex-2-set-cases:
   $(\exists B::\text{Enum.finite-2 set} . P B) \longleftrightarrow P \{\} \vee P \{b\text{-}1\} \vee P \{b\text{-}2\} \vee P \{b\text{-}1,b\text{-}2\}$ 
proof -
  let ?U = UNIV::\text{Enum.finite-2 set set}
  have ?U ⊆ \{\{\},\{b\text{-}1\},\{b\text{-}2\},\{b\text{-}1,b\text{-}2\}\}
  proof
    fix x
    have x ⊆ \{b\text{-}1,b\text{-}2\}
    by auto
    thus x ∈ \{\{\},\{b\text{-}1\},\{b\text{-}2\},\{b\text{-}1,b\text{-}2\}\}
    by auto
  qed
  hence ?U = \{\{\},\{b\text{-}1\},\{b\text{-}2\},\{b\text{-}1,b\text{-}2\}\}
  by auto
  thus ?thesis
  by (metis UNIV-I empty-ifff insertE)
qed

abbreviation B-0 ≡ \{\}::\text{Enum.finite-2 set}
abbreviation B-1 ≡ \{b\text{-}1\}
abbreviation B-2 ≡ \{b\text{-}2\}
abbreviation B-3 ≡ \{b\text{-}1,b\text{-}2\}
abbreviation mkf x y ≡ λz . if z = b\text{-}1 then x else y

lemma mkf:
  f = mkf (f b\text{-}1) (f b\text{-}2)
  by auto

lemma mkf2:
  f b\text{-}1 = X ∧ f b\text{-}2 = Y ⟹ f = mkf X Y
  by auto

lemma ex-2-mrel-cases:
   $(\exists f::\text{Enum.finite-2} \Rightarrow \text{Enum.finite-2 set} . P f) \longleftrightarrow$ 
   $P (\text{mkf } B\text{-}0 B\text{-}0) \vee P (\text{mkf } B\text{-}0 B\text{-}1) \vee P (\text{mkf } B\text{-}0 B\text{-}2) \vee P (\text{mkf } B\text{-}0 B\text{-}3) \vee$ 
   $P (\text{mkf } B\text{-}1 B\text{-}0) \vee P (\text{mkf } B\text{-}1 B\text{-}1) \vee P (\text{mkf } B\text{-}1 B\text{-}2) \vee P (\text{mkf } B\text{-}1 B\text{-}3) \vee$ 
   $P (\text{mkf } B\text{-}2 B\text{-}0) \vee P (\text{mkf } B\text{-}2 B\text{-}1) \vee P (\text{mkf } B\text{-}2 B\text{-}2) \vee P (\text{mkf } B\text{-}2 B\text{-}3) \vee$ 
   $P (\text{mkf } B\text{-}3 B\text{-}0) \vee P (\text{mkf } B\text{-}3 B\text{-}1) \vee P (\text{mkf } B\text{-}3 B\text{-}2) \vee P (\text{mkf } B\text{-}3 B\text{-}3)$ 
proof
  assume ∃f::\text{Enum.finite-2} ⇒ \text{Enum.finite-2 set} . P f
  from this obtain f where 1: P f
  by auto
  have ⋀x . f x ⊆ B-3

```

```

by auto
hence 2:  $\bigvee f x = B-0 \vee f x = B-1 \vee f x = B-2 \vee f x = B-3$ 
by auto
have  $f = \text{mkf } B-0 \vee f = \text{mkf } B-0 \vee f = \text{mkf } B-0 \vee f = \text{mkf } B-0$   

 $B-3 \vee f = \text{mkf } B-1 \vee f = \text{mkf } B-1 \vee f = \text{mkf } B-1 \vee f = \text{mkf } B-1$   

 $B-3 \vee f = \text{mkf } B-2 \vee f = \text{mkf } B-2 \vee f = \text{mkf } B-2 \vee f = \text{mkf } B-2$   

 $B-3 \vee f = \text{mkf } B-3 \vee f = \text{mkf } B-3 \vee f = \text{mkf } B-3 \vee f = \text{mkf } B-3$ 
using 2[of b-1] 2[of b-2] mkf2[of f] by blast
thus  $P(\text{mkf } B-0 \vee B-1) \vee P(\text{mkf } B-0 \vee B-2) \vee P(\text{mkf } B-0 \vee B-3)$   

 $\vee P(\text{mkf } B-1 \vee B-0) \vee P(\text{mkf } B-1 \vee B-1) \vee P(\text{mkf } B-1 \vee B-2) \vee P(\text{mkf } B-1 \vee B-3)$   

 $\vee P(\text{mkf } B-2 \vee B-0) \vee P(\text{mkf } B-2 \vee B-1) \vee P(\text{mkf } B-2 \vee B-2) \vee P(\text{mkf } B-2 \vee B-3)$   

 $\vee P(\text{mkf } B-3 \vee B-0) \vee P(\text{mkf } B-3 \vee B-1) \vee P(\text{mkf } B-3 \vee B-2) \vee P(\text{mkf } B-3 \vee B-3)$ 
using 1 by auto
next
assume  $P(\text{mkf } B-0 \vee B-1) \vee P(\text{mkf } B-0 \vee B-2) \vee P(\text{mkf } B-0 \vee B-3)$   

 $\vee P(\text{mkf } B-1 \vee B-0) \vee P(\text{mkf } B-1 \vee B-1) \vee P(\text{mkf } B-1 \vee B-2) \vee P(\text{mkf } B-1 \vee B-3)$   

 $\vee P(\text{mkf } B-2 \vee B-0) \vee P(\text{mkf } B-2 \vee B-1) \vee P(\text{mkf } B-2 \vee B-2) \vee P(\text{mkf } B-2 \vee B-3)$   

 $\vee P(\text{mkf } B-3 \vee B-0) \vee P(\text{mkf } B-3 \vee B-1) \vee P(\text{mkf } B-3 \vee B-2) \vee P(\text{mkf } B-3 \vee B-3)$ 
thus  $\exists f::\text{Enum.finite-2} \Rightarrow \text{Enum.finite-2 set} . P f$ 
by auto
qed

lemma counter-10:
 $\exists R::(\text{Enum.finite-2}, \text{Enum.finite-2}) \text{ mrel} . \neg (U::(\text{Enum.finite-2}, \text{Enum.finite-2})$   

 $\text{mrel}) * (U * R) \subseteq U * R$ 
apply (rule exI[where ?x={b-1-1,b-1-2}])
apply (unfold s-prod-def)
apply (unfold ex-2-set-cases)
apply (unfold ex-2-mrel-cases)
apply (clarify simp: mr-simp ex-2-cases all-2-cases impl-2-cases)
by auto

lemma counter-11:
 $\exists (R::(\text{Enum.finite-2}, \text{Enum.finite-2}) \text{ mrel}) (s::(\text{Enum.finite-2}, \text{Enum.finite-2})$   

 $\text{mrel}) (t::(\text{Enum.finite-2}, \text{Enum.finite-2}) \text{ mrel}) . \neg (\text{inner-univalent } s \wedge$   

 $\text{inner-univalent } t \longrightarrow R * (s * t) = (R * s) * t)$ 
apply (rule exI[where ?x={b-1-3}])
apply (rule exI[where ?x={b-1-1,b-2-1}])
apply (rule exI[where ?x={b-1-1,b-1-2}])

```

```

apply (unfold s-prod-def)
apply (unfold ex-2-set-cases)
apply (unfold ex-2-mrel-cases)
apply (clar simp simp: mr-simp ex-2-cases all-2-cases impl-2-cases)
by (auto simp: times-eq-if)

lemma counter-12:
   $\neg(\exists S . 1_{\cup\cup} \odot S = 1_{\cup\cup})$ 
  by (metis Int-absorb2 UNIV-I U-U cl9 co-prod cp-ii-unit-upper
disjoint-iff-not-equal ic-antidist-ii ic-iu-unit ic-top iu-unit-down p-prod-comm
p-prod-idl s-prod-idl s-prod-p-idl)

lemma counter-13:
   $\neg(\exists S . \forall R . R \odot S = R)$ 
  by (meson counter-12)
end

end

```

References

- [1] H. Furusawa, W. Guttmann, and G. Struth. Determinism of multirelations. *arXiv*, 2305.11344, 2023. <https://arxiv.org/abs/2305.11344>.
- [2] H. Furusawa, W. Guttmann, and G. Struth. Modal algebra of multirelations. *arXiv*, 2305.11346, 2023. <https://arxiv.org/abs/2305.11346>.
- [3] H. Furusawa, W. Guttmann, and G. Struth. On the inner structure of multirelations. *arXiv*, 2305.11342, 2023. <https://arxiv.org/abs/2305.11342>.
- [4] H. Furusawa and G. Struth. Binary multirelations. *Archive of Formal Proofs*, 2015. Formal proof development, <https://isa-afp.org/entries/Multirelations.html>.
- [5] W. Guttmann. Algebras for iteration, infinite executions and correctness of sequential computations. *Archive of Formal Proofs*, 2021. Formal proof development, https://isa-afp.org/entries/Correctness_Algebras.html.