Inner Structure, Determinism and Modal Algebra of Multirelations

Walter Guttmann and Georg Struth

September 13, 2023

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

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** ; *75*)
notation *converse* ($-^{\smile}$ [*1000*] *999*)

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

lemma *modular-law*: $R ; S \cap T \subseteq (R \cap T ; S^{\smile}) ; S$
  by *blast*

lemma *compl-conv*: $-(R^{\smile}) = (-R)^{\smile}$
  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 $R = (R^{\smile} ; R \subseteq Id)$

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

definition *injective* :: $('a,'b)$ *rel* $\Rightarrow$ *bool* **where**

injective R = (R ; $R^{\smile}$ ⊆ *Id*)

definition *surjective* :: ('a,'b) *rel* ⇒ *bool* **where**
surjective R = (*Id* ⊆ $R^{\smile}$; R)

definition *deterministic* :: ('a,'b) *rel* ⇒ *bool* **where**
deterministic R = (*univalent* R ∧ *total* R)

definition *bijective* :: ('a,'b) *rel* ⇒ *bool* **where**
bijective R = (*injective* R ∧ *surjective* R)

lemma *univalent-set*: *univalent* R = (∀ a b c. (a,b) ∈ R ∧ (a,c) ∈ R ⟶ 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 = (∀ a. ∃ b. (a,b) ∈ 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 = (∀ a . ∃!B . (a,B) ∈ 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 = (∀ S. R ; −S = −(R ; S))
unfolding *deterministic-set* **by** (*standard*, *force*, *blast*)

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

lemma *injective-set*: *injective* S = (∀ a b c. (a,c) ∈ S ∧ (b,c) ∈ S ⟶ a = b)
by (*meson converseD converseI inj-univ univalent-set*)

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

lemma *surjective-set*: *surjective* S = (∀ b. ∃ a. (a,b) ∈ S)
by (*simp add*: *surj-tot total-set*)

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

lemma *bij-det*: *bijective* R = *deterministic* ($R^{\smile}$)

by (*simp add*: *bijective-def deterministic-def inj-univ surj-tot*)

lemma *univ-relcomp*: *univalent R* $\Longrightarrow$ *univalent S* $\Longrightarrow$ *univalent* (*R* ; *S*)
by (*simp add*: *single-valued-relcomp univ-single-valued*)

lemma *tot-relcomp*: *total R* $\Longrightarrow$ *total S* $\Longrightarrow$ *total* (*R* ; *S*)
by (*meson relcomp.simps total-set*)

lemma *det-relcomp*: *deterministic R* $\Longrightarrow$ *deterministic S* $\Longrightarrow$ *deterministic* (*R* ; *S*)
by (*simp add*: *deterministic-def tot-relcomp univ-relcomp*)

lemma *inj-relcomp*: *injective R* $\Longrightarrow$ *injective S* $\Longrightarrow$ *injective* (*R* ; *S*)
by (*simp add*: *converse-relcomp inj-univ univ-relcomp*)

lemma *surj-relcomp*: *surjective R* $\Longrightarrow$ *surjective S* $\Longrightarrow$ *surjective* (*R* ; *S*)
by (*simp add*: *converse-relcomp surj-tot tot-relcomp*)

lemma *bij-relcomp*: *bijective R* $\Longrightarrow$ *bijective S* $\Longrightarrow$ *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* $\Longrightarrow R \;;\; (S \cap T) = R \;;\; S \cap R \;;\; T$
unfolding *univalent-set relcomp-def relcompp-apply* **by** *force*

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

lemma *univ-modular*: *univalent S* $\Longrightarrow R \;;\; S \cap T = (R \cap T \;;\; S^{\smile}) \;;\; S$
unfolding *univalent-set converse-unfold relcomp.simps* **by** *force*

1.2 Inverse image and the diagonal and graph functors

definition *Invim* :: ('*a*,'*b*) *rel* ⇒ '*b set* ⇒ '*a set* **where**
Invim R = *Image* ($R^{\smile}$)

definition *Delta* :: '*a set* ⇒ ('*a*,'*a*) *rel* (Δ) **where**
$\Delta\ P = \{(p,p)\ |p.\ p \in P\}$

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

lemma *Image-Grph* [*simp*]: *Image* $\circ$ *Grph* $=$ *image*
unfolding *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 *dom* :: $('a,'b)$ *rel* $\Rightarrow$ $('a,'a)$ *rel* **where**
dom R $=$ *Id* $\cap$ *R* ; $R^{\smile}$

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

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

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

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

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

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

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

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

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

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

lemma *rel-fdia-set*: $|R\rangle\ Q = \{(a,a)\ |a.\ \exists\, b.\ (a,b) \in R \wedge (b,b) \in$ *dom* $Q\}$

unfolding *rel-fdia-def dom-set relcomp-unfold* **by** *force*

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

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

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

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

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

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

lemma *dom-subid*: $(dom\ R = R) = (R = Id \cap R)$
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*: $(dom\ R = R) = (cod\ R = R)$
by (*metis dom-def cod-def converse-Id converse-Int converse-converse converse-relcomp*)

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

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

lemma *cod-top*: $cod\ R = Id \cap top\ ;\ R$
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*]: $(dom\ R)^{\smile} = dom\ R$
by (*simp add*: *dom-def converse-Int converse-relcomp*)

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

lemma *surj-cod*: $surjective\ R = (cod\ R = Id)$
by (*simp add*: *cod-def surj-tot total-dom*)

lemma *fdia-demod*: $(\ |R\rangle\ P \subseteq dom\ Q) = (R\ ;\ dom\ P \subseteq dom\ Q\ ;\ R)$
unfolding *rel-fdia-set dom-set relcomp-unfold* **by** *force*

lemma *bbox-demod*: $(dom\ P \subseteq [R|\ Q) = (R\ ;\ dom\ P \subseteq dom\ Q\ ;\ R)$
unfolding *rel-bbox-set dom-def* **by** *force*

lemma *bdia-demod*: $(\langle R|\ P \subseteq dom\ Q) = (dom\ P\ ;\ R \subseteq R\ ;\ dom\ Q)$
proof −
have $(\langle R|\ P \subseteq dom\ Q) = (\ |R^{\smile}\rangle\ P \subseteq dom\ Q)$
by (*simp add*: *rel-bdia-def*)
also have $\ldots = ((R^{\smile})\ ;\ dom\ P \subseteq dom\ Q\ ;\ (R^{\smile}))$
by (*simp add*: *fdia-demod*)
also have $\ldots = (((dom\ P\ ;\ R)^{\smile}) \subseteq ((R\ ;\ dom\ Q)^{\smile}))$
by (*simp add*: *converse-relcomp*)
also have $\ldots = (dom\ P\ ;\ R \subseteq R\ ;\ dom\ Q)$
by *simp*
finally show *?thesis***.**
qed

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

lemma *fdia-demod-top*: $(\ |R\rangle\ P \subseteq dom\ Q) = (R\ ;\ dom\ P\ ;\ top \subseteq dom\ Q\ ;\ top)$
by (*metis dom-def dom-gla-top rel-fdia-def top-dom*)

lemma *bbox-demod-top*: $(dom\ P \subseteq [R|\ Q) = (R\ ;\ dom\ P\ ;\ top \subseteq dom\ Q\ ;\ top)$
unfolding *rel-bbox-def rel-fbox-def dom-def top-def* **by** *force*

lemma *fdia-bbox-galois*: $(\ |R\rangle\ P \subseteq dom\ Q) = (dom\ P \subseteq [R|\ Q)$
by (*meson bbox-demod-top fdia-demod-top*)

lemma *bdia-fbox-galois*: $(\langle R|\ P \subseteq dom\ Q) = (dom\ P \subseteq |R]\ Q)$
by (*simp add*: *fdia-bbox-galois rel-bbox-def rel-bdia-def*)

lemma *fdia-bdia-conjugation*: $(\ |R\rangle\ P \subseteq neg\ (dom\ Q)) = (\langle R|\ Q \subseteq neg\ (dom\ P))$
unfolding *rel-fdia-set rel-bdia-set dom-set* **by** *force*

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

1.4 Residuation

definition *lres* :: $('a,'c)\ rel \Rightarrow ('b,'c)\ rel \Rightarrow ('a,'b)\ rel$ (**infixl** // *75*)
where $R\ /\!/\ S = \{(a,b).\ \forall c.\ (b,c) \in S \longrightarrow (a,c) \in R\}$

definition *rres* :: $('c,'a)\ rel \Rightarrow ('c,'b)\ rel \Rightarrow ('a,'b)\ rel$ (**infixl** \ *75*)
where $R \setminus S = \{(b,a).\ \forall c.\ (c,b) \in R \longrightarrow (c,a) \in S\}$

lemma *rres-lres-conv*: $R \setminus S = (S^{\smile}\ /\!/\ R^{\smile})^{\smile}$
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*: $R /\!/ S = -(-R ; S^{\smile})$
  unfolding *lres-def converse-unfold* **by** *force*

lemma *rres-compl*: $R \setminus S = -(R^{\smile} ; -S)$
  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* $R \Longrightarrow (R ; S) /\!/ S = R ; (S /\!/ S)$
  by (*metis* (*no-types*, *lifting*) *O-assoc deterministic-var2 lres-compl*)

lemma *det-rres*: *deterministic* $(R^{\smile}) \Longrightarrow S \setminus (S ; R) = (S \setminus S) ; R$
  by (*simp add*: *converse-relcomp det-lres rres-lres-conv*)

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

lemma *lres-bij*: *bijective* $S \Longrightarrow (R /\!/ T^{\smile}) ; S = R /\!/ (T ; S)^{\smile}$
  unfolding *bijective-def injective-set surjective-set relcomp-unfold cod-def Id-def lres-def converse-unfold* **by** *blast*

lemma *dom-rres-top*: $(dom\ P \subseteq R \setminus (dom\ Q ; top)) = (dom\ P ; top \subseteq R \setminus (dom\ Q ; top))$
  unfolding *dom-def top-def rres-def relcomp-unfold Id-def converse-unfold* **by** *clarsimp blast*

lemma *dom-rres-top-var*: $(dom\ P \subseteq R \setminus (dom\ Q ; top)) = (P ; top \subseteq R \setminus (Q ;$

top))
  by (*metis dom-rres-top dom-top*)

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

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

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

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

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

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

lemma *wlp-prop-var*: $dom\ ((R \setminus (dom\ Q\ ;\ top))) = neg\ (cod\ ((neg\ (dom\ Q));\ R))$
  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 $|R\rangle\ (Id \cap P) \subseteq dom\ ((Id \cap Q)\ ;\ R)$
    by (*metis* (*no-types*, *lifting*) *dom-subid dom-top fdia-demod-top inf.absorb-iff2 inf-le1 relcomp-distrib2 sup.order-iff*)
  hence $|R\rangle\ (Id \cap P) \subseteq (Id \cap Q) \cap dom\ R$
    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*: ($|R\rangle\ (Id \cap P) \subseteq Id \cap Q) = (Id \cap P \subseteq Id \cap -\ cod$ $((Id \cap -Q);\ R))$
  unfolding *rel-fdia-def dom-def cod-def Id-def* **by** *blast*

lemma *dom-demod-var2*: ($|R\rangle\ (Id \cap P) \subseteq Id \cap Q) = (Id \cap P \subseteq R \setminus ((Id \cap Q)$

; R))
  by (*meson dom-demod rres-galois*)

1.5 Symmetric quotient

definition *syq* :: $('c,'a)$ *rel* $\Rightarrow$ $('c,'b)$ *rel* $\Rightarrow$ $('a,'b)$ *rel* (**infixl** $\div$ *75*)
  where $R \div S = (R \setminus S) \cap (R^{\smile} /\!\!/ S^{\smile})$

lemma *syq-set*: $R \div S = \{(a,b).\ \forall\, c.\ (c,a) \in R \longleftrightarrow (c,b) \in S\}$
  unfolding *syq-def relcomp-unfold lres-def rres-def* **by** *force*

lemma *converse-syq* [*simp*]: $(R \div S)^{\smile} = S \div R$
  unfolding *syq-def converse-def rres-def lres-def* **by** *blast*

lemma *syq-compl*: $R \div S = -\ (R^{\smile}\ ;\ -S) \cap -\ (-(R^{\smile})\ ;\ S)$
  by (*simp add*: *lres-compl rres-compl syq-def*)

lemma *syq-compl2* [*simp*]: $-R \div -S = R \div S$
  unfolding *syq-compl* **by** *blast*

lemma *syq-expand1*: $R\ ;\ (R \div S) = S \cap (top\ ;\ (R \div S))$
  unfolding *syq-set top-def relcomp-unfold* **by** *force*

lemma *syq-expand2*: $(R \div S)\ ;\ S^{\smile} = R^{\smile} \cap ((R \div S)\ ;\ top)$
  unfolding *syq-set top-def relcomp-unfold* **by** *force*

lemma *syq-comp1*: $(R \div S)\ ;\ (S \div T) = (R \div T) \cap (top\ ;\ (S \div T))$
  unfolding *syq-set top-def relcomp-unfold* **by** *fastforce*

lemma *syq-comp2*: $(R \div S)\ ;\ (S \div T) = (R \div T) \cap ((R \div S)\ ;\ top)$
  unfolding *syq-set top-def relcomp-unfold* **by** *fastforce*

lemma *syq-bij*: *bijective* $T \Longrightarrow (R \div S)\ ;\ T = R \div (S\ ;\ T)$
  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* (Λ) **where**
  $\Lambda\ R = \{(a,B)\ | a\ B.\ B = \{b.\ (a,b) \in R\}\}$

definition *epsilon* :: $('a,'a\ set)\ rel$ **where**
epsilon = $\{(a,A).\ a \in A\}$

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

definition *alpha* :: $('a,'b\ set)\ rel \Rightarrow ('a,'b)\ rel$ (α) **where**
$\alpha\ R = R\ ;\ epsiloff$

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

lemma *Lambda-syq*: $\Lambda\ R = R^{\smile} \div epsilon$
unfolding *Lambda-def syq-set epsilon-def* **by** *blast*

lemma *epsiloff-epsilon*: $epsiloff = epsilon^{\smile}$
unfolding *epsiloff-def epsilon-def converse-unfold* **by** *simp*

lemma *alpha-set*: $\alpha\ R = \{(a,b)\ |a\ b.\ b \in \bigcup\{B.\ (a,B) \in R\}\}$
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*: $f = \Lambda\ R \Longrightarrow R = \alpha\ f$
unfolding *Lambda-def alpha-set* **by** *simp*

lemma *Lambda-epsiloff-up2*: $deterministic\ f \Longrightarrow R = \alpha\ f \Longrightarrow f = \Lambda\ R$
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*: $deterministic\ f \Longrightarrow \Lambda\ (\alpha\ f) = f$
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 $\alpha\ f = \alpha\ g \Longrightarrow f = g$
by (*metis Lambda-epsiloff-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 $\ldots = f\ ;\ \alpha\ (\Lambda\ R)$
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-epsiloff* [*simp*]: Λ *epsiloff* = *Id*
proof −
have Λ *epsiloff* = Λ (*Id* ; *epsiloff*)
by *simp*
also have ... = *Id*
by (*metis Lambda-epsiloff-up alpha-def det-Id*)
finally show *?thesis***.**
qed

lemma *alpha-epsiloff* [*simp*]: α *Id* = *epsiloff*
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*: $R \subseteq S \Longrightarrow$ *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* $(\mathcal{P})$ **where**
$\mathcal{P}\ R = \Lambda$ (*epsiloff* ; R)

lemma *pow-set*: $\mathcal{P}\ R = \{(A,B).\ B = Image\ R\ A\}$
unfolding *pow-def epsiloff-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*: $\mathcal{P}\ (R^{\smile}) = \{(Q,P).\ P = \{a.\ \exists\, b.\ (a,b) \in R \wedge b \in Q\}\}$
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 $\Lambda\ R\ ;\ \mathcal{P}\ S = \Lambda\ R\ ;\ \Lambda\ (\mathit{epsiloff}\ ;\ S)$
by (*simp add*: *pow-def*)
also have $\ldots = \Lambda\ (\Lambda\ R\ ;\ \mathit{epsiloff}\ ;\ S)$
by (*simp add*: *Lambda-fusion-var O-assoc*)
also have $\ldots = \Lambda\ (R\ ;\ S)$
by (*metis alpha-Lambda-canc alpha-def*)
finally show *?thesis***..**
qed

lemma *pow-func1* [*simp*]: $\mathcal{P}\ \mathit{Id} = \mathit{Id}$
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*]: $\mathit{Grph} \circ \mathit{Image} = \mathcal{P}$
apply (*simp add*: *fun-eq-iff*)
unfolding *pow-def Grph-def Image-def Lambda-def epsiloff-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* (η) **where**
$\eta = \Lambda\ \mathit{Id}$

definition *mu* :: ($'a\ set\ set,\ 'a\ set$) *rel* (μ) **where**
$\mu = \mathit{pow}\ \mathit{epsiloff}$

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

lemma *alpha-eta* [*simp*]: $\alpha\ \eta = \mathit{Id}$
by (*simp add*: *eta-def*)

lemma *det-eta*: *deterministic* η

unfolding *deterministic-set eta-set* **by** *simp*

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

lemma *det-mu*: *deterministic* μ
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 $\ldots = R\ ;\ \Lambda\ Id$
using *Lambda-fusion assms* **by** *blast*
also have $\ldots = R\ ;\ \eta$
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 $\ldots = \Lambda\ (Id\ ;\ R)$
using *Lambda-pow* **by** *blast*
also have $\ldots = \Lambda\ R$
by *simp*
also have $\ldots = R\ ;\ \eta$
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 Λ R ; $\mu = R$; η ; μ
by (*simp add*: *Lambda-eta assms*)
also have ... = R
by (*simp add*: *O-assoc*)
finally show *?thesis*.
qed

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

lemma *lambda-alpha-mu*: Λ (α R) = Λ R ; μ
by (*simp add*: *Lambda-pow alpha-def mu-def*)

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

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

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

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

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

lemma *alpha-eta-id* [*simp*]: α (R ; η) = R
by *simp*

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

lemma *lambda-eta-alpha* [*simp*]: Λ (α (α R ; η)) = Λ (α R)

by *simp*

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

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

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

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

2.4 Subset relation

definition *Omega* :: ($'a\ set$, $'a\ set$) *rel* (Ω) **where**
Ω = *epsilon* $\backslash$ *epsilon*

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

lemma *conv-Omega*: *Omega*$^\smile$ = *epsiloff* // *epsiloff*
by (*simp add*: *Omega-def epsiloff-epsilon rres-lres-conv*)

lemma *epsilon-eta-Omega* [*simp*]: η ; Ω = *epsilon*
unfolding *eta-set Omega-set epsilon-def* **by** *force*

lemma *epsiloff-eta-Omega* [*simp*]: $\Omega^\smile$; $\eta^\smile$ = *epsiloff*
by (*metis converse-relcomp epsiloff-epsilon epsilon-eta-Omega*)

lemma *epsilon-Omega* [*simp*]: *epsilon* ; Ω = *epsilon*
by (*simp add*: *Omega-def*)

lemma *conv-Omega-epsiloff* [*simp*]: $\Omega^\smile$; *epsiloff* = *epsiloff*
by (*simp add*: *conv-Omega*)

lemma *Lambda-conv* [*simp*]: $(\Lambda\ R)^\smile$ = *epsilon* $\div$ $R^\smile$
by (*simp add*: *Lambda-syq*)

lemma *Lambda-Omega*: $\Lambda\ R\ ;\ \Omega = R^\smile$ $\backslash$ *epsilon*
proof −
have $\Lambda\ R\ ;\ \Omega = \Lambda\ R$; (*epsilon* $\backslash$ *epsilon*)
by (*simp add*: *Omega-def*)
also have ... = $\Lambda\ R$; −(*epsiloff* ; −*epsilon*)
by (*simp add*: *epsiloff-epsilon rres-compl*)
also have ... = −($\Lambda\ R$; *epsiloff* ; −*epsilon*)
by (*metis O-assoc det-lambda deterministic-var2*)
also have ... = − (R ; −*epsilon*)

by (*metis alpha-Lambda-canc alpha-def*)
also have ... = $R^\smile$ \ *epsilon*
by (*simp add*: *rres-compl*)
finally show *?thesis*.
qed

lemma *syq-epsiloff-prop* [*simp*]: $\Omega^\smile$; (*epsilon* ÷ *R*) = *epsiloff* // $R^\smile$
by (*metis Lambda-Omega Lambda-syq converse-converse converse-relcomp converse-syq epsiloff-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* ($\mathcal{C}$) **where**
$\mathcal{C}$ = *epsilon* ÷ −*epsilon*

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

lemma *Compl-Compl* [*simp*]: $\mathcal{C}$; $\mathcal{C}$ = *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*: $\mathcal{C} = \Lambda$ (−*epsiloff*)
by (*metis Compl-def Lambda-syq boolean-algebra-class.boolean-algebra.double-compl compl-conv converse-converse epsiloff-epsilon syq-compl2*)

lemma *converse-Compl* [*simp*]: $\mathcal{C}^\smile = \mathcal{C}$
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-epsiloff* [*simp*]: $\mathcal{C}$; −*epsiloff* = *epsiloff*
by (*metis Compl-Compl Compl-def-var alpha-Lambda-canc alpha-epsiloff alpha-relcomp*)

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

lemma *compl-epsilon-Compl* [*simp*]: −*epsilon* ; $\mathcal{C}$ = *epsilon*
by (*metis Compl-compl-epsiloff compl-conv converse-Compl converse-converse*

converse-relcomp epsiloff-epsilon)

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

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

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

2.6 Kleisli lifting and Kleisli composition

definition *klift* :: $('a,'b\ set)\ rel \Rightarrow ('a\ set,'b\ set)\ rel$ ($\text{-}_{\mathcal{P}}$ [*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*: $(R)_{\mathcal{P}} = \Lambda\ (epsiloff \; ; R \; ; epsiloff)$
by (*simp add*: *pow-def O-assoc alpha-def klift-def*)

lemma *klift-set*: $(R)_{\mathcal{P}} = \{(A,B).\ B = \bigcup (Image\ R\ A)\}$
  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 $(R)_{\mathcal{P}} = \mathcal{P}\ (R \; ; epsiloff)$
    by (*simp add*: *alpha-def klift-def*)
  also have $\ldots = \mathcal{P}\ R \; ; \mathcal{P}\ epsiloff$
    by (*simp add*: *pow-func2*)
  also have $\ldots = \mathcal{P}\ R \; ; \mu$
    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* $R \Longrightarrow \eta\ ;\ (R)_{\mathcal{P}} = R$
  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 $\ldots = R\ ;\ ((S)_{\mathcal{P}}\ ;\ (T)_{\mathcal{P}})$
    by (*simp add*: *klift-ext1*)
  also have $\ldots = (R \cdot_{\mathcal{P}} S) \cdot_{\mathcal{P}} T$
    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* $R \Longrightarrow \eta \cdot_{\mathcal{P}} R = R$
  by (*simp add*: *kcomp-def klift-ext2*)

2.7 Relational box

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

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

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

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

lemma *rbox-semicomm*: $(Q,P) \in rbox\ R\ ;\ \Omega^{\smile} = (\Delta\ P\ ;\ R \subseteq R\ ;\ \Delta\ Q)$
unfolding *rbox-subset Delta-def* **by** *blast*

lemma *rbox-semicomm-var*: $(Q,P) \in rbox\ R\ ;\ \Omega^{\smile} = (\Delta\ P \subseteq (R\ ;\ \Delta\ Q)\ /\!/\ R)$
by (*simp add*: *lres-galois rbox-semicomm*)

lemma *rbox-omega*: $rbox\ epsiloff = \Lambda\ (\Omega^{\smile})$
by (*simp add*: *conv-Omega rbox-def*)

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

lemma *pow-rbox*: $((Q,P) \in rbox\ R\ ;\ \Omega^{\smile}) = ((P,Q) \in \mathcal{P}\ R\ ;\ \Omega)$
proof −
have $(Q,P) \in rbox\ R\ ;\ \Omega^{\smile} = (\Delta\ P\ ;\ R \subseteq R\ ;\ \Delta\ Q)$
by (*simp add*: *rbox-semicomm*)
also have $\ldots = ((P,Q) \in \mathcal{P}\ R\ ;\ \Omega)$
by (*simp add*: *pow-semicomm*)
finally show *?thesis*.
qed

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

lemma *pow-rbox-Compl*: $\mathcal{P}\ R = \mathcal{C}\ ;\ rbox\ (R^{\smile})\ ;\ \mathcal{C}$
by (*metis Compl-Compl Id-O-R O-assoc R-O-Id converse-converse rbox-pow-Compl*)

lemma *pow-conjugation*: $\mathcal{C}$; $(\mathcal{P}\ (R^{\smile})\ ;\ \Omega)^{\smile} = \mathcal{P}\ R\ ;\ \mathcal{C}\ ;\ \Omega^{\smile}$
proof −
have $\mathcal{P}\ R\ ;\ \mathcal{C}\ ;\ \Omega^{\smile} = \Lambda\ (-(epsiloff\ ;\ R))\ ;\ -(-\ epsiloff\ ;\ epsilon)$
by (*simp add*: *Lambda-Compl pow-def conv-Omega epsiloff-epsilon lres-compl*)
also have $\ldots = -(\Lambda\ (-(epsiloff\ ;\ R))\ ;\ -\ epsiloff\ ;\ epsilon)$
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 $\ldots = -(\Lambda\ (-(epsiloff\ ;\ R))\ ;\ \mathcal{C}\ ;\ epsiloff\ ;\ epsilon)$
by (*metis Compl-epsiloff alpha-def alpha-relcomp*)
also have $\ldots = -(\Lambda\ (epsiloff\ ;\ R)\ ;\ epsiloff\ ;\ epsilon)$
by (*simp add*: *Lambda-Compl*)
also have $\ldots = -(epsiloff\ ;\ R\ ;\ epsilon)$
by (*metis Lambda-epsiloff-up1 alpha-def*)
also have $\ldots = -(\mathcal{C}\ ;\ -epsiloff\ ;\ (epsiloff\ ;\ R^{\smile})^{\smile})$
by (*metis Compl-compl-epsiloff O-assoc converse-converse converse-relcomp epsiloff-epsilon*)
also have $\ldots = \mathcal{C}\ ;\ (epsiloff\ /\!\!/\ (epsiloff\ ;\ R^{\smile}))$
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 $\ldots = \mathcal{C};\ (\mathcal{P}\ (R^{\smile})\ ;\ \Omega)^{\smile}$
by (*simp add*: *Lambda-Omega pow-def epsiloff-epsilon rres-lres-conv*)
finally show *?thesis*..
qed

lemma *pow-rbox-eq*: $rbox\ R\ ;\ \Omega^{\smile} = (\mathcal{P}\ R\ ;\ \Omega)^{\smile}$
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** $\cdot$ *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* (1_σ) **where**
$1_\sigma = (\bigcup a.\ \{(a,\{a\})\})$

definition *p-prod* :: $('a,'b)$ *mrel* $\Rightarrow$ $('a,'b)$ *mrel* $\Rightarrow$ $('a,'b)$ *mrel* (**infixl** $\|$ *70*)
where
$R \| S = \{(a,A).\ (\exists B\ C.\ A = B \cup C \wedge (a,B) \in R \wedge (a,C) \in S)\}$

definition *p-id* :: $('a,'b)$ *mrel* (1_π) **where**
$1_\pi = (\bigcup a.\ \{(a,\{\})\})$

definition *U* :: $('a,'b)$ *mrel* **where**
$U = \{(a,A)\ |a\ A.\ True\}$

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*]: $1_\pi \| R = R$
by (*simp add*: *mr-simp*)

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

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

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

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

lemma *subidem-par*: $R \subseteq R \| R$
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*: $X \neq \{\} \Longrightarrow (\bigcup S \in X.\ R \cdot S) \subseteq R \cdot \bigcup X$
by (*simp add*: *mr-simp*) *blast*

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

lemma *s-prod-isor*: $R \subseteq S \Longrightarrow T \cdot R \subseteq T \cdot S$
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 (*clarsimp 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*: $R \subseteq S \Longrightarrow R \parallel T \subseteq S \parallel T$
by (*metis p-prod-comm p-prod-distl sup.orderE sup.orderI*)

lemma *p-prod-isor*: $R \subseteq S \Longrightarrow T \parallel R \subseteq T \parallel S$

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 $\exists B\ C.\ A = B \cup C \wedge (\exists D.\ (a,D) \in R \wedge (\exists f.\ (\forall d \in D.\ (d,f\ d) \in T) \wedge B = \bigcup\ ((\lambda x.\ f\ x)\ `\ D))) \wedge (\exists E.\ (a,E) \in S \wedge (\exists g.\ (\forall e \in E.\ (e,g\ e) \in T) \wedge C = \bigcup\ ((\lambda x.\ g\ x)`\ E)))$
by (*simp add*: *mr-simp*)
hence $\exists D\ E.\ (a,D \cup E) \in R \parallel S \wedge (\exists f\ g.\ (\forall d \in D.\ (d,f\ d) \in T) \wedge (\forall e \in E.\ (e,g\ e) \in T) \wedge A = (\bigcup\ ((\lambda x.\ f\ x)\ `\ D)) \cup (\bigcup\ ((\lambda x.\ g\ x)`\ E)))$
by (*auto simp*: *mr-simp*)
hence $\exists D\ E.\ (a,D \cup E) \in R \parallel S \wedge (\exists f\ g.\ (\forall d \in D{-}E.\ (d,f\ d) \in T) \wedge (\forall e \in E{-}D.\ (e,g\ e) \in T) \wedge (\forall x \in D \cap E.\ (x,f\ x) \in T \wedge (x,g\ x) \in T) \wedge A = (\bigcup\ ((\lambda x.\ f\ x)\ `\ (D{-}E))) \cup (\bigcup\ ((\lambda x.\ g\ x)\ `\ (E{-}D)) \cup (\bigcup\ ((\lambda y.\ f\ y \cup g\ y)\ `\ (D \cap E)))))$
by *auto blast*
hence $\exists D\ E.\ (a,D \cup E) \in R \parallel S \wedge (\exists f\ g.\ (\forall d \in D{-}E.\ (d,f\ d) \in T) \wedge (\forall e \in E{-}D.\ (e,g\ e) \in T) \wedge (\forall x \in D \cap E.\ (x,f\ x \cup g\ x) \in T) \wedge A = (\bigcup\ ((\lambda x.\ f\ x)\ `\ (D{-}E))) \cup (\bigcup\ ((\lambda x.\ g\ x)\ `\ (E{-}D)) \cup (\bigcup\ ((\lambda y.\ f\ y \cup g\ y)\ `\ (D \cap E)))))$
apply *clarify*
subgoal for *D E f g*
apply (*rule exI*[*of - D*])
apply (*rule exI*[*of - E*])
apply *clarify*

apply (*rule exI*[*of* - *f*])
apply (*rule exI*[*of* - *g*])
using *assms* **by** (*auto simp*: *p-prod-def*, *blast*)
done
hence $\exists D\ E.\ (a,D \cup E) \in R \parallel S \wedge (\exists h.\ (\forall d \in D{-}E.\ (d,h\ d) \in T) \wedge (\forall e \in E{-}D.\ (e,\ h\ e) \in T) \wedge (\forall x \in D \cap E.\ (x,\ h\ x) \in T) \wedge A = (\bigcup\ ((\lambda x.\ h\ x)\ \text{`}\ (D{-}E))) \cup (\bigcup\ ((\lambda x.\ h\ x)\ \text{`}\ (E{-}D)) \cup (\bigcup\ ((\lambda y.\ h\ y)\ \text{`}\ (D \cap E)))))$
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 $(\exists B.\ (a,B) \in R \parallel S \wedge (\exists h.\ (\forall b{\in}B.\ (b,h\ b) \in T) \wedge A = \bigcup((\lambda x.\ h\ x)\ \text{`}\ B)))$
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 $\exists B.\ (a,B) \in R \wedge (\exists f.\ (\forall b \in B.\ \exists C\ D.\ f\ b = C \cup D \wedge (b,C) \in S \wedge (b,D) \in T) \wedge A = \bigcup((\lambda x.\ f\ x)\ \text{`}\ B))$
by (*clarsimp simp*: *mr-simp*)
hence $\exists B.\ (a,B) \in R \wedge (\exists f\ g\ h.\ (\forall b \in B.\ f\ b = g\ b \cup h\ b \wedge (b,g\ b) \in S \wedge (b,h\ b) \in T) \wedge A = \bigcup((\lambda x.\ f\ x)\ \text{`}\ B))$
by (*clarsimp simp*: *bchoice*, *metis*)
hence $\exists B.\ (a,B) \in R \wedge (\exists g\ h.\ (\forall b \in B.\ (b,g\ b) \in S \wedge (b,h\ b) \in T) \wedge A = (\bigcup\ ((\lambda x.\ g\ x)\ \text{`}\ B)) \cup (\bigcup\ ((\lambda x.\ h\ x)\ \text{`}\ B)))$
by *blast*
hence $\exists C\ D.\ (\exists B.\ (a,B) \in R \wedge (\exists g.\ (\forall b \in B.\ (b,g\ b) \in S) \wedge C = \bigcup\ ((\lambda x.\ g\ x)\ \text{`}\ B))) \wedge (\exists B.\ (a,B) \in R \wedge (\exists h.\ (\forall b \in B.\ (b,h\ b) \in T) \wedge D = \bigcup\ ((\lambda x.\ h\ x)\text{`}\ B))) \wedge A = C \cup D$
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 $(a,A) \in R \cdot (P \cap 1_\sigma)$
shows $((a,A) \in R \wedge (\forall a \in A.\ (a,\{a\}) \in (P \cap 1_\sigma)))$
using *assms* **by** (*auto simp*: *mr-simp*)

lemma *s-prod-test-aux2*:
  assumes $(a,A) \in R$
  and $\forall\, a \in A.\ (a,\{a\}) \in S$
  shows $(a,A) \in R \cdot S$
  using *assms* **by** (*fastforce simp*: *mr-simp*)

lemma *s-prod-test*: $(a,A) \in R \cdot (P \cap 1_\sigma) \longleftrightarrow (a,A) \in R \wedge (\forall\, a \in A.\ (a,\{a\}) \in (P \cap 1_\sigma))$
  by (*meson s-prod-test-aux1 s-prod-test-aux2*)

lemma *s-prod-test-var*: $R \cdot (P \cap 1_\sigma) = \{(a,A).\ (a,A) \in R \wedge (\forall\, a \in A.\ (a,\{a\}) \in (P \cap 1_\sigma))\}$
  apply (*rule antisym*)
  by (*fastforce simp*: *mr-simp*)+

lemma *test-s-prod-aux1*:
  assumes $(a,A) \in (P \cap 1_\sigma) \cdot R$
  shows $(a,\{a\}) \in (P \cap 1_\sigma) \wedge (a,A) \in R$
  using *assms* **by** (*auto simp*: *mr-simp*)

lemma *test-s-prod-aux2*:
  assumes $(a,A) \in R$
  and $(a,\{a\}) \in P$
  shows $(a,A) \in P \cdot R$
  using *assms s-prod-def* **by** *fastforce*

lemma *test-s-prod*: $(a,A) \in (P \cap 1_\sigma) \cdot R \longleftrightarrow (a,\{a\}) \in (P \cap 1_\sigma) \wedge (a,A) \in R$
  by (*meson test-s-prod-aux1 test-s-prod-aux2*)

lemma *test-s-prod-var*: $(P \cap 1_\sigma) \cdot R = \{(a,A).\ (a,\{a\}) \in (P \cap 1_\sigma) \wedge (a,A) \in R\}$
  by (*simp add*: *set-eq-iff test-s-prod*)

lemma *test-assoc1*: $(R \cdot (P \cap 1_\sigma)) \cdot S = R \cdot ((P \cap 1_\sigma) \cdot S)$
  apply (*rule antisym*)
  apply (*simp add*: *s-prod-assoc1*)
  apply (*clarsimp simp*: *mr-simp*)
  by (*metis UN-singleton*)

lemma *test-assoc2*: $((P \cap 1_\sigma) \cdot R) \cdot S = (P \cap 1_\sigma) \cdot (R \cdot S)$
  apply (*rule antisym*)
  apply (*simp add*: *s-prod-assoc1*)
  by (*fastforce simp*: *mr-simp s-prod-assoc1*)

lemma *test-assoc3*: $(R \cdot S) \cdot (P \cap 1_\sigma) = R \cdot (S \cdot (P \cap 1_\sigma))$
proof (*rule antisym*)
  show $(R \cdot S) \cdot (P \cap 1_\sigma) \subseteq R \cdot (S \cdot (P \cap 1_\sigma))$
    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-subdistrl*: $(P \cap 1_\sigma) \cdot (S \parallel T) = ((P \cap 1_\sigma) \cdot S) \parallel ((P \cap 1_\sigma) \cdot T)$
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-multipliciativer*: $(P \cap Q \cap 1_\sigma) \cdot T = ((P \cap 1_\sigma) \cdot T) \cap ((Q \cap 1_\sigma) \cdot T)$
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 $\exists B.\ (a,B) \in R \wedge (\exists f.\ (\forall b \in B.\ (\exists\ C.\ (b,C) \in S \wedge (\exists g.\ (\forall x \in C.\ (x,g\ x) \in T \cdot \{\}) \wedge f\ b = \bigcup((\lambda x.\ g\ x)\ `\ C)))) \wedge A = \bigcup((\lambda x.\ f\ x)\ `\ B))$
by (*clarsimp simp*: *mr-simp*)
hence $\exists B.\ (a,B) \in R \wedge (\exists f.\ (\forall b \in B.\ (\exists\ C.\ (b,C) \in S \wedge (\forall x \in C.\ (x,\{\}) \in T \cdot \{\}) \wedge f\ b = \{\})) \wedge A = \bigcup((\lambda x.\ f\ x)\ `\ B))$
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 $\exists B.\ (a,B) \in R \wedge (\exists f.\ (\forall b \in B.\ (b,f\ b) \in S) \wedge (\forall x \in \bigcup((\lambda x.\ f\ x)\ `\ B).\ (x,\{\}) \in T \cdot \{\})) \wedge A = \{\}$
by (*metis UN-E*)
hence $\exists C\ B.\ (a,B) \in R \wedge (\exists f.\ (\forall b \in B.\ (b,\ f\ b) \in S) \wedge C = \bigcup((\lambda x.\ f\ x)\ `\ B) \wedge (\forall x \in C.\ (x,\{\}) \in T \cdot \{\})) \wedge A = \{\}$
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 $\ldots = R \cdot \{(a,\{\}) \mid a.\ \exists B.\ (a,B) \in S\}$
apply (*clarsimp simp*: *mr-simp*)
apply *safe*
apply *blast*
apply *clarsimp*
by *metis*
also have $\ldots = R \cdot (S \cdot 1_\pi)$
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*: $T \cdot \{\} \subseteq R \cdot \{\} \Longrightarrow T \cdot \{\} \subseteq S \cdot \{\} \Longrightarrow T \cdot \{\} \subseteq (R \cdot \{\}) \parallel (S \cdot \{\})$
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*: $(T{::}('a,'b)\ mrel) \cdot 1_\pi \subseteq (R{::}('a,'b)\ mrel) \cdot 1_\pi \Longrightarrow T \cdot 1_\pi \subseteq (S{::}('a,'b)\ mrel) \cdot 1_\pi \Longrightarrow T \cdot 1_\pi \subseteq (R \cdot 1_\pi) \parallel (S \cdot 1_\pi)$
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 (*clarsimp 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 $\ldots = (R \parallel S) \cdot \{(a,\{\}) \mid a.\ \exists B.\ (a,B) \in T\}$
apply (*rule antisym*)
apply (*metis cl4 dual-order.refl p-id-st x-c-par-idem*)
by (*simp add*: *seq-conc-subdistr*)
also have $\ldots = (R \parallel S) \cdot T \cdot 1_\pi$
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 (*clarsimp simp*: *mr-simp*)
apply (*metis UN-constant*)
apply (*clarsimp 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* $R \Longrightarrow$ *univalent* $S \Longrightarrow$ *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* $R \Longrightarrow$ *deterministic* $S \Longrightarrow$ *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* 1_σ
unfolding *deterministic-set s-id-def* **by** *simp*

3.4 Domain

definition *Dom* :: $('a,'b)$ *mrel* $\Rightarrow$ $('a,'a)$ *mrel* **where**
Dom R $= \{(a,\{a\}) \mid a.\ \exists B.\ (a,B) \in R\}$

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

lemma *d-def-expl*: *Dom R* $= R \cdot 1_\pi \parallel 1_\sigma$
by (*force simp*: *mrd-simp set-eqI*)

lemma *s-subid-iff2*: $(R \cap 1_\sigma = R) = (Dom\ R = R)$
by (*metis c6 cl9 d-def-expl inf.order-iff p-prod-comm p-prod-ild p-prod-isor*)

lemma *cl8-var*: *Dom R* $\cdot S = R \cdot 1_\pi \parallel S$
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*]: $R \cdot 1_\pi \parallel 1_\sigma \cdot S = R \cdot 1_\pi \parallel S$
by *simp*

lemma *cl10-var*: *Dom* $(R - 1_\pi) = 1_\sigma \cap ((R - 1_\pi) \cdot NC)$
apply (*rule set-eqI*, *clarsimp simp*: *mrd-simp*)
apply *safe*
apply (*metis SUP-constant insert-not-empty*)
by *blast*

lemma *c10*: $(R \cap NC) \cdot 1_\pi \parallel 1_\sigma = 1_\sigma \cap ((R \cap NC) \cdot NC)$
by (*metis Int-Diff cl10-var d-def-expl*)

lemma *cl9-var* [*simp*]: *Dom* $(R \cap 1_\sigma) = R \cap 1_\sigma$
by (*simp add*: *d-def-expl*)

lemma *d-s-id* [*simp*]: *Dom R* $\cap 1_\sigma =$ *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* $\subseteq 1_\sigma$
by (*simp add*: *le-iff-inf*)

lemma *d-assoc1*: *Dom R* $\cdot (S \cdot T) = (Dom\ R \cdot S) \cdot T$
by (*metis d-s-id test-assoc2*)

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

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

lemma *cd-2-var*: *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*]: *Dom* $R \cdot 1_\pi = R \cdot 1_\pi$
  by (*simp add*: *cl8-var*)

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

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

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

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

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

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

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

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

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

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

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

lemma *d-comm-ax*: *Dom R* · *Dom S* = *Dom S* · *Dom R*
by *force*

lemma *d-s-id-prop* [*simp*]: *Dom* 1_σ = 1_σ
by (*simp add*: *d-def-expl*)

lemma *d-s-prod-closed* [*simp*]: *Dom* (*Dom R* · *Dom S*) = *Dom R* · *Dom S*
using *d-exp-ax d-loc-ax* **by** *blast*

lemma *d-p-prod-closed* [*simp*]: *Dom* (*Dom R* ‖ *Dom S*) = *Dom R* ‖ *Dom S*
using *d-s-prod-closed* **by** *auto*

lemma *d-idem2* [*simp*]: *Dom R* · *Dom R* = *Dom R*
by (*metis d-exp-ax d-rest-ax*)

lemma *d-assoc*: (*Dom R* · *Dom S*) · *Dom T* = *Dom R* · (*Dom S* · *Dom T*)
using *d-assoc1* **by** *blast*

lemma *iso-1* [*simp*]: *Dom R* · 1_π ‖ 1_σ = *Dom R*
using *d-def-expl* **by** *force*

lemma *d-idem-par* [*simp*]: *Dom R* ‖ *Dom R* = *Dom R*
by (*simp add*: *d-conc-s-prod-ax*)

lemma *d-inter-r*: *Dom R* · (*S* ‖ *T*) = *Dom R* · *S* ‖ *Dom R* · *T*
by (*metis d-s-id seq-conc-subdistrl*)

lemma *d-add-ax*: *Dom* (*R* ∪ *S*) = *Dom R* ∪ *Dom S*
by (*simp add*: *d-def-expl p-prod-comm p-prod-distl s-prod-distr*)

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

lemma *d-distl*: *Dom R* · (*S* ∪ *T*) = *Dom R* · *S* ∪ *Dom R* · *T*
by (*metis d-s-id s-distl-test*)

lemma *d-sup-distl*: *Dom R* · $\bigcup$*X* = ($\bigcup$*S* ∈ *X*. *Dom R* · *S*)
by (*metis d-s-id s-distl-sup-test*)

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

lemma *d-absorb1* [*simp*]: *Dom R* ∪ *Dom R* · *Dom S* = *Dom R*
by *simp*

lemma *d-absorb2* [*simp*]: *Dom R* · (*Dom R* ∪ *Dom S*) = *Dom R*
by (*metis d-absorb1 d-idem2 d-s-id s-distl-test*)

lemma *d-dist1*: *Dom R* · (*Dom S* ∪ *Dom T*) = *Dom R* · *Dom S* ∪ *Dom R* ·

Dom T
  by (*simp add*: *cl8-var p-prod-distl*)

lemma *d-dist2*: *Dom R* ∪ (*Dom S* · *Dom T*) = (*Dom R* ∪ *Dom S*) · (*Dom R* ∪ *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*]: *Dom* (*Dom R* ∪ *Dom S*) = *Dom R* ∪ *Dom S*
  by (*simp add*: *d-add-ax*)

lemma *x-zero-prop*: *R* · {} ‖ *S* = *Dom* (*R* · {}) · *S*
  by (*simp add*: *cl8-var*)

lemma *cda-add-ax*: *Dom* ((*R* ∪ *S*) · *T*) = *Dom* (*R* · *T*) ∪ *Dom* (*S* · *T*)
  by (*simp add*: *d-add-ax s-prod-distr*)

lemma *d-x-zero*: *Dom* (*R* · {}) = *R* · {} ‖ 1_σ
  by (*simp add*: *d-def-expl*)

lemma *cda-ax2*:
  assumes (*R* ‖ *S*) · *Dom T* = *R* · *Dom T* ‖ *S* · *Dom T*
  shows *Dom* ((*R* ‖ *S*) · *T*) = *Dom* (*R* · *T*) · *Dom* (*S* · *T*)
  by (*metis assms d-conc6 d-conc-s-prod-ax d-loc-ax*)

lemma *d-lb1*: *Dom R* · *Dom S* ⊆ *Dom R*
  using *d-absorb1* **by** *blast*

lemma *d-lb2*: *Dom R* · *Dom S* ⊆ *Dom S*
  using *d-comm-ax d-lb1* **by** *blast*

lemma *d-glb*: *Dom T* ⊆ *Dom R* ∧ *Dom T* ⊆ *Dom S* ⟹ *Dom T* ⊆ *Dom R* · *Dom S*
  by *simp*

lemma *d-glb-iff*: *Dom T* ⊆ *Dom R* ∧ *Dom T* ⊆ *Dom S* ⟷ *Dom T* ⊆ *Dom R* · *Dom S*
  by *force*

lemma *d-interr*: *R* · *Dom P* ‖ *S* · *Dom P* = (*R* ‖ *S*) · *Dom P*
  by (*simp add*: *cl4 seq-conc-subdistr subset-antisym*)

lemma *cl10-d*: *Dom* (*R* ∩ *NC*) = 1_σ ∩ (*R* ∩ *NC*) · *NC*
  by (*simp add*: *c10 d-def-expl*)

lemma *cl11-d* [*simp*]: *Dom* (*R* ∩ *NC*) · *NC* = (*R* ∩ *NC*) · *NC*
  by (*simp add*: *cl8-var*)

lemma *cl10-d-var1*: *Dom* (*R* ∩ *NC*) = 1_σ ∩ *R* · *NC*
  by (*simp add*: *cl10-d x-y-prop*)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *x-c-U-split-d*: *Dom R* · *U* = *R* · *U* ∪ (*R* · {} ∥ *U*)
  by (*simp add*: *cl8-var*)

lemma *x-U-prop2*: *R* · *NC* = *Dom* (*R* ∩ *NC*) · *NC* ∪ *R* · {}
  by *simp*

lemma *x-U-prop3*: *R* · *U* = *Dom* (*R* ∩ *NC*) · *U* ∪ *R* · {}
  by (*metis ax5-d x-y-split*)

lemma *d-x-nc* [*simp*]: *Dom* (*R* · *NC*) = *Dom R*
  by (*metis d-loc-ax d-nc dc dc-prop2*)

lemma *d-x-U* [*simp*]: *Dom* (*R* · *U*) = *Dom R*
  by (*metis d-U d-loc-ax dc dc-prop2*)

lemma *d-llp1*: *Dom R* ⊆ *Dom S* ⟹ *R* ⊆ *Dom S* · *R*
  by (*metis d-rest-ax s-prod-isol*)

lemma *d-llp2*: *R* ⊆ *Dom S* · *R* ⟹ *Dom R* ⊆ *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*: *Dom* (*R* · *S*) ⊆ *Dom T* ⟹ *R* · *Dom S* ⊆ *Dom T* · *R*
proof −
  assume *h*: *Dom* (*R* · *S*) ⊆ *Dom T*
  have *R* · *Dom S* = *Dom* (*R* · *Dom S*) · (*R* · *Dom S*)
    using *d-rest-ax* **by** *blast*
  also have ... ⊆ *Dom T* · (*R* · *Dom S*)
    by (*metis d-loc-ax h s-prod-distr subset-Un-eq*)
  also have ... ⊆ *Dom T* · *R*
    by (*metis d-s-id-ax s-prod-idr s-prod-isor*)
  finally show *R* · *Dom S* ⊆ *Dom T* · *R*.
qed

lemma *demod2*: *R* · *Dom S* ⊆ *Dom T* · *R* ⟹ *Dom* (*R* · *S*) ⊆ *Dom T*
proof −
  assume *h*: *R* · *Dom S* ⊆ *Dom T* · *R*
  have *Dom* (*R* · *S*) = *Dom* (*R* · *Dom S*)
    by *auto*
  also have ... ⊆ *Dom* (*Dom T* · *R*)
    by (*metis d-add-ax h subset-Un-eq*)
  also have ... = *Dom T* · *Dom R*
    by *simp*
  also have ... ⊆ *Dom T*
    by (*simp add*: *d-lb1*)
  finally show *Dom* (*R* · *S*) ⊆ *Dom T*.
qed

lemma *d-meet-closed* [*simp*]: *Dom* (*Dom x* ∩ *Dom y*) = *Dom x* ∩ *Dom y*

by (*metis d-s-id inf-assoc s-subid-iff2*)

lemma *d-add-var*: *Dom P* · *R* ∪ *Dom Q* · *R* = *Dom* (*P* ∪ *Q*) · *R*
by (*simp add*: *d-add-ax s-prod-distr*)

lemma *d-interr-U*: *Dom x* · *U* ∥ *Dom y* · *U* = *Dom* (*x* ∥ *y*) · *U*
by (*metis U-nc U-par-idem cl4 d-conc6 seq-conc-subdistr subset-antisym*)

lemma *d-meet*: *Dom x* · *z* ∩ *Dom y* · *z* = (*Dom x* ∩ *Dom y*) · *z*
by (*simp add*: *d-meet-distr-var*)

lemma *cs-hom-meet*: *Dom* (*x* · 1_π ∩ *y* · 1_π) = *Dom* (*x* · 1_π) ∩ *Dom* (*y* · 1_π)
by (*metis d-conc6 d-conc-inter dc-prop2 p-subid-par-eq-meet-var*)

lemma *iso3* [*simp*]: *Dom* (*Dom x* · *U*) = *Dom x*
by *simp*

lemma *iso4* [*simp*]: *Dom* (*x* · 1_π ∥ *U*) · *U* = *x* · 1_π ∥ *U*
by (*metis cl8-var iso3*)

lemma *iso3-sharp* [*simp*]: *Dom* (*Dom* (*x* ∩ *NC*) · *NC*) = *Dom* (*x* ∩ *NC*)
by *simp*

lemma *iso4-sharp* [*simp*]: *Dom* ((*x* ∩ *NC*) · *NC*) · *NC* = (*x* ∩ *NC*) · *NC*
by *simp*

3.5 Vectors

lemma *vec-iff1*:
assumes ∀ *a*. (∃ *A*. (*a*,*A*) ∈ *R*) ⟶ (∀ *A*. (*a*,*A*) ∈ *R*)
shows *R* · 1_π ∥ *U* = *R*
using *assms* **by** (*auto simp*: *mr-simp*)

lemma *vec-iff2*:
assumes *R* · 1_π ∥ *U* = *R*
shows (∀ *a*. (∃ *A*. (*a*,*A*) ∈ *R*) ⟶ (∀ *A*. (*a*,*A*) ∈ *R*))
using *assms* **apply** (*clarsimp simp*: *mr-simp*)
by (*smt* (*z3*) *SUP-bot case-prod-conv mem-Collect-eq sup-bot-left*)

lemma *vec-iff*: (∀ *a*. (∃ *A*. (*a*,*A*) ∈ *R*) ⟶ (∀ *A*. (*a*,*A*) ∈ *R*)) ⟷ *R* · 1_π ∥ *U* = *R*
by (*metis vec-iff1 vec-iff2*)

lemma *U-par-zero* [*simp*]: {} · *R* ∥ *U* = {}
by (*simp add*: *p-prod-comm*)

lemma *U-par-s-id* [*simp*]: 1_σ · 1_π ∥ *U* = *U*
by *auto*

lemma *U-par-p-id* [*simp*]: 1_π · 1_π ∥ *U* = *U*

by *auto*

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

3.6 Up-closure and Parikh composition

definition *s-prod-pa* :: $('a,'b)$ *mrel* $\Rightarrow$ $('b,'c)$ *mrel* $\Rightarrow$ $('a,'c)$ *mrel* (**infixl** $\otimes$ *75*)
where
$R \otimes S = \{(a,A). (\exists B. (a,B) \in R \wedge (\forall b \in B. (b,A) \in S))\}$

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

lemma *p-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-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-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 *twolem*: $R \otimes (S \parallel U) \subseteq R \cdot S \parallel U$
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*: $Dom\ ((R \cdot 1_\pi \parallel U) \cap (S \cdot 1_\pi \parallel U)) = Dom\ (R \cdot 1_\pi \parallel U) \cap Dom\ (S \cdot 1_\pi \parallel U)$
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* (τ) **where**
$\tau\ R = R \cdot \{\}$

definition *nu* :: $('a,'b)$ *mrel* $\Rightarrow$ $('a,'b)$ *mrel* (ν) **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** (*clarsimp simp*: *mr-simp*)

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

lemma *tarski-prod*: $(\nu\ R \cdot NC) \cdot (\nu\ S \cdot NC) = ($*if* $\nu\ S = \{\}$ *then* $\{\}$ *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* (ν *R*) · *NC*
  by (*simp add*: *a*)
also have ... = ν *R* · *NC*
  by (*simp add*: *nu-def*)
finally have (ν *R* · *NC*) · (ν *S* · *NC*) = ν *R* · *NC***.**
thus *?thesis*
  using *False* **by** *auto*
qed

lemma *nc-prod-aux* [*simp*]: (ν *R* · *NC*) · *NC* = ν *R* · *NC*
apply (*clarsimp simp*: *mr-simp*)
apply *safe*
apply *clarsimp*
apply (*smt* (*verit*) *SUP-bot-conv*(*1*) *ex-in-conv*)
apply *clarsimp*
by (*smt* (*verit*, *best*) *SUP-bot-conv*(*2*) *UNIV-I UN-constant*)

lemma *nc-vec-add-closed*: (ν *R* · *NC* ∪ ν *S* · *NC*) · *NC* = ν *R* · *NC* ∪ ν *S* · *NC*
by (*simp add*: *s-prod-distr*)

lemma *nc-vec-par-is-meet*: ν *R* · *NC* ∥ ν *S* · *NC* = ν *R* · *NC* ∩ ν *S* · *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*: (ν *R* · *NC* ∩ ν *S* · *NC*) · *NC* = ν *R* · *NC* ∩ ν *S* · *NC*
apply (*clarsimp simp*: *nu-def mr-simp*)
apply *safe*
apply (*metis SUP-const UN-I ex-in-conv*)
apply (*clarsimp*, *smt* (*verit*) *SUP-bot-conv*(*2*) *ex-in-conv*)
by (*clarsimp*, *smt* (*verit*, *ccfv-threshold*) *SUP-bot-conv*(*1*) *SUP-const UNIV-I all-not-in-conv*)

lemma *nc-vec-par-closed*: (ν *R* · *NC* ∥ ν *S* · *NC*) · *NC* = ν *R* · *NC* ∥ ν *S* · *NC*
by (*metis U-nc-par nc-prod-aux uc-interr*)

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

lemma *iso5-sharp* [*simp*]: (ν *R* · 1_π ∥ *NC*) · 1_π = ν *R* · 1_π
by (*simp add*: *c3*)

lemma *iso6-sharp* [*simp*]: (ν *R* · *NC* · 1_π) ∥ *NC* = ν *R* · *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*: $R \leq S \Longrightarrow \tau\ R \leq \tau\ S$
by (*simp add*: *inf.order-iff tau-def x-zero-meet-closed*)

lemma *nu-iso*: $R \leq S \Longrightarrow \nu\ R \leq \nu\ S$
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*]: ν *1*$_\pi$ = {}
by (*simp add*: *nu-def*)

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

lemma *nu-nc* [*simp*]: ν *NC* = *NC*
using *nu-def* **by** *auto*

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

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

lemma *tau-add* [*simp*]: τ (*R* ∪ *S*) = τ *R* ∪ τ *S*
by (*simp add*: *tau-def x-zero-add-closed*)

lemma *nu-add* [*simp*]: ν (*R* ∪ *S*) = ν *R* ∪ ν *S*
by (*simp add*: *Int-Un-distrib2 nu-def*)

lemma *tau-meet* [*simp*]: τ (*R* ∩ *S*) = τ *R* ∩ τ *S*
by (*simp add*: *tau-def x-zero-meet-closed*)

lemma *nu-meet* [*simp*]: ν (*R* ∩ *S*) = ν *R* ∩ ν *S*
by (*simp add*: *inf-commute inf-left-commute nu-def*)

lemma *tau-seq*: τ (*R* · *S*) = τ *R* ∪ ν *R* · τ *S*
unfolding *nu-def tau-def*
by (*metis sup-commute x-y-split zero-assoc3*)

lemma *tau-par* [*simp*]: τ (*R* ∥ *S*) = τ *R* ∥ τ *S*
by (*metis U-par-zero tau-def uc-interr*)

lemma *nu-par-aux1*: *R* ∥ τ *S* = *Dom* (τ *S*) · *R*
by (*metis p-prod-comm tau-def x-zero-prop*)

lemma *nu-par-aux3* [*simp*]: ν (ν *R* ∥ τ *S*) = ν *R* ∥ τ *S*
by (*metis nc-ccomp-closed-alt p-prod-comm*)

lemma *nu-par-aux4* [*simp*]: ν (τ *R* ∥ τ *S*) = {}
by (*metis nu-def tau-def tau-par zero-nc*)

lemma *nu-par*: ν (*R* ∥ *S*) = *Dom* (τ *R*) · ν *S* ∪ *Dom* (τ *S*) · ν *R* ∪ (ν *R* ∥ ν *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* · *S* = τ *R* ∪ ν *R* · *S*

by (*metis nu-def sup-commute tau-def x-y-split*)

lemma *pprod-tau-nu*: $R \parallel S = (\nu\ R \parallel \nu\ S) \cup Dom\ (\tau\ R) \cdot \nu\ S \cup Dom\ (\tau\ S) \cdot \nu\ R \cup (\tau\ R \parallel \tau\ S)$
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*: $\nu\ R = R \Longrightarrow \nu\ (R \parallel S) = R \parallel S$
by (*metis nc-ccomp-closed-alt p-prod-comm*)

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *nu-seq-precongr*: $\nu\ R \leq \nu\ S \Longrightarrow \nu\ (R \cdot T) \leq \nu\ (S \cdot T)$
by (*metis nu-iso s-prod-isol sprod-tau-nu-var*)

definition
$tcg\ R\ S = (\tau\ R \leq \tau\ S \wedge \tau\ S \leq \tau\ R)$

definition
  ncg R S = (ν *R* $\leq$ ν *S* $\wedge$ ν *S* $\leq$ ν *R*)

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

lemma *tcg-trans*: *tcg R S* $\Longrightarrow$ *tcg S T* $\Longrightarrow$ *tcg R T*
  by (*meson subset-trans tcg-def*)

lemma *tcg-sym*: *tcg R S* $\Longrightarrow$ *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* $\Longrightarrow$ *ncg S T* $\Longrightarrow$ *ncg R T*
  by (*meson ncg-def order-trans*)

lemma *ncg-sym*: *ncg R S* $\Longrightarrow$ *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* $\Longrightarrow$ τ (*R* $\cup$ *T*) = τ (*S* $\cup$ *T*)
  by *simp*

lemma *tcg-meet*: τ *R* = τ *S* $\Longrightarrow$ τ (*R* $\cap$ *T*) = τ (*S* $\cap$ *T*)
  by *simp*

lemma *tcg-par*: τ *R* = τ *S* $\Longrightarrow$ τ (*R* $\parallel$ *T*) = τ (*S* $\parallel$ *T*)
  by *simp*

lemma *tcg-seql*: τ *R* = τ *S* $\Longrightarrow$ τ (*T* $\cdot$ *R*) = τ (*T* $\cdot$ *S*)
  by *simp*

lemma *ncg-add*: ν *R* = ν *S* $\Longrightarrow$ ν (*R* $\cup$ *T*) = ν (*S* $\cup$ *T*)
  by *simp*

lemma *ncg-meet*: ν *R* = ν *S* $\Longrightarrow$ ν (*R* $\cap$ *T*) = ν (*S* $\cap$ *T*)
  by *simp*

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

3.8 Powers

primrec *p-power* :: ('a,'a) *mrel* ⇒ *nat* ⇒ ('a,'a) *mrel* **where**
  p-power R 0 = 1_σ |
  p-power R (Suc n) = *R* · *p-power R n*

primrec *power-rd* :: ('a,'a) *mrel* ⇒ *nat* ⇒ ('a,'a) *mrel* **where**
  power-rd R 0 = {} |
  power-rd R (Suc n) = 1_σ ∪ *R* · *power-rd R n*

primrec *power-sq* :: ('a,'a) *mrel* ⇒ *nat* ⇒ ('a,'a) *mrel* **where**
  power-sq R 0 = 1_σ |
  power-sq R (Suc n) = 1_σ ∪ *R* · *power-sq R n*

lemma *power-rd-chain*: *power-rd R n* ≤ *power-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-sq R n* ≤ *power-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-power* (1_σ ∪ *R*) *n* ≤ *p-power* (1_σ ∪ *R*) (*n + 1*)
  apply (*induct n*)
  apply *simp*
  by (*simp add*: *s-prod-isor*)

lemma *pow-prop*: *p-power* (1_σ ∪ *R*) (*n + 1*) = 1_σ ∪ *R* · *p-power* (1_σ ∪ *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-rd R n* ≤ *power-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-sq R n* ≤ *power-rd R* (*Suc n*)
  apply (*induct n*)
  apply *simp*
  by (*smt* (*verit, del-insts*) *UnCI UnE power-rd.simps*(*2*) *power-sq.simps*(*2*) *s-prod-subdistl 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* (λ*X*. *S* ∪ *R* · *X*)
  by (*rule monoI*, (*clarsimp simp*: *mr-simp*), *blast*)

lemma *gfp-lfp-prop*: *gfp* (λ*X*. *R* · *X*) ∪ *lfp* (λ*X*. *S* ∪ *R* · *X*) ⊆ *gfp* (λ*X*. *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* (λ*X*. *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 ⋀(*R*::(′*a*,′*a*) *mrel*) (*S*::(′*a*,′*a*) *mrel*) (*T*::(′*a*,′*a*) *mrel*). *R* · (*S* · *T*) = (*R* · *S*) · *T*
  shows *star* (*R*::(′*a*,′*a*) *mrel*) ≤ *star* (ν *R*) · (1_σ ∪ τ *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 $\bigwedge(R::('a,'a)\ mrel).\ star\ R \cdot star\ R \leq star\ R$
shows $star\ (\nu\ (R::('a,'a)\ mrel)) \cdot (1_\sigma \cup \tau\ R) \leq star\ R$
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 $\bigwedge(R::('a,'a)\ mrel).\ star\ R \cdot star\ R \leq star\ R$
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 $star\ (\nu\ (R::('a,'a)\ mrel)) \cdot (1_\sigma \cup \tau\ R) = star\ R$
using *assms nu-star1 nu-star2* **by** *blast*

lemma *tau-star*: $star\ (\tau\ R) = 1_\sigma \cup \tau\ R$
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 $\bigwedge(R::('a,'a)\ mrel).\ star\ R \cdot star\ R \leq star\ R$
shows $\tau\ (star\ (R::('a,'a)\ mrel)) = star\ (\nu\ R) \cdot \tau\ R$
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*: $star\ (\nu\ R) \leq \nu\ (star\ R)$
proof −
have *a*: $1_\sigma \subseteq star\ R$
by (*simp add*: *star-refl*)
have *b*: $(R \cap NC) \cdot (star\ R \cap NC) \subseteq star\ R$
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 $(R \cap NC) \cdot (star\ R \cap NC) \subseteq NC$
by (*metis nc-scomp-closed nu-def*)
thus *?thesis*
by (*metis Un-least a b c le-infI nu-def star-induct*)
qed

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

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

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

lemma *tau-star-nu* [*simp*]: τ (*star* (ν *R*)) = {}
  using *alpha-fp tau-def nu-star-nu* **by** *metis*

lemma *d-star-unfold* [*simp*]: *Dom S* ∪ *Dom* (*R* · *Dom* (*star R* · *S*)) = *Dom* (*star R* · *S*)
proof −
  have *Dom S* ∪ *Dom* (*R* · *Dom* (*star R* · *S*)) = *Dom S* ∪ *Dom* (*R* · (*star R* · *Dom S*))
    by (*metis d-loc-ax*)
  also have ... = *Dom* (1_σ · *Dom S* ∪ (*R* · (*star R* · *Dom S*)))
    by (*simp add*: *d-add-ax*)
  also have ... = *Dom* (1_σ · *Dom S* ∪ (*R* · *star R*) · *Dom S*)
    by (*metis d-comm-ax d-s-id-inter d-s-id-prop s-prod-idl test-assoc3*)
  moreover have ... = *Dom* ((1_σ ∪ *R* · *star R*) · *Dom S*)
    using *s-prod-distr* **by** *metis*
  ultimately show *?thesis*
    by *simp*
qed

lemma *d-star-sim1*:
  assumes ⋀*R S T*. *Dom* (*T*::('*a*,'*b*) *mrel*) ∪ (*R*::('*a*,'*a*) *mrel*) · (*S*::('*a*,'*a*) *mrel*) ≤ *S* ⟹ *star R* · *Dom T* ≤ *S*
  shows (*R*::('*a*,'*a*) *mrel*) · *Dom* (*T*::('*a*,'*b*) *mrel*) ≤ *Dom T* · (*S*::('*a*,'*a*) *mrel*) ⟹ *star R* · *Dom T* ≤ *Dom T* · *star S*
proof −
  fix *R S*::('*a*,'*a*) *mrel* **and** *T* ::('*a*,'*b*) *mrel*
  assume *a*: *R* · *Dom T* ≤ *Dom T* · *S*
  hence (*R* · *Dom T*) · *star S* ≤ (*Dom T* · *S*) · *star S*
    by (*simp add*: *s-prod-isol*)
  hence *b*: *R* · (*Dom T* · *star S*) ≤ *Dom T* · (*S* · *star S*)
    by (*metis d-assoc1 d-s-id-ax inf.order-iff test-assoc1*)
  hence *R* · (*Dom T* · *star S*) ≤ *Dom T* · *star S*
    by (*meson order-trans s-prod-isor star-unfold-part*)
  hence *Dom T* ∪ *R* · (*Dom T* · *star S*) ≤ *Dom T* · *star S*
    by (*metis le-supI s-prod-idr s-prod-isor star-refl*)
  thus *star R* · *Dom T* ≤ *Dom T* · *star S*
    using *assms* **by** *presburger*
qed

lemma *d-star-induct*:
  assumes ⋀*R S T*. *Dom* (*T*::('*a*,'*b*) *mrel*) ∪ (*R*::('*a*,'*a*) *mrel*) · (*S*::('*a*,'*a*) *mrel*) ≤ *S* ⟹ *star R* · *Dom T* ≤ *S*
  shows *Dom* ((*R*::('*a*,'*a*) *mrel*) · (*S*::('*a*,'*a*) *mrel*)) ≤ *Dom S* ⟹ *Dom* (*star R* · *S*) ≤ *Dom S*
  by (*metis assms d-star-sim1 dc-prop2 demod1 demod2*)

3.10 Omega

definition *omega* :: $('a,'a)$ *mrel* ⇒ $('a,'a)$ *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*: $S \leq R \cdot S \Longrightarrow S \leq$ *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*: $R \leq S \Longrightarrow$ *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* $(\nu\ R) \cup \tau\ R \cup \nu\ R \cdot$ *star* $(\nu\ R) \cdot \tau\ R$
using *s-prod-distr s-prod-idl* **by** *blast*
also have ... = $\tau\ R \cup \nu\ R \cdot$ *omega* $(\nu\ R) \cup \nu\ R \cdot ($*star* $(\nu\ R) \cdot \tau\ R)$

by (*simp add*: *cl5 sup-commute tau-def*)
also have ... $\leq \tau\ R \cup \nu\ R \cdot (omega\ (\nu\ R) \cup star\ (\nu\ R) \cdot \tau\ R)$
by (*smt* (*verit*) *Un-subset-iff s-prod-isor sup.cobounded2 sup.coboundedI2 sup-commute*)
also have ... $= R \cdot (omega\ (\nu\ R) \cup star\ (\nu\ R) \cdot \tau\ R)$
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 epsiloff-def p-id-def* **by** *force*

4.1 Peleg lifting

definition *plift* :: $('a,'b)\ mrel \Rightarrow ('a\ set,'b\ set)\ rel\ (\text{-}_*\ [1000]\ 999)$ **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*: $deterministic\ R \Longrightarrow R_* = (R)_{\mathcal{P}}$
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 *pliftext-3* [*simp*]: $\eta_* = Id$
by (*simp add*: *det-eta det-plift-klift*)

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

lemma *d-plift-sub*: $A \subseteq B \Longrightarrow (B,B) \in (Dom\ R)_* \Longrightarrow (A,A) \in (Dom\ R)_*$
  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 $R_* = (Dom\ R)_* \ ; (R)_{\mathcal{P}}$
proof −
  have $\forall A\ B\ .\ (A,B) \in R_* \longleftrightarrow (A,B) \in (Dom\ R)_* \ ; (R)_{\mathcal{P}}$
  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 $(A,B) \in (Dom\ R)_* \ ; (R)_{\mathcal{P}}$
      using *1 d-dom-plift dom-set* **by** *fastforce*
  next
    fix $A\ B$
    assume $(A,B) \in (Dom\ R)_* \ ; (R)_{\mathcal{P}}$
    from *this* **obtain** C **where** *2*: $(A,C) \in (Dom\ R)_* \wedge (C,B) \in (R)_{\mathcal{P}}$
      by *auto*
    from *this* **obtain** f **where** *3*: $(\forall a \in A.\ (a,f(a)) \in Dom\ R) \wedge C = \bigcup(f\ \text{`}\ A)$
      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 $(A,B) \in (R)_{\mathcal{P}}$
using *2* **by** *simp*
thus $(A,B) \in R_*$
unfolding *plift-def klift-set-var*
apply *clarsimp*
apply (*rule exI*[**where** *?x=g*])
using *4* **by** (*smt* (*verit*, *ccfv-SIG*) *Collect-cong assms image-def univalent-set*)
qed
thus $R_* = (Dom\ R)_*\ ;\ (R)_{\mathcal{P}}$
by *force*
qed

lemma *plift-ext1*:
assumes *univalent f*
shows $(R\ ;\ f_*)_* = R_*\ ;\ f_*$
proof −
have $\forall A\ B\ .\ (A,B) \in (R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}})_* \longleftrightarrow (A,B) \in R_*\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}$
proof (*intro allI*, *rule iffI*)
fix *A B*
assume *1*: $(A,B) \in (R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}})_*$
from *this* **obtain** *g* **where** *2*: $(\forall a \in A.\ (a,g(a)) \in R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}) \wedge B = \bigcup(g\ `\ A)$
by (*smt* (*verit*) *Pair-inject case-prodE mem-Collect-eq plift-def*)
hence $\forall a \in A\ .\ \exists C\ D\ .\ (a,C) \in R \wedge (C,D) \in (Dom\ f)_* \wedge (D,g(a)) \in (f)_{\mathcal{P}}$
by (*meson relcompEpair*)
hence $\forall a \in A\ .\ \exists C\ .\ (a,C) \in R \wedge (C,C) \in (Dom\ f)_* \wedge (C,g(a)) \in (f)_{\mathcal{P}}$
using *d-pid-plift* **by** *fastforce*
from *this* **obtain** *h* **where** *3*: $\forall a \in A\ .\ (a,h\ a) \in R \wedge (h\ a,h\ a) \in (Dom\ f)_* \wedge (h\ a,g(a)) \in (f)_{\mathcal{P}}$
by *metis*
let $?h = \bigcup(h\ `\ A)$
have *4*: $(A,?h) \in R_*$
using *3 plift-def* **by** *fastforce*
have *5*: $(?h,?h) \in (Dom\ f)_*$
using *3*
unfolding *plift-def Dom-def*
apply *clarsimp*
by (*metis UN-extend-simps*(*9*) *UN-singleton*)
have $(?h,B) \in (f)_{\mathcal{P}}$
using *2 3*
unfolding *klift-set*
by *auto*
thus $(A,B) \in R_*\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}$
using *4 5* **by** *blast*
next
fix *A B*
assume $(A,B) \in R_*\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}$
from *this* **obtain** *C D* **where** *6*: $(A,C) \in R_* \wedge (C,D) \in (Dom\ f)_* \wedge (D,B)$

$\in (f)_{\mathcal{P}}$
    by *blast*
  hence *7*: $C = D$
    using *d-pid-plift* **by** *auto*
  from *6* **obtain** g **where** *8*: $(\forall a \in A.\ (a,g(a)) \in R) \wedge C = \bigcup(g \text{ ` } A)$
    by (*smt* (*verit*) *Pair-inject case-prodE mem-Collect-eq plift-def*)
  hence *9*: $\forall a \in A.\ (g(a),g(a)) \in (Dom\ f)_*$
    using *6 7* **by** (*metis d-plift-sub UN-iff subsetI*)
  let $?h = \lambda a\ .\ \bigcup(f \text{ `` } g\ a)$
  have $\forall a \in A.\ (g(a),?h\ a) \in (f)_{\mathcal{P}}$
    using *6* **by** (*simp add*: *klift-set*)
  hence *10*: $\forall a \in A.\ (a,?h\ a) \in R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}$
    using *8 9* **by** *blast*
  have $B = \bigcup(f \text{ `` } D)$
    using *6 klift-set* **by** *fastforce*
  hence *11*: $B = (\bigcup a{\in}A\ .\ ?h\ a)$
    using *7 8 image-empty* **by** *blast*
  show $(A,B) \in (R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}})_*$
    apply (*subst plift-def*)
    apply *clarsimp*
    apply (*rule exI*[**where** *?x=?h*])
    using *10 11* **by** *simp*
 qed
 hence $(R\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}})_* = R_*\ ;\ (Dom\ f)_*\ ;\ (f)_{\mathcal{P}}$
  by *force*
 thus *?thesis*
  by (*metis* (*no-types*, *opaque-lifting*) *assms univ-plift-klift O-assoc*)
qed

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

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

lemma *eta-funct*: $R\ ;\ S\ ;\ \eta = (R\ ;\ \eta) \cdot (S\ ;\ \eta)$
proof −
 have $(R\ ;\ \eta) \cdot (S\ ;\ \eta) = R\ ;\ \eta\ ;\ (S\ ;\ \eta)_*$
  by (*simp add*: *pcomp-plift*)
 also have $\ldots = R\ ;\ (\eta \cdot (S\ ;\ \eta))$
  by (*simp add*: *O-assoc pcomp-plift*)
 also have $\ldots = R\ ;\ S\ ;\ \eta$
  by (*simp add*: *O-assoc pcomp-plift*)
 finally show *?thesis***..**
qed

lemma *alpha-funct-det*: *deterministic R* $\Longrightarrow$ *deterministic S* $\Longrightarrow \alpha\ (R \cdot S) = \alpha\ R\ ;\ \alpha\ S$
 by (*metis Lambda-epsiloff-up2 Lambda-funct alpha-Lambda-canc*)

lemma *pcomp-det*: *deterministic* $S \Longrightarrow R \cdot S = R$; $(S)_{\mathcal{P}}$
by (*simp add*: *det-plift-klift pcomp-plift*)

lemma *pcomp-det2*: *deterministic* $R \Longrightarrow$ *deterministic* $S \Longrightarrow (R \cdot S)_{\mathcal{P}} = (R)_{\mathcal{P}}$; $(S)_{\mathcal{P}}$
by (*simp add*: *klift-ext1 pcomp-det*)

lemma *pcomp-alpha*: α $(R \cdot S) = R$; α $((S)_*)$
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 ; η

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

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

lemma *fis-det-comp*: *deterministic* $R \Longrightarrow$ *deterministic* $S \Longrightarrow$ *fis* $(R \cdot S) =$ *fis* R $\cdot$ *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-subh*: $R \subseteq S \Longrightarrow R \subseteq S$; (*epsiloff* // *epsiloff*)
by (*metis R-O-Id alpha-def alpha-epsiloff lres-galois order-refl relcomp-mono*)

lemma *alpha-Lambda-galois*: $(\alpha$ $R \subseteq S) = (R \subseteq \Lambda$ S ; (*epsiloff* // *epsiloff*))
proof −
have *a*: $(\alpha$ $R \subseteq S) = (R \subseteq S$ // *epsiloff*)
by (*simp add*: *alpha-def lres-galois*)
have S // *epsiloff* $= (\Lambda$ S ; *epsiloff*) // *epsiloff*
by (*metis alpha-Lambda-canc alpha-def*)
also have $\ldots = \Lambda$ S ; (*epsiloff* // *epsiloff*)
by (*simp add*: *det-lambda det-lres*)
finally have S // *epsiloff* $= \Lambda$ S ; (*epsiloff* // *epsiloff*)
.
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 *epsiloff-eta-lres*: *epsiloff* ; $\eta \subseteq$ *epsiloff* // *epsiloff*
proof −
have *epsiloff* ; η ; *epsiloff* = *epsiloff* ; α (Λ *Id*)
by (*simp add*: *O-assoc alpha-def eta-def*)
also have ... = *epsiloff*
by *simp*
finally have *epsiloff* ; η ; *epsiloff* = *epsiloff*.
thus *?thesis*
by (*smt* (*verit*) *lres-galois order-refl*)
qed

lemma *eta-alpha-galois*: (R ; $\eta \subseteq S$; (*epsiloff* // *epsiloff*)) = ($R \subseteq \alpha\ S$)
proof
assume R ; $\eta \subseteq S$; (*epsiloff* // *epsiloff*)
hence $R \subseteq S$; (*epsiloff* // *epsiloff*) ; *epsiloff*
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 ; $\eta \subseteq \alpha\ S$; η
by (*simp add*: *relcomp-mono*)
hence R ; $\eta \subseteq S$; *epsiloff* ; η
by (*simp add*: *alpha-def*)
thus R ; $\eta \subseteq S$; (*epsiloff* // *epsiloff*)
using *epsiloff-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*: $R \subseteq S \Longrightarrow \Lambda\ R \subseteq \Lambda\ S$; (*epsiloff* // *epsiloff*)
by (*metis alpha-Lambda-canc alpha-Lambda-galois*)

lemma *eta-iso*: $R \subseteq S \Longrightarrow R$; $\eta \subseteq S$; η ; (*epsiloff* // *epsiloff*)
by (*simp add*: *eta-alpha-galois*)

lemma *alpha-iso*: $R \subseteq S$; (*epsiloff* // *epsiloff*) $\Longrightarrow \alpha\ R \subseteq \alpha\ S$
by (*metis* (*no-types*, *lifting*) *alpha-def alpha-ord-pres alpha-relcomp conv-Omega conv-Omega-epsiloff*)

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

lemma *eta-canc-dcl*: $\alpha\ R\ ;\ \eta \subseteq R\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff})$
  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 $R \subseteq S\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff})$
  shows $\Lambda\ (\alpha\ R) \subseteq S\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff})$
proof −
  have $\alpha\ R \subseteq \alpha\ S$
    by (*simp add*: *alpha-iso assms*(*2*))
  hence $\Lambda\ (\alpha\ R) \subseteq \Lambda\ (\alpha\ S)\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff})$
    by (*simp add*: *Lambda-iso*)
  thus *?thesis*
    using *assms*(*1*) **by** *auto*
qed

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

lemma *fis-fus-galois*: $(\alpha\ R\ ;\ \eta \subseteq S\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff})) = (R \subseteq \Lambda\ (\alpha\ S)\ ;\ (\mathit{epsiloff}\ /\!/\ \mathit{epsiloff}))$
  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*]: $\alpha\ (R\ ;\ \Omega^{\smile}) = \alpha\ R$

by (*metis alpha-def alpha-relcomp conv-Omega-epsiloff*)

lemma *fis-fis* [*simp*]: *fis* ∘ *fis* = *fis*
unfolding *fun-eq-iff fis-def* **by** *simp*

lemma *fus-fus* [*simp*]: *fus* ∘ *fus* = *fus*
unfolding *fun-eq-iff fus-def* **by** *simp*

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

lemma *fus-fis* [*simp*]: *fus* ∘ *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*: $(R)_{\mathcal{P}}$ = *fus* (*epsiloff* ; R)
by (*simp add*: *alpha-def fus-def klift-var*)

lemma *fus-eta-klift*: *fus* $R = \eta\ ;\ (R)_{\mathcal{P}}$
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*]: ν (*fis* R) = *fis* R
by (*metis alpha-fp empty-iff equals0I fis-alpha relcompE*)

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

lemma *tau-fis* [*simp*]: τ (*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*: $U\ ;\ (P \cap \eta)_*\ ;\ \Omega^\smile = U\ ;\ (P \cap \eta)_*$
  unfolding *U-def relcomp-unfold plift-def Omega-set eta-set converse-def*
  apply *safe*
  apply *clarsimp*
  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*: $dom\ (fis\ R) = dom\ (\alpha\ R)$
  unfolding *dom-set fis-set alpha-set* **by** *blast*

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

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

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

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

4.6 Box and diamond

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

definition *dia* :: $('a, 'b)\ mrel \Rightarrow ('b\ set, 'a\ set)\ rel$ **where**
  $dia\ R = \mathcal{P}\ ((\alpha\ R)^\smile)$

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

lemma *dia-set*: *dia R* = {(*B*, *A*). *A* = {*a*. ∃ *C*. (*a*, *C*) ∈ *R* ∧ *C* ∩ *B* ≠ {}}}
unfolding *dia-def pow-set Image-def alpha-set converse-def* **by** *force*

lemma *box-Omega*: *box R* = Λ (Ω˘ // *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* ≠ {} ⟷ *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** ∪∪ *65*)
where *inner-union* ≡ *p-prod*

definition *inner-intersection* :: (*'a*,*'b*) *mrel* ⇒ (*'a*,*'b*) *mrel* ⇒ (*'a*,*'b*) *mrel* (**infixl** ∩∩ *65*) **where**
R ∩∩ *S* ≡ { (*a*,*B*) . ∃ *C D* . *B* = *C* ∩ *D* ∧ (*a*,*C*) ∈ *R* ∧ (*a*,*D*) ∈ *S* }

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

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

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

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

lemma *ic-involutive* [*simp*]:
$\sim\sim R = R$
by (*simp add*: *mr-simp*)

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

lemma *ic-antidist-iu*:
$\sim(R \cup\cup S) = \sim R \cap\cap \sim S$
apply (*clarsimp 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*:
$B = \{a\} \cap D \Longrightarrow -D = \{a\} \Longrightarrow B = \{\}$
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*:
$R \subseteq S \Longrightarrow R \cup\cup T \subseteq S \cup\cup T$
by (*metis iu-right-dist-ou subset-Un-eq*)

lemma *iu-right-isotone*:
$R \subseteq S \Longrightarrow T \cup\cup R \subseteq T \cup\cup S$
by (*simp add*: *iu-commute iu-left-isotone*)

lemma *iu-isotone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow R \cup\cup P \subseteq S \cup\cup Q$
by (*meson dual-order.trans iu-left-isotone iu-right-isotone*)

lemma *ii-left-isotone*:
$R \subseteq S \Longrightarrow R \cap\cap\ T \subseteq S \cap\cap\ T$
by (*metis ii-right-dist-ou subset-Un-eq*)

lemma *ii-right-isotone*:
$R \subseteq S \Longrightarrow T \cap\cap\ R \subseteq T \cap\cap\ S$
by (*simp add*: *ii-commute ii-left-isotone*)

lemma *ii-isotone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow R \cap\cap\ P \subseteq S \cap\cap\ Q$
by (*meson ii-left-isotone ii-right-isotone order-trans*)

lemma *iu-right-subdist-ii*:
$(R \cap\cap\ S) \cup\cup\ T \subseteq (R \cup\cup\ T) \cap\cap\ (S \cup\cup\ T)$
apply (*clarsimp simp*: *mr-simp*)
by (*metis sup-inf-distrib2*)

lemma *ii-right-subdist-iu*:
$(R \cup\cup\ S) \cap\cap\ T \subseteq (R \cap\cap\ T) \cup\cup\ (S \cap\cap\ T)$
apply (*clarsimp simp*: *mr-simp*)
by (*metis inf-sup-distrib2*)

lemma *ic-isotone*:
$R \subseteq S \Longrightarrow \sim R \subseteq \sim S$
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*:
$R \subseteq R \cap\cap\ R$

unfolding *inner-intersection-def* **by** *force*

definition *inner-Union* :: $('i \Rightarrow ('a,'b)\ mrel) \Rightarrow 'i\ set \Rightarrow ('a,'b)\ mrel$ ($\bigcup\bigcup$ -|- [*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* :: $('i \Rightarrow ('a,'b)\ mrel) \Rightarrow 'i\ set \Rightarrow ('a,'b)\ mrel$ ($\bigcap\bigcap$ -|- [*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*:
$\sim\bigcup\bigcup X|I = \bigcap\bigcap(inner\text{-}complement\ o\ X)|I$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply (*metis* (*mono-tags*, *lifting*) *Compl-UN double-compl*)
by (*clarsimp simp*: *mr-simp*) *blast*

lemma *ic-antidist-iI*:
$\sim\bigcap\bigcap X|I = \bigcup\bigcup(inner\text{-}complement\ o\ X)|I$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply (*metis Compl-INT double-complement*)
by (*clarsimp simp*: *mr-simp*) *blast*

lemma *iu-right-dist-oU*:
$\bigcup X \cup\cup T = (\bigcup R{\in}X\ .\ R \cup\cup T)$
by (*clarsimp simp*: *mr-simp*) *blast*

lemma *ii-right-dist-oU*:
$\bigcup X \cap\cap T = (\bigcup R{\in}X\ .\ R \cap\cap T)$
by (*clarsimp simp*: *mr-simp*) *blast*

lemma *iu-right-subdist-iI*:
$\bigcap\bigcap X|I \cup\cup T \subseteq \bigcap\bigcap(\lambda i\ .\ X\ i \cup\cup T)|I$
apply (*clarsimp simp*: *mr-simp*)
by (*metis INT-simps*(*6*))

lemma *ii-right-subdist-iU*:
$\bigcup\bigcup X|I \cap\cap T \subseteq \bigcup\bigcup(\lambda i\ .\ X\ i \cap\cap T)|I$

by (*clarsimp simp*: *mr-simp*, *metis UN-extend-simps*(*4*))

lemma *ic-dist-oU*:
$\sim\bigcup X = \bigcup(\textit{inner-complement} \text{ ‘ } X)$
by (*auto simp*: *mr-simp*)

lemma *ic-dist-oI*:
$\sim\bigcap X = \bigcap(\textit{inner-complement} \text{ ‘ } X)$
by (*auto simp*: *mr-simp*)

lemma *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 $\forall b{\in}B . \exists g . f\ b = \bigcup(g \text{ ‘ } I) \wedge (\forall i{\in}I . (b,g\ i) \in X\ i)$
from *this* **obtain** *g* **where** *2*: $\forall b{\in}B . f\ b = \bigcup(g\ b \text{ ‘ } I) \wedge (\forall i{\in}I . (b,g\ b\ i) \in X\ i)$
by *metis*
hence *3*: $\bigcup(f \text{ ‘ } B) = (\bigcup b{\in}B . \bigcup(g\ b \text{ ‘ } I))$
by (*meson SUP-cong*)
let $?h = \lambda i . \bigcup b{\in}B . g\ b\ i$
have $\bigcup(f \text{ ‘ } B) = \bigcup(?h \text{ ‘ } I) \wedge (\forall i{\in}I . \exists B . (a,B) \in R \wedge (\exists f . (\forall b{\in}B . (b,f\ b) \in X\ i) \wedge ?h\ i = \bigcup(f \text{ ‘ } B)))$
using *1 2 3* **by** (*metis SUP-commute*)
thus *?thesis*
by *auto*
qed
done

lemma *sp-right-subdist-iU*:
$(\bigcup\bigcup X|I) * R \subseteq \bigcup\bigcup(\lambda i . X\ i * R)|I$
by (*clarsimp simp*: *mr-simp*, *blast*)

lemma *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 $\forall i{\in}I . \exists B . (a,B) \in X\ i \wedge (\exists g . (\forall b{\in}B . (b,g\ b) \in R) \wedge f\ i = \bigcup(g \text{ ‘ } B))$
from *this* **obtain** *B g* **where** *1*: $\forall i{\in}I . (a,B\ i) \in X\ i \wedge (\forall b{\in}B\ i . (b,g\ i\ b) \in R) \wedge f\ i = \bigcup(g\ i \text{ ‘ } B\ i)$
by *metis*
let $?B = \bigcup(B \text{ ‘ } I)$
let $?g = \lambda b . \bigcup\{\ g\ i\ b \mid i .\ i{\in}I \wedge b \in B\ i\ \}$
have $\forall b{\in}?B . (b,?g\ b) \in R$

proof (*rule ballI*)
fix *b*
let *?I* = { *i* | *i* . *i*∈*I* ∧ *b* ∈ *B i* }
assume *2*: *b* ∈ ⋃(*B* ' *I*)
have (*b*,*?g b*) ∈ ⋃⋃(λ*j* . *R*)|*?I*
apply (*clarsimp simp*: *mr-simp*)
apply (*rule exI*[*of* - λ*i* . *g i b*])
using *1* **by** *blast*
thus (*b*,*?g b*) ∈ *R*
using *2* **by** (*smt* (*verit*) *assms UN-E empty-Collect-eq subset-iff*)
qed
hence *?B* = ⋃(*B* ' *I*) ∧ (∀ *i*∈*I* . (*a*,*B i*) ∈ *X i*) ∧ (∀ *b*∈*?B* . (*b*,*?g b*) ∈ *R*) ∧ ⋃(*f* ' *I*) = ⋃(*?g* ' *?B*)
using *1* **by** *auto*
thus ∃ *B* . (∃ *f* . *B* = ⋃(*f* ' *I*) ∧ (∀ *i*∈*I* . (*a*,*f i*) ∈ *X i*)) ∧ (∃ *g* . (∀ *b*∈*B* . (*b*,*g b*) ∈ *R*) ∧ ⋃(*f* ' *I*) = ⋃(*g* ' *B*))
by (*metis* (*no-types*, *lifting*))
qed
done

5.2 Dual

abbreviation *dual* :: ('*a*,'*b*) *mrel* ⇒ ('*a*,'*b*) *mrel* (-^{d} [*100*] *100*)
where R^d ≡ ∼−*R*

lemma *dual*:
R^d = { (*a*,*B*) . (*a*,−*B*) ∉ *R* }
by (*simp add*: *inner-complement-def*)

declare *dual* [*mr-simp*]

lemma *dual-antitone*:
R ⊆ *S* ⟹ S^d ⊆ R^d
by (*simp add*: *ic-isotone*)

lemma *ic-oc-dual*:
∼*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*:
$(R ∪ S)^d$ = R^d ∩ S^d
by (*simp add*: *ic-dist-oi*)

lemma *dual-antidist-oi*:
$(R ∩ S)^d$ = R^d ∪ S^d

by (*simp add*: *ic-dist-ou*)

lemma *dual-dist-oc*:
$(-R)^d = -R^d$
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*:
$(\bigcup X)^d = \bigcap (dual \text{ ` } X)$
by (*simp add*: *ic-dist-oI uminus-Sup*)

lemma *dual-antidist-oI*:
$(\bigcap X)^d = \bigcup (dual \text{ ` } X)$
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
$R \odot S \equiv \{ (a,C) . \exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f\, b) \in S) \wedge C = \bigcap \{ f\, b \mid b . b \in B \}) \}$

lemma *co-prod-im*:
$R \odot S = \{ (a,C) . \exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f\, b) \in S) \wedge C = \bigcap ((\lambda x . f\, x) \text{ ` } B)) \}$
by (*auto simp*: *co-prod-def*)

lemma *co-prod-iff*:
$(a,C) \in (R \odot S) \longleftrightarrow (\exists B . (a,B) \in R \wedge (\exists f . (\forall b \in B . (b,f\, b) \in S) \wedge C = \bigcap \{ f\, b \mid b . b \in B \}))$
by (*unfold co-prod-im*, *auto*)

declare *co-prod-im* [*mr-simp*]

lemma *co-prod*:
$R \odot S = \sim(R * \sim S)$
apply (*clarsimp simp*: *mr-simp*)
by (*smt* (*verit*) *Collect-cong Compl-INT Compl-UN case-prodI2 double-complement old.prod.case*)

lemma *cp-left-isotone*:
$R \subseteq S \Longrightarrow R \odot T \subseteq S \odot T$
by (*simp add*: *co-prod ic-isotone s-prod-isol*)

lemma *cp-right-isotone*:
$R \subseteq S \Longrightarrow T \odot R \subseteq T \odot S$

by (*smt* (*verit*) *co-prod-iff in-mono subrelI*)

lemma *cp-isotone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow R \odot P \subseteq S \odot Q$
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 (*clarsimp 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 $R \odot (\bigcap\bigcap X|I) = \sim(R * (\bigcup\bigcup(\textit{inner-complement o } X)|I))$
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* :: $'a \times 'b\ set \Rightarrow 'a \times 'b\ set \Rightarrow bool$ (**infix** $\preceq_{\cup\cup}$ *50*)
where
  $x \preceq_{\cup\cup} y \equiv fst\ x = fst\ y \wedge snd\ x \subseteq snd\ y$

definition *inner-order-ii* :: $'a \times 'b\ set \Rightarrow 'a \times 'b\ set \Rightarrow bool$ (**infix** $\preceq_{\cap\cap}$ *50*)
where
  $x \preceq_{\cap\cap} y \equiv fst\ x = fst\ y \wedge snd\ x \supseteq snd\ y$

lemma *inner-order-dual*:
  $x \preceq_{\cup\cup} y \longleftrightarrow y \preceq_{\cap\cap} x$
  by (*metis inner-order-ii-def inner-order-iu-def*)

interpretation *inner-order-iu*: *order* $(\preceq_{\cup\cup})$ $\lambda x\ y\ .\ x \preceq_{\cup\cup} y \wedge x \neq y$
  by (*unfold-locales*, *auto simp add*: *inner-order-iu-def*)

5.5 Up-closure, down-closure and convex-closure

abbreviation *up* :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (-$\uparrow$ [*100*] *100*)
  where $R\uparrow \equiv R \cup\cup U$

abbreviation *down* :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (-$\downarrow$ [*100*] *100*)
  where $R\downarrow \equiv R \cap\cap U$

abbreviation *convex* :: $('a,'b)\ mrel \Rightarrow ('a,'b)\ mrel$ (-$\updownarrow$ [*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*:
  $R\updownarrow = \{ (a,B) . \exists C\ D . (a,C) \in R \wedge (a,D) \in R \wedge C \subseteq B \wedge B \subseteq D \}$
  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*:
  $\sim(R\updownarrow) = (\sim R)\updownarrow$
  by (*simp add*: *ic-dist-oi ic-down ic-up inf-commute*)

lemma *up-isotone*:
  $R \subseteq S \Longrightarrow R\uparrow \subseteq S\uparrow$
  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*:
  $(R \cap\cap S)\uparrow = R\uparrow \cap\cap S\uparrow$
proof (*rule antisym*)
  show $(R \cap\cap S)\uparrow \subseteq R\uparrow \cap\cap S\uparrow$
    by (*simp add*: *iu-right-subdist-ii*)
next
  have $\bigwedge a\ B\ C\ D\ E . (a,B) \in R \Longrightarrow (a,C) \in S \Longrightarrow \exists F . (\exists G . (B \cup D) \cap (C \cup E) = F \cup G) \wedge (\exists H\ I . F = H \cap I \wedge (a,H) \in R \wedge (a,I) \in S)$
  proof −
    fix *a B C D E*

assume *1*: $(a,B) \in R$
assume *2*: $(a,C) \in S$
let $?F = B \cap C$
let $?G = (B \cap E) \cup (D \cap C) \cup (D \cap E)$
have $(B \cup D) \cap (C \cup E) = ?F \cup ?G$
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 (*clarsimp simp*: *mr-simp*)
qed

lemma *down-isotone*:
$R \subseteq S \Longrightarrow R\downarrow \subseteq S\downarrow$
by (*fact ii-left-isotone*)

lemma *down-increasing*:
$R \subseteq R\downarrow$
by (*metis ic-involutive ic-isotone ic-up up-increasing*)

lemma *down-idempotent* [*simp*]:
$R\downarrow\downarrow = R\downarrow$
by (*simp add*: *ic-down ic-injective*)

lemma *down-dist-ou*:
$(R \cup S)\downarrow = R\downarrow \cup S\downarrow$
by (*fact ii-right-dist-ou*)

lemma *down-dist-iu*:
$(R \cup\cup S)\downarrow = R\downarrow \cup\cup S\downarrow$
by (*simp add*: *ic-antidist-ii ic-antidist-iu ic-injective up-dist-ii*)

lemma *down-dist-ii*:
$(R \cap\cap S)\downarrow = R\downarrow \cap\cap S\downarrow$
by (*metis down-idempotent ii-assoc ii-commute*)

lemma *convex-isotone*:
$R \subseteq S \Longrightarrow R\updownarrow \subseteq S\updownarrow$
by (*meson Int-mono down-isotone up-isotone*)

lemma *convex-increasing*:
$R \subseteq R\updownarrow$
by (*simp add*: *down-increasing up-increasing*)

lemma *convex-idempotent* [*simp*]:
$R\updownarrow\updownarrow = R\updownarrow$
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 $?f = \lambda c\ .\ if\ c \in B\ then\ \{c\}\ else\ \{\}$
    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 $\bigwedge a\ B\ f\ .\ (a,B) \in R \Longrightarrow \forall b{\in}B\ .\ \exists C\ .\ (\exists D\ .\ f\ b = C \cap D) \wedge (b,C) \in S$

$\Longrightarrow \exists E . (\exists F . (\bigcup b{\in}B . f\ b) = E \cap F) \wedge (\exists B . (a,B) \in R \wedge (\exists g . (\forall b{\in}B . (b,g\ b) \in S) \wedge E = (\bigcup b{\in}B . g\ b)))$
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*:
$R{\downarrow} \cap S{\downarrow} = (R \cap\cap S){\downarrow}$
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*:
$R{\downarrow} \cap S = \{\} \Longrightarrow R \cap S{\uparrow} = \{\}$
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 $R{\downarrow} \cap S = \{\} \Longrightarrow R \cap S{\uparrow} = \{\}$
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*:
$R{\downarrow} \subseteq S \longleftrightarrow R \subseteq -((-S){\uparrow})$
by (*metis disjoint-eq-subset-Compl double-compl oi-down-up-iff*)

lemma *up-double-complement-down*:
$R{\uparrow} \subseteq S \longleftrightarrow R \subseteq -((-S){\downarrow})$
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*:
$(\bigcup X){\uparrow} = \bigcup (up \text{ ` } X)$

by (*simp add*: *iu-right-dist-oU*)

lemma *up-dist-iU*:
assumes $I \neq \{\}$
shows $(\bigcup\bigcup X|I)\uparrow = \bigcup\bigcup(up\ o\ X)|I$
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 $\forall i{\in}I . \exists B . (\exists C . f\ i = B \cup C) \wedge (a,B) \in X\ i$
from *this* **obtain** *g* **where** $\forall i{\in}I . (\exists C . f\ i = g\ i \cup C) \wedge (a,g\ i) \in X\ i$
by *metis*
hence $(\exists C . \bigcup(f\ `\ I) = \bigcup(g\ `\ I) \cup C) \wedge (\bigcup(g\ `\ I) = \bigcup(g\ `\ I) \wedge (\forall i{\in}I .$
$(a,g\ i) \in X\ i))$
by *auto*
thus $\exists B . (\exists C . \bigcup(f\ `\ I) = B \cup C) \wedge (\exists f . B = \bigcup(f\ `\ I) \wedge (\forall i{\in}I . (a,f\ i) \in$
$X\ i))$
by *auto*
qed
done

lemma *up-dist-iI*:
$(\bigcap\bigcap X|I)\uparrow = \bigcap\bigcap(up\ o\ X)|I$
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 $\forall i{\in}I . \exists B . (\exists C . f\ i = B \cup C) \wedge (a,B) \in X\ i$
from *this* **obtain** *g* **where** $\forall i{\in}I . (\exists C . f\ i = g\ i \cup C) \wedge (a,g\ i) \in X\ i$
by *metis*
hence $(\exists C . \bigcap(f\ `\ I) = \bigcap(g\ `\ I) \cup C) \wedge (\bigcap(g\ `\ I) = \bigcap(g\ `\ I) \wedge (\forall i{\in}I .$
$(a,g\ i) \in X\ i))$
by *auto*
thus $\exists B . (\exists C . \bigcap(f\ `\ I) = B \cup C) \wedge (\exists f . B = \bigcap(f\ `\ I) \wedge (\forall i{\in}I . (a,f\ i) \in$
$X\ i))$
by *auto*
qed
done

lemma *down-dist-oU*:
$(\bigcup X)\downarrow = \bigcup(down\ `\ X)$
by (*simp add*: *ii-right-dist-oU*)

lemma *down-dist-iU*:

$(\bigcup\bigcup X|I)\downarrow = \bigcup\bigcup(down\ o\ X)|I$
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 $(\exists C\ .\ \bigcup(f\ `\ I) = \bigcup(g\ `\ I) \cap C) \wedge (\bigcup(g\ `\ I) = \bigcup(g\ `\ I) \wedge (\forall i{\in}I\ .\ (a,g\ i) \in X\ i))$
by *auto*
thus $\exists B\ .\ (\exists C\ .\ \bigcup(f\ `\ I) = B \cap C) \wedge (\exists f\ .\ B = \bigcup(f\ `\ I) \wedge (\forall i{\in}I\ .\ (a,f\ i) \in X\ i))$
by *auto*
qed
done

lemma *down-dist-iI*:
assumes $I \neq \{\}$
shows $(\bigcap\bigcap X|I)\downarrow = \bigcap\bigcap(down\ o\ X)|I$
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 $(\exists C\ .\ \bigcap(f\ `\ I) = \bigcap(g\ `\ I) \cap C) \wedge (\bigcap(g\ `\ I) = \bigcap(g\ `\ I) \wedge (\forall i{\in}I\ .\ (a,g\ i) \in X\ i))$
by *auto*
thus $\exists B\ .\ (\exists C\ .\ \bigcap(f\ `\ I) = B \cap C) \wedge (\exists f\ .\ B = \bigcap(f\ `\ I) \wedge (\forall i{\in}I\ .\ (a,f\ i) \in X\ i))$
by *auto*
qed
done

lemma *iU-up-oI*:
assumes $I \neq \{\}$
shows $\bigcup\bigcup(up\ o\ X)|I = \bigcap(up\ `\ X\ `\ I)$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply (*metis UN-absorb sup-assoc*)
apply (*clarsimp simp*: *mr-simp*)

by (*metis UN-constant assms*)

lemma *iI-down-oI*:
assumes $I \neq \{\}$
shows $\bigcap\bigcap(down\ o\ X)|I = \bigcap(down\ `\ X\ `\ I)$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply (*metis INF-absorb Int-assoc*)
apply (*clarsimp 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 (*clarsimp 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 (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *up-dist-iU-oI*:
$\bigcap(up\ `\ X\ `\ I) = (\bigcup\bigcup X|I)\uparrow$
apply (*rule antisym*)
apply (*clarsimp 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 (*clarsimp 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 (*clarsimp simp*: *mr-simp*)

using *assms* **apply** *blast*
apply (*clarsimp 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 (*clarsimp simp*: *mr-simp*)
using *assms* **apply** *blast*
apply (*clarsimp 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*:
$1_{\cup\cup}\updownarrow = 1_{\cup\cup}$
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*:
$1_{\cap\cap}\updownarrow = 1_{\cap\cap}$
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*:
$1\updownarrow = 1$
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*:
$U\updownarrow = U$
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*:
$\{\}\updownarrow = \{\}$
by (*simp add*: *bot-down*)

lemma *down-oi-up-convex*:
$(R\downarrow \cap S\uparrow)\updownarrow = R\downarrow \cap S\uparrow$
unfolding *up down convex* **by** *blast*

lemma *convex-iff-down-oi-up*:
$Q = Q\updownarrow \longleftrightarrow (\exists\, R\ S\ .\ Q = R\downarrow \cap S\uparrow)$
using *down-oi-up-convex* **by** *blast*

lemma *convex-closed-oI*:
$(\bigcap R{\in}X\ .\ R\updownarrow)\updownarrow = (\bigcap R{\in}X\ .\ R\updownarrow)$
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*:
$(R\updownarrow \cap S\updownarrow)\updownarrow = R\updownarrow \cap S\updownarrow$
using *convex-closed-oI*[*of* $\{R,S\}$] **by** *simp*

lemma
$(R\updownarrow \cup\cup S\updownarrow)\updownarrow = R\updownarrow \cup\cup S\updownarrow$
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* ⇒ ('a,'b) *mrel* ⇒ *bool* (**infixl** $\sqsubseteq\uparrow$ *50*)
where
$R \sqsubseteq\uparrow S \equiv S \subseteq R\uparrow$

abbreviation *convex-less-eq* :: ('a,'b) *mrel* ⇒ ('a,'b) *mrel* ⇒ *bool* (**infixl** $\sqsubseteq\updownarrow$ *50*)
where
$R \sqsubseteq\updownarrow S \equiv R \sqsubseteq\downarrow S \wedge R \sqsubseteq\uparrow S$

abbreviation *Convex-less-eq* :: ('a,'b) *mrel* ⇒ ('a,'b) *mrel* ⇒ *bool* (**infixl** $\sqsubseteq\Updownarrow$ *50*) **where**
$R \sqsubseteq\Updownarrow S \equiv R \subseteq S\updownarrow$

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 (*clarsimp 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*:
$R \sqsubseteq\Updownarrow S \longleftrightarrow (\forall a\ C\ .\ (a,C) \in R \longrightarrow (\exists B\ D\ .\ (a,B) \in S \wedge (a,D) \in S \wedge B \subseteq C \wedge C \subseteq D))$
by (*meson lower-less-eq semilattice-inf-class.le-inf-iff upper-less-eq*)

lemma *Convex-lower-upper*:
$R \sqsubseteq\Updownarrow S \longleftrightarrow R \sqsubseteq\downarrow S \wedge S \sqsubseteq\uparrow R$
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*:
$R \sqsubseteq\updownarrow R$
by (*simp add*: *lower-reflexive upper-reflexive*)

lemma *Convex-reflexive*:
$R \sqsubseteq\Updownarrow R$
by (*fact convex-increasing*)

lemma *lower-transitive*:
$R \sqsubseteq\downarrow S \Longrightarrow S \sqsubseteq\downarrow T \Longrightarrow R \sqsubseteq\downarrow T$
using *down-idempotent down-isotone* **by** *blast*

lemma *upper-transitive*:
$R \sqsubseteq\uparrow S \Longrightarrow S \sqsubseteq\uparrow T \Longrightarrow R \sqsubseteq\uparrow T$
using *up-idempotent up-isotone* **by** *blast*

lemma *convex-transitive*:
$R \sqsubseteq\updownarrow S \Longrightarrow S \sqsubseteq\updownarrow T \Longrightarrow R \sqsubseteq\updownarrow T$
by (*meson lower-transitive upper-transitive*)

lemma *Convex-transitive*:
$R \sqsubseteq\Updownarrow S \Longrightarrow S \sqsubseteq\Updownarrow T \Longrightarrow R \sqsubseteq\Updownarrow T$
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*:
$\{\} \sqsubseteq\Updownarrow R$
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*:
$R \sqsubseteq\Updownarrow U$
using *U-par-idem top-down top-upper-least* **by** *auto*

lemma *lower-iu-increasing*:
$R \sqsubseteq\downarrow R \cup\cup R$
by (*meson dual-order.trans lower-reflexive subidem-par*)

lemma *upper-iu-increasing*:
$R \sqsubseteq\uparrow R \cup\cup S$
using *p-prod-isor top-upper-least* **by** *auto*

lemma *convex-iu-increasing*:
$R \sqsubseteq\updownarrow R \cup\cup R$
by (*simp add*: *lower-iu-increasing upper-iu-increasing*)

lemma *Convex-iu-increasing*:
$R \sqsubseteq\Updownarrow R \cup\cup R$
by (*simp add*: *iu-up lower-iu-increasing upper-reflexive*)

lemma *lower-ii-decreasing*:
$R \cap\cap S \sqsubseteq\downarrow R$
by (*metis ii-right-isotone top-down top-lower-greatest*)

lemma *upper-ii-decreasing*:
$R \cap\cap R \sqsubseteq\uparrow R$
using *convex-reflexive ii-sub-idempotent* **by** *fastforce*

lemma *convex-ii-decreasing*:
$R \cap\cap R \sqsubseteq\updownarrow R$
by (*simp add*: *lower-ii-decreasing upper-ii-decreasing*)

lemma *Convex-ii-increasing*:
$R \sqsubseteq\Updownarrow R \cap\cap R$
by (*simp add*: *ii-down lower-reflexive upper-ii-decreasing*)

lemma *iu-lower-left-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow R \cup\cup T \sqsubseteq\downarrow S \cup\cup T$
by (*simp add*: *down-dist-iu iu-isotone lower-reflexive*)

lemma *iu-upper-left-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow R \cup\cup T \sqsubseteq\uparrow S \cup\cup T$
by (*metis* (*no-types*, *lifting*) *iu-assoc iu-commute iu-left-isotone*)

lemma *iu-convex-left-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow R \cup\cup T \sqsubseteq\updownarrow S \cup\cup T$
by (*simp add*: *iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-Convex-left-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow R \cup\cup T \sqsubseteq\Updownarrow S \cup\cup T$
by (*simp add*: *iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-lower-right-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow T \cup\cup R \sqsubseteq\downarrow T \cup\cup S$
by (*simp add*: *iu-commute iu-lower-left-isotone*)

lemma *iu-upper-right-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow T \cup\cup R \sqsubseteq\uparrow T \cup\cup S$
by (*simp add*: *iu-assoc iu-right-isotone*)

lemma *iu-convex-right-isotone*:

$R \sqsubseteq\!\updownarrow S \Longrightarrow T \cup\cup R \sqsubseteq\!\updownarrow T \cup\cup S$
by (*simp add*: *iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-Convex-right-isotone*:
$R \sqsubseteq\!\Updownarrow S \Longrightarrow T \cup\cup R \sqsubseteq\!\Updownarrow T \cup\cup S$
by (*simp add*: *iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-lower-isotone*:
$R \sqsubseteq\!\downarrow S \Longrightarrow P \sqsubseteq\!\downarrow Q \Longrightarrow R \cup\cup P \sqsubseteq\!\downarrow S \cup\cup Q$
by (*simp add*: *down-dist-iu iu-isotone*)

lemma *iu-upper-isotone*:
$R \sqsubseteq\!\uparrow S \Longrightarrow P \sqsubseteq\!\uparrow Q \Longrightarrow R \cup\cup P \sqsubseteq\!\uparrow S \cup\cup Q$
by (*simp add*: *iu-isotone up-dist-iu*)

lemma *iu-convex-isotone*:
$R \sqsubseteq\!\updownarrow S \Longrightarrow P \sqsubseteq\!\updownarrow Q \Longrightarrow R \cup\cup P \sqsubseteq\!\updownarrow S \cup\cup Q$
by (*simp add*: *iu-lower-isotone iu-upper-isotone*)

lemma *iu-Convex-isotone*:
$R \sqsubseteq\!\Updownarrow S \Longrightarrow P \sqsubseteq\!\Updownarrow Q \Longrightarrow R \cup\cup P \sqsubseteq\!\Updownarrow S \cup\cup Q$
by (*simp add*: *down-dist-iu iu-isotone up-dist-iu*)

lemma *ii-lower-left-isotone*:
$R \sqsubseteq\!\downarrow S \Longrightarrow R \cap\cap T \sqsubseteq\!\downarrow S \cap\cap T$
by (*simp add*: *down-dist-ii ii-isotone lower-reflexive*)

lemma *ii-upper-left-isotone*:
$R \sqsubseteq\!\uparrow S \Longrightarrow R \cap\cap T \sqsubseteq\!\uparrow S \cap\cap T$
by (*simp add*: *ii-isotone up-dist-ii upper-reflexive*)

lemma *ii-convex-left-isotone*:
$R \sqsubseteq\!\updownarrow S \Longrightarrow R \cap\cap T \sqsubseteq\!\updownarrow S \cap\cap T$
by (*simp add*: *ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-Convex-left-isotone*:
$R \sqsubseteq\!\Updownarrow S \Longrightarrow R \cap\cap T \sqsubseteq\!\Updownarrow S \cap\cap T$
by (*simp add*: *ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-lower-right-isotone*:
$R \sqsubseteq\!\downarrow S \Longrightarrow T \cap\cap R \sqsubseteq\!\downarrow T \cap\cap S$
by (*simp add*: *ii-assoc ii-right-isotone*)

lemma *ii-upper-right-isotone*:
$R \sqsubseteq\!\uparrow S \Longrightarrow T \cap\cap R \sqsubseteq\!\uparrow T \cap\cap S$
by (*simp add*: *ii-commute ii-upper-left-isotone*)

lemma *ii-convex-right-isotone*:
$R \sqsubseteq\!\updownarrow S \Longrightarrow T \cap\cap R \sqsubseteq\!\updownarrow T \cap\cap S$

by (*simp add*: *ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-Convex-right-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow T \cap\cap R \sqsubseteq\Updownarrow T \cap\cap S$
by (*simp add*: *ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow P \sqsubseteq\downarrow Q \Longrightarrow R \cap\cap P \sqsubseteq\downarrow S \cap\cap Q$
by (*simp add*: *down-dist-ii ii-isotone*)

lemma *ii-upper-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow P \sqsubseteq\uparrow Q \Longrightarrow R \cap\cap P \sqsubseteq\uparrow S \cap\cap Q$
by (*simp add*: *ii-isotone up-dist-ii*)

lemma *ii-convex-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow P \sqsubseteq\updownarrow Q \Longrightarrow R \cap\cap P \sqsubseteq\updownarrow S \cap\cap Q$
by (*simp add*: *ii-lower-isotone ii-upper-isotone*)

lemma *ii-Convex-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow P \sqsubseteq\Updownarrow Q \Longrightarrow R \cap\cap P \sqsubseteq\Updownarrow S \cap\cap Q$
by (*simp add*: *ii-lower-isotone ii-upper-isotone*)

lemma *ou-lower-left-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow R \cup T \sqsubseteq\downarrow S \cup T$
by (*meson le-sup-iff lower-reflexive lower-transitive*)

lemma *ou-upper-left-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow R \cup T \sqsubseteq\uparrow S \cup T$
by (*metis Un-subset-iff sup.coboundedI1 up-dist-ou upclosed-ext*)

lemma *ou-convex-left-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow R \cup T \sqsubseteq\updownarrow S \cup T$
by (*meson ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-Convex-left-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow R \cup T \sqsubseteq\Updownarrow S \cup T$
by (*meson le-inf-iff ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-lower-right-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow T \cup R \sqsubseteq\downarrow T \cup S$
by (*metis Un-commute ou-lower-left-isotone*)

lemma *ou-upper-right-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow T \cup R \sqsubseteq\uparrow T \cup S$
by (*metis Un-commute ou-upper-left-isotone*)

lemma *ou-convex-right-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow T \cup R \sqsubseteq\updownarrow T \cup S$
by (*meson ou-lower-right-isotone ou-upper-right-isotone*)

lemma *ou-Convex-right-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow T \cup R \sqsubseteq\Updownarrow T \cup S$
by (*metis Un-commute ou-Convex-left-isotone*)

lemma *ou-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow P \sqsubseteq\downarrow Q \Longrightarrow R \cup P \sqsubseteq\downarrow S \cup Q$
using *down-dist-ou* **by** *blast*

lemma *ou-upper-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow P \sqsubseteq\uparrow Q \Longrightarrow R \cup P \sqsubseteq\uparrow S \cup Q$
by (*simp add*: *iu-right-dist-ou sup.coboundedI1 sup.coboundedI2*)

lemma *ou-convex-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow P \sqsubseteq\updownarrow Q \Longrightarrow R \cup P \sqsubseteq\updownarrow S \cup Q$
by (*meson ou-lower-isotone ou-upper-isotone*)

lemma *ou-Convex-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow P \sqsubseteq\Updownarrow Q \Longrightarrow R \cup P \sqsubseteq\Updownarrow S \cup Q$
by (*metis le-inf-iff ou-lower-isotone ou-upper-isotone*)

lemma *sp-lower-left-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow T * R \sqsubseteq\downarrow T * S$
by (*simp add*: *down-dist-sp s-prod-isor*)

lemma *sp-upper-left-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow T * R \sqsubseteq\uparrow T * S$
by (*meson cl3 dual-order.trans s-prod-isor upper-iu-increasing*)

lemma *sp-convex-left-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow T * R \sqsubseteq\updownarrow T * S$
by (*simp add*: *sp-lower-left-isotone sp-upper-left-isotone*)

lemma *sp-Convex-left-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow T * R \sqsubseteq\Updownarrow T * S$
by (*simp add*: *sp-lower-left-isotone sp-upper-left-isotone*)

lemma *cp-lower-left-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow T \odot R \sqsubseteq\downarrow T \odot S$
by (*smt* (*verit*) *co-prod ic-antidist-ii ic-antidist-iu ic-isotone ic-top sp-upper-left-isotone*)

lemma *cp-upper-left-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow T \odot R \sqsubseteq\uparrow T \odot S$
by (*simp add*: *cp-right-isotone up-dist-cp*)

lemma *cp-convex-left-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow T \odot R \sqsubseteq\updownarrow T \odot S$
by (*simp add*: *cp-lower-left-isotone cp-upper-left-isotone*)

lemma *cp-Convex-left-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow T \odot R \sqsubseteq\Updownarrow T \odot S$
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*:
$R \sqsubseteq\updownarrow S \longleftrightarrow \sim S \sqsubseteq\updownarrow \sim R$
by (*meson lower-ic-upper upper-ic-lower*)

lemma *Convex-ic*:
$R \sqsubseteq\Updownarrow S \longleftrightarrow \sim R \sqsubseteq\Updownarrow \sim S$
by (*metis le-inf-iff lower-ic-upper upper-ic-lower*)

lemma *up-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow R\uparrow \sqsubseteq\downarrow S\uparrow$
by (*fact iu-lower-left-isotone*)

lemma *up-upper-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow R\uparrow \sqsubseteq\uparrow S\uparrow$
by (*fact iu-left-isotone*)

lemma *up-convex-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow R\uparrow \sqsubseteq\updownarrow S\uparrow$
by (*fact iu-convex-left-isotone*)

lemma *up-Convex-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow R\uparrow \sqsubseteq\Updownarrow S\uparrow$
by (*fact iu-Convex-left-isotone*)

lemma *down-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow R\downarrow \sqsubseteq\downarrow S\downarrow$
by (*fact down-isotone*)

lemma *down-upper-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow R\downarrow \sqsubseteq\uparrow S\downarrow$
by (*fact ii-upper-left-isotone*)

lemma *down-convex-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow R\downarrow \sqsubseteq\updownarrow S\downarrow$
by (*fact ii-convex-left-isotone*)

lemma *down-Convex-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow R\downarrow \sqsubseteq\Updownarrow S\downarrow$
by (*fact ii-Convex-left-isotone*)

lemma *convex-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow R\updownarrow \sqsubseteq\downarrow S\updownarrow$
by (*metis convex-idempotent convex-increasing le-inf-iff lower-transitive*)

lemma *convex-upper-isotone*:
$R \sqsubseteq\uparrow S \Longrightarrow R\updownarrow \sqsubseteq\uparrow S\updownarrow$
by (*simp add*: *convex-lower-isotone ic-convex upper-ic-lower*)

lemma *convex-convex-isotone*:
$R \sqsubseteq\updownarrow S \Longrightarrow R\updownarrow \sqsubseteq\updownarrow S\updownarrow$
by (*simp add*: *convex-lower-isotone convex-upper-isotone*)

lemma *convex-Convex-isotone*:
$R \sqsubseteq\Updownarrow S \Longrightarrow R\updownarrow \sqsubseteq\Updownarrow S\updownarrow$
by (*fact convex-isotone*)

lemma *subset-lower*:
$R \subseteq S \Longrightarrow R \sqsubseteq\downarrow S$
using *lower-reflexive* **by** *auto*

lemma *subset-upper*:
$R \subseteq S \Longrightarrow S \sqsubseteq\uparrow R$
using *upper-reflexive* **by** *blast*

lemma *subset-Convex*:
$R \subseteq S \Longrightarrow R \sqsubseteq\Updownarrow S$
by (*simp add*: *subset-lower subset-upper*)

lemma *oi-subset-lower-left-isotone*:
$R \subseteq S \Longrightarrow R \cap T \sqsubseteq\downarrow S \cap T$
using *lower-reflexive* **by** *fastforce*

lemma *oi-subset-upper-left-antitone*:
$R \subseteq S \Longrightarrow S \cap T \sqsubseteq\uparrow R \cap T$
using *upper-reflexive* **by** *force*

lemma *oi-subset-Convex-left-isotone*:
$R \subseteq S \Longrightarrow R \cap T \sqsubseteq\Updownarrow S \cap T$
by (*simp add*: *oi-subset-lower-left-isotone oi-subset-upper-left-antitone*)

lemma *oi-subset-lower-right-isotone*:
$R \subseteq S \Longrightarrow T \cap R \sqsubseteq\downarrow T \cap S$
by (*simp add*: *oi-subset-lower-left-isotone semilattice-inf-class.inf-commute*)

lemma *oi-subset-upper-right-antitone*:

$R \subseteq S \Longrightarrow T \cap S \sqsubseteq\uparrow T \cap R$
by (*simp add*: *oi-subset-upper-left-antitone semilattice-inf-class.inf-commute*)

lemma *oi-subset-Convex-right-isotone*:
$R \subseteq S \Longrightarrow T \cap R \sqsubseteq\Updownarrow T \cap S$
using *oi-subset-Convex-left-isotone* **by** *blast*

lemma *oi-subset-lower-isotone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow R \cap P \sqsubseteq\downarrow S \cap Q$
by (*meson Int-mono subset-lower*)

lemma *oi-subset-upper-antitone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow S \cap Q \sqsubseteq\uparrow R \cap P$
by (*meson Int-mono subset-upper*)

lemma *oi-subset-Convex-isotone*:
$R \subseteq S \Longrightarrow P \subseteq Q \Longrightarrow R \cap P \sqsubseteq\Updownarrow S \cap Q$
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 (*clarsimp simp*: *mr-simp*) *blast*

lemma
$R \sqsubseteq\downarrow S \longleftrightarrow R = R \cap\cap S$

nitpick[*expect=genuine,card=1*]
oops

lemma
$R \sqsubseteq\uparrow S \longleftrightarrow S = R \cup\cup S$
nitpick[*expect=genuine,card=1*]
oops

lemma *convex-oi-Convex-iu*:
$R\updownarrow \cap S\updownarrow \sqsubseteq\Updownarrow R \cup\cup S$
by (*meson inf-le1 inf-le2 iu-Convex-isotone order-trans subidem-par*)

lemma *convex-oi-Convex-ii*:
$R\updownarrow \cap S\updownarrow \sqsubseteq\Updownarrow R \cap\cap S$
by (*meson ii-Convex-isotone ii-sub-idempotent inf-le1 inf-le2 order-trans*)

lemma *convex-oi-iu-ii*:
$R\updownarrow \cap S\updownarrow = (R \cup\cup S)\uparrow \cap (R \cap\cap S)\downarrow$
by (*metis down-dist-ii-oi inf-assoc inf-left-commute up-dist-iu-oi*)

lemma *ii-lower-iu*:
$R \cap\cap S \sqsubseteq\downarrow R \cup\cup S$
apply (*clarsimp simp*: *mr-simp*)
by (*metis Un-Int-eq*(*2*) *inf-left-commute*)

lemma *ii-upper-iu*:
$R \cap\cap S \sqsubseteq\uparrow R \cup\cup S$
by (*simp add*: *ic-antidist-ii ic-antidist-iu ii-lower-iu upper-ic-lower*)

lemma *ii-convex-iu*:
$R \cap\cap S \sqsubseteq\updownarrow R \cup\cup S$
by (*simp add*: *ii-lower-iu ii-upper-iu*)

lemma *convex-oi-iu-ii-convex*:
$R\updownarrow \cap S\updownarrow = (R \cup\cup S)\updownarrow \cap (R \cap\cap S)\updownarrow$
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*:
$Id = 1 ; 1^{\smile}$
unfolding *Id-def converse-def relcomp-unfold s-id-def* **by** *force*

lemma *dom-explicit*:
$Dom\ R = R ; U \cap 1$
by (*clarsimp simp*: *mr-simp Dom-def*) *blast*

lemma *dom-explicit-2*:

Dom R = *R* ; *top* ∩ *1*
apply (*clarsimp simp*: *mr-simp Dom-def*)
apply *safe*
apply (*simp add*: *relcomp.relcompI top-def*)
apply *blast*
by *blast*

lemma *total-dom*:
total R ⟷ *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 ⟷ $1_{\cup\cup}$ = *R* ∗ $1_{\cup\cup}$
by (*metis total-dom U-c cd-iso dc dc-prop2*)

lemma *domain-pointwise*:
x ∈ *R* ∗ $1_{\cup\cup}$ ⟷ (∃ *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 ⟷ (∀ *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* $\leq$ *one-class.one*
proof (*rule ccontr*)
assume *5*: $\neg$ *card ?B* $\leq$ *one-class.one*
from *this* **obtain** *B* **where** *6*: (*a*,*B*) $\in$ *R*
by *fastforce*
hence *card* (*?B* $-$ {*B*}) $\geq$ *one-class.one*
using *2 5* **by** *auto*
from *this* **obtain** *C* **where** (*a*,*C*) $\in$ *R* $\wedge$ *B* $\neq$ *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
qed
next
assume *5*: $\forall$ *a* . *finite* { *B* . (*a*,*B*) $\in$ *R* } $\wedge$ *card* { *B* . (*a*,*B*) $\in$ *R* } $\leq$ *one-class.one*
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*
let *?B* = { *B* . (*a*,*B*) $\in$ *R* }
have *6*: *finite ?B*
using *5* **by** *simp*
assume (*a*,*B*) $\in$ *R* $\wedge$ (*a*,*C*) $\in$ *R*
hence {*B*,*C*} $\subseteq$ *?B*
by *simp*
hence *card* {*B*,*C*} $\leq$ *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 $\longleftrightarrow$ ($\forall$ *S* . *R* $*$ $1_{\cup\cup}$ = *S* $*$ $1_{\cup\cup}$ $\wedge$ *S* $\subseteq$ *R* $\longrightarrow$ *S* = *R*)
proof
assume *1*: $\forall$ *S* . *R* $*$ $1_{\cup\cup}$ = *S* $*$ $1_{\cup\cup}$ $\wedge$ *S* $\subseteq$ *R* $\longrightarrow$ *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 ≠ C
      let ?S = R − { (a,C) }
      have 4: R ∗ 1∪∪ = ?S ∗ 1∪∪
      proof
        show R ∗ 1∪∪ ⊆ ?S ∗ 1∪∪
        proof
          fix x::'a × 'f set
          assume x ∈ R ∗ 1∪∪
          from this obtain b D where (b,D) ∈ R ∧ x = (b,{})
            by (meson domain-pointwise)
          thus x ∈ ?S ∗ 1∪∪
            using 2 3 by (metis domain-pointwise Pair-inject insertE insert-Diff)
        qed
      next
        show ?S ∗ 1∪∪ ⊆ R ∗ 1∪∪
          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 ∀ S . R ∗ 1∪∪ = S ∗ 1∪∪ ∧ S ⊆ R ⟶ S = R
  proof
    fix S
    show R ∗ 1∪∪ = S ∗ 1∪∪ ∧ S ⊆ R ⟶ S = R
    proof
      assume 6: R ∗ 1∪∪ = S ∗ 1∪∪ ∧ S ⊆ R
      have R ⊆ S
      proof
        fix x
        assume 7: x ∈ R
        from this obtain a B where 8: x = (a,B)
          by fastforce
        show x ∈ S
        proof (cases ∃ C . C ≠ B ∧ (a,C) ∈ 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 ⟷ (∀ *a* . { *B* . (*a*,*B*) ∈ *R* } ≠ {})
 by (*simp add*: *total-set*)

lemma *total-3*:
 total R ⟷ (∀ *a* . *finite* { *B* . (*a*,*B*) ∈ *R* } ⟶ *card* { *B* . (*a*,*B*) ∈ *R* } ≥ *one-class.one*)
 by (*metis finite.emptyI nonempty-set-card total-2*)

lemma *total-4*: *total R* ⟷ $1_{\cup\cup}$ ⊆ *R* ∗ $1_{\cup\cup}$
 by (*simp add*: *c6 order-antisym-conv total-eq*)

lemma *deterministic-2*:
 deterministic R ⟷ (∀ *a* . *card* { *B* . (*a*,*B*) ∈ *R* } = *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*↕
 apply (*rule antisym*)
  apply (*simp add*: *lower-reflexive upper-reflexive*)
 apply (*clarsimp 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* ∪∪ *S*
 apply (*rule antisym*)
  apply (*meson convex-reflexive upper-ii-upper-bound*)
 apply (*clarsimp 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 (*clarsimp simp*: *mr-simp*)
by (*metis assms semilattice-inf-class.inf.idem univalent-set*)

lemma *univalent-down-iu-idempotent*:
assumes *univalent S*
shows $S = S\downarrow \cup\cup S$
apply (*rule antisym*)
apply (*meson convex-reflexive subset-upper upper-ii-upper-bound*)
apply (*clarsimp simp*: *mr-simp*)
by (*metis assms lattice-class.sup-inf-absorb sup-commute univalent-set*)

lemma *univalent-up-ii-idempotent*:
assumes *univalent S*
shows $S = S\uparrow \cap\cap S$
apply (*rule antisym*)
apply (*metis assms ii-left-isotone univalent-ii-idempotent upclosed-ext*)
apply (*clarsimp simp*: *mr-simp*)
by (*metis Int-commute assms lattice-class.inf-sup-absorb univalent-set*)

lemma *univalent-convex-iu-idempotent*:
assumes *univalent S*
shows $S = S\updownarrow \cup\cup S$
by (*metis assms univalent-convex univalent-iu-idempotent*)

lemma *univalent-convex-ii-idempotent*:
assumes *univalent S*
shows $S = S\updownarrow \cap\cap S$
by (*metis assms univalent-convex univalent-ii-idempotent*)

lemma *univalent-iu-closed*:
univalent R $\Longrightarrow$ *univalent S* $\Longrightarrow$ *univalent* $(R \cup\cup S)$
by (*smt* (*verit*, *best*) *case-prodD mem-Collect-eq p-prod-def univalent-set*)

lemma *univalent-ii-closed*:
univalent R $\Longrightarrow$ *univalent S* $\Longrightarrow$ *univalent* $(R \cap\cap S)$
by (*smt* (*verit*, *ccfv-SIG*) *CollectD Pair-inject case-prodE inner-intersection-def univalent-set*)

lemma *total-lower*:
total R $\longleftrightarrow$ $1_{\cup\cup} \sqsubseteq\downarrow R$
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 (*clarsimp 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*:
  $total\ R \Longrightarrow total\ S \Longrightarrow total\ (R \cup\cup S)$
  by (*meson lower-transitive total-lower total-lower-iu*)

lemma *total-ii-closed*:
  $total\ R \Longrightarrow total\ S \Longrightarrow total\ (R \cap\cap S)$
  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 $V \sqsubseteq\uparrow R \longleftrightarrow (\forall\, a\ B\ C\ .\ (a,B) \in R \wedge (a,C) \in V \longrightarrow C \subseteq B)$
proof –
have $V \sqsubseteq\uparrow R \longleftrightarrow (\forall\, a\ C\ .\ (a,C) \in R \longrightarrow (\exists B\ .\ (a,B) \in V \wedge B \subseteq C))$
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 $T \sqsubseteq\downarrow S$
shows $T \sqsubseteq\uparrow S$
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 $T \sqsubseteq\downarrow S$
shows $T \sqsubseteq\updownarrow S$
by (*simp add*: *assms total-univalent-lower-implies-upper*)

lemma *total-univalent-upper-implies-lower*:
assumes *total T*
and *univalent S*
and $S \sqsubseteq\uparrow T$
shows $S \sqsubseteq\downarrow T$
proof (*clarsimp 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 $S \sqsubseteq\updownarrow T$
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 $S \sqsubseteq\downarrow T \longleftrightarrow S \sqsubseteq\updownarrow T$
by (*simp add*: *assms deterministic-lower-upper*)

lemma *deterministic-upper-convex*:
assumes *deterministic T*
and *deterministic S*
shows $S \sqsubseteq\uparrow T \longleftrightarrow S \sqsubseteq\updownarrow T$
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 $\Longrightarrow$ $R\downarrow * T \subseteq (R * T)\downarrow$
  by (*simp add*: *down-dist-sp total-down-sp-sp-down*)

lemma *total-down-dist-sp*:
  total T $\Longrightarrow$ $(R * T)\downarrow = R\downarrow * T\downarrow$
  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 (*clarsimp 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*: $R \cup\cup S = S \Longrightarrow R \sqsubseteq\uparrow S$
    using *upper-iu-increasing* **by** *blast*
  have *2*: $R \sqsubseteq\uparrow S \Longrightarrow S \subseteq R \cup\cup S$
    by (*simp add*: *upper-ii-upper-bound*)
  have $R \sqsubseteq\uparrow S \Longrightarrow R \cup\cup S \subseteq S$
    apply (*clarsimp 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 $(R \sqsubseteq\downarrow S) = (R \cap\cap S = R)$
  apply (*clarsimp 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* ⇒ $('a,'b)$ *mrel* ⇒ *bool* (**infixl** $=\downarrow$ *50*)
where
  $R =\downarrow S \equiv R \sqsubseteq\downarrow S \wedge S \sqsubseteq\downarrow R$

abbreviation *upper-eq* :: $('a,'b)$ *mrel* ⇒ $('a,'b)$ *mrel* ⇒ *bool* (**infixl** $=\uparrow$ *50*)
where
  $R =\uparrow S \equiv R \sqsubseteq\uparrow S \wedge S \sqsubseteq\uparrow R$

abbreviation *convex-eq* :: $('a,'b)$ *mrel* ⇒ $('a,'b)$ *mrel* ⇒ *bool* (**infixl** $=\updownarrow$ *50*)
where
  $R =\updownarrow S \equiv R \sqsubseteq\updownarrow S \wedge S \sqsubseteq\updownarrow R$

lemma *Convex-eq*:
  $R =\updownarrow S \equiv R \sqsubseteq\Updownarrow S \wedge S \sqsubseteq\Updownarrow R$
  by (*smt* (*z3*) *semilattice-inf-class.le-inf-iff*)

lemma *convex-lower-upper*:
  $R =\updownarrow S \longleftrightarrow R =\downarrow S \wedge R =\uparrow S$
  by *auto*

lemma *lower-eq-down*:
  $R =\downarrow S \longleftrightarrow R\downarrow = S\downarrow$
  using *down-idempotent down-lower-isotone lower-reflexive* **by** *blast*

lemma *upper-eq-up*:
  $R =\uparrow S \longleftrightarrow R\uparrow = S\uparrow$
  by (*metis p-prod-comm upclosed-ext upper-ii-up*)

lemma *convex-eq-convex*:
  $R =\updownarrow S \longleftrightarrow R\updownarrow = S\updownarrow$
  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*:
$R =\updownarrow R$
by (*simp add*: *lower-reflexive upper-reflexive*)

lemma *lower-eq-symmetric*:
$R =\downarrow S \Longrightarrow S =\downarrow R$
by *simp*

lemma *upper-eq-symmetric*:
$R =\uparrow S \Longrightarrow S =\uparrow R$
by *simp*

lemma *convex-eq-symmetric*:
$R =\updownarrow S \Longrightarrow S =\updownarrow R$
by *simp*

lemma *lower-eq-transitive*:
$R =\downarrow S \Longrightarrow S =\downarrow T \Longrightarrow R =\downarrow T$
using *lower-transitive* **by** *auto*

lemma *upper-eq-transitive*:
$R =\uparrow S \Longrightarrow S =\uparrow T \Longrightarrow R =\uparrow T$
using *upper-transitive* **by** *auto*

lemma *convex-eq-transitive*:
$R =\updownarrow S \Longrightarrow S =\updownarrow T \Longrightarrow R =\updownarrow T$
by (*meson lower-transitive upper-transitive*)

lemma *ou-lower-eq-left-congruence*:
$R =\downarrow S \Longrightarrow R \cup T =\downarrow S \cup T$
using *ou-lower-left-isotone* **by** *blast*

lemma *ou-upper-eq-left-congruence*:
$R =\uparrow S \Longrightarrow R \cup T =\uparrow S \cup T$

using *ou-upper-left-isotone* **by** *blast*

lemma *ou-convex-eq-left-congruence*:
$R =\updownarrow S \Longrightarrow R \cup T =\updownarrow S \cup T$
by (*meson ou-lower-left-isotone ou-upper-left-isotone*)

lemma *ou-lower-eq-right-congruence*:
$R =\downarrow S \Longrightarrow T \cup R =\downarrow T \cup S$
using *ou-lower-right-isotone* **by** *blast*

lemma *ou-upper-eq-right-congruence*:
$R =\uparrow S \Longrightarrow T \cup R =\uparrow T \cup S$
using *ou-upper-right-isotone* **by** *blast*

lemma *ou-convex-eq-right-congruence*:
$R =\updownarrow S \Longrightarrow T \cup R =\updownarrow T \cup S$
by (*meson ou-lower-right-isotone ou-upper-right-isotone*)

lemma *ou-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow P =\downarrow Q \Longrightarrow R \cup P =\downarrow S \cup Q$
using *ou-lower-isotone* **by** *blast*

lemma *ou-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow P =\uparrow Q \Longrightarrow R \cup P =\uparrow S \cup Q$
using *ou-upper-isotone* **by** *blast*

lemma *ou-convex-eq-congruence*:
$R =\updownarrow S \Longrightarrow P =\updownarrow Q \Longrightarrow R \cup P =\updownarrow S \cup Q$
by (*meson ou-lower-isotone ou-upper-isotone*)

lemma *iu-lower-eq-left-congruence*:
$R =\downarrow S \Longrightarrow R \cup\cup T =\downarrow S \cup\cup T$
using *iu-lower-left-isotone* **by** *blast*

lemma *iu-upper-eq-left-congruence*:
$R =\uparrow S \Longrightarrow R \cup\cup T =\uparrow S \cup\cup T$
using *iu-upper-left-isotone* **by** *blast*

lemma *iu-convex-eq-left-congruence*:
$R =\updownarrow S \Longrightarrow R \cup\cup T =\updownarrow S \cup\cup T$
by (*simp add*: *iu-lower-left-isotone iu-upper-left-isotone*)

lemma *iu-lower-eq-right-congruence*:
$R =\downarrow S \Longrightarrow T \cup\cup R =\downarrow T \cup\cup S$
using *iu-lower-right-isotone* **by** *blast*

lemma *iu-upper-eq-right-congruence*:
$R =\uparrow S \Longrightarrow T \cup\cup R =\uparrow T \cup\cup S$
using *iu-upper-right-isotone* **by** *blast*

lemma *iu-convex-eq-right-congruence*:
$R =\updownarrow S \Longrightarrow T \cup\cup R =\updownarrow T \cup\cup S$
by (*simp add*: *iu-lower-right-isotone iu-upper-right-isotone*)

lemma *iu-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow P =\downarrow Q \Longrightarrow R \cup\cup P =\downarrow S \cup\cup Q$
using *iu-lower-isotone* **by** *blast*

lemma *iu-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow P =\uparrow Q \Longrightarrow R \cup\cup P =\uparrow S \cup\cup Q$
using *iu-upper-isotone* **by** *blast*

lemma *iu-convex-eq-congruence*:
$R =\updownarrow S \Longrightarrow P =\updownarrow Q \Longrightarrow R \cup\cup P =\updownarrow S \cup\cup Q$
by (*simp add*: *iu-lower-isotone iu-upper-isotone*)

lemma *ii-lower-eq-left-congruence*:
$R =\downarrow S \Longrightarrow R \cap\cap T =\downarrow S \cap\cap T$
using *ii-lower-left-isotone* **by** *blast*

lemma *ii-upper-eq-left-congruence*:
$R =\uparrow S \Longrightarrow R \cap\cap T =\uparrow S \cap\cap T$
using *ii-upper-left-isotone* **by** *blast*

lemma *ii-convex-eq-left-congruence*:
$R =\updownarrow S \Longrightarrow R \cap\cap T =\updownarrow S \cap\cap T$
by (*simp add*: *ii-lower-left-isotone ii-upper-left-isotone*)

lemma *ii-lower-eq-right-congruence*:
$R =\downarrow S \Longrightarrow T \cap\cap R =\downarrow T \cap\cap S$
using *ii-lower-right-isotone* **by** *blast*

lemma *ii-upper-eq-right-congruence*:
$R =\uparrow S \Longrightarrow T \cap\cap R =\uparrow T \cap\cap S$
using *ii-upper-right-isotone* **by** *blast*

lemma *ii-convex-eq-right-congruence*:
$R =\updownarrow S \Longrightarrow T \cap\cap R =\updownarrow T \cap\cap S$
by (*simp add*: *ii-lower-right-isotone ii-upper-right-isotone*)

lemma *ii-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow P =\downarrow Q \Longrightarrow R \cap\cap P =\downarrow S \cap\cap Q$
using *ii-lower-isotone* **by** *blast*

lemma *ii-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow P =\uparrow Q \Longrightarrow R \cap\cap P =\uparrow S \cap\cap Q$
using *ii-upper-isotone* **by** *blast*

lemma *ii-convex-eq-congruence*:
$R =\updownarrow S \Longrightarrow P =\updownarrow Q \Longrightarrow R \sqcap\sqcap P =\updownarrow S \sqcap\sqcap Q$
by (*simp add*: *ii-lower-isotone ii-upper-isotone*)

lemma *sp-lower-eq-left-congruence*:
$R =\downarrow S \Longrightarrow T * R =\downarrow T * S$
by (*simp add*: *sp-lower-left-isotone*)

lemma *sp-upper-eq-left-congruence*:
$R =\uparrow S \Longrightarrow T * R =\uparrow T * S$
by (*simp add*: *sp-upper-left-isotone*)

lemma *sp-convex-eq-left-congruence*:
$R =\updownarrow S \Longrightarrow T * R =\updownarrow T * S$
by (*simp add*: *sp-lower-left-isotone sp-upper-left-isotone*)

lemma *cp-lower-eq-left-congruence*:
$R =\downarrow S \Longrightarrow T \odot R =\downarrow T \odot S$
by (*simp add*: *cp-lower-left-isotone*)

lemma *cp-upper-eq-left-congruence*:
$R =\uparrow S \Longrightarrow T \odot R =\uparrow T \odot S$
by (*simp add*: *cp-upper-left-isotone*)

lemma *cp-convex-eq-left-congruence*:
$R =\updownarrow S \Longrightarrow T \odot R =\updownarrow T \odot S$
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*:
$R =\updownarrow S \longleftrightarrow \sim R =\updownarrow \sim S$
by (*meson lower-ic-upper upper-ic-lower*)

lemma *up-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow R\uparrow =\downarrow S\uparrow$
by (*fact iu-lower-eq-left-congruence*)

lemma *up-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow R\uparrow =\uparrow S\uparrow$
by (*fact iu-upper-eq-left-congruence*)

lemma *up-convex-eq-congruence*:

$R =\updownarrow S \Longrightarrow R\uparrow =\updownarrow S\uparrow$
by (*fact iu-convex-eq-left-congruence*)

lemma *down-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow R\downarrow =\downarrow S\downarrow$
by (*fact ii-lower-eq-left-congruence*)

lemma *down-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow R\downarrow =\uparrow S\downarrow$
by (*fact ii-upper-eq-left-congruence*)

lemma *down-convex-eq-congruence*:
$R =\updownarrow S \Longrightarrow R\downarrow =\updownarrow S\downarrow$
by (*fact ii-convex-eq-left-congruence*)

lemma *convex-lower-eq-congruence*:
$R =\downarrow S \Longrightarrow R\updownarrow =\downarrow S\updownarrow$
by (*simp add*: *convex-lower-isotone*)

lemma *convex-upper-eq-congruence*:
$R =\uparrow S \Longrightarrow R\updownarrow =\uparrow S\updownarrow$
by (*simp add*: *convex-upper-isotone*)

lemma *convex-convex-eq-congruence*:
$R =\updownarrow S \Longrightarrow R\updownarrow =\updownarrow S\updownarrow$
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 $(R =\updownarrow S) \longleftrightarrow (R = S)$
  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*:
$R =\updownarrow R\updownarrow$
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*:
$(\forall P . P \sqsubseteq\downarrow Q \longleftrightarrow P \sqsubseteq\downarrow R \wedge P \sqsubseteq\downarrow S) \longleftrightarrow Q =\downarrow R \cap\cap S$
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*:
$(\forall P . P \sqsubseteq\uparrow Q \longleftrightarrow P \sqsubseteq\uparrow R \wedge P \sqsubseteq\uparrow S) \longleftrightarrow Q =\uparrow R \cup S$
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*:
$I \neq \{\} \Longrightarrow (\bigcap\bigcap(\lambda j . R)|I) =\downarrow R$
by (*metis iI-down lower-eq-down*)

lemma *upper-iU-idempotent*:
$I \neq \{\} \Longrightarrow (\bigcup\bigcup(\lambda j . R)|I) =\uparrow R$
by (*metis iU-up upper-eq-up*)

lemma *down-closed-intersection-closed*:
$R = R\downarrow \Longrightarrow \forall I . I \neq \{\} \longrightarrow (\bigcap\bigcap(\lambda j . R)|I) \subseteq R$

by (*metis lower-iI-idempotent*)

lemma *up-closed-union-closed*:
$R = R\uparrow \Longrightarrow \forall I\ .\ I \neq \{\} \longrightarrow (\bigcup\bigcup(\lambda j\ .\ R)|I) \subseteq R$
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*:
  $R\updownarrow \subseteq S\updownarrow \longleftrightarrow R \sqsubseteq\downarrow S \wedge S \sqsubseteq\uparrow R$
  by (*meson convex-eq-up-closed le-inf-iff lower-transitive upper-transitive*)

lemma *convex-order-Convex*:
  $R\updownarrow \subseteq S\updownarrow \longleftrightarrow R \sqsubseteq\Updownarrow S$
  by (*meson Convex-lower-upper convex-order-lower-upper*)

7 Fusion and Fission

7.1 Atoms and co-atoms

definition *atoms* :: ('*a*,'*b*) *mrel* ($A_{\cup\cup}$)
  where $A_{\cup\cup} \equiv \{\ (a,\{b\})\ |\ a\ b\ .\ True\ \}$

definition *co-atoms* :: ('*a*,'*b*) *mrel* ($A_{\cap\cap}$)
  where $A_{\cap\cap} \equiv \{\ (a, UNIV\ -\ \{b\})\ |\ a\ b\ .\ True\ \}$

declare *atoms-def* [*mr-simp*] *co-atoms-def* [*mr-simp*]

lemma *atoms-solution*:
  $A_{\cup\cup}\uparrow = -1_{\cup\cup}$
  apply (*rule antisym*)
   apply (*clarsimp simp*: *mr-simp*)
  apply (*clarsimp 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* × '*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 (*clarsimp 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 (*clarsimp 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* $\wedge$ *finite B* $\longrightarrow$ *card B* $\geq$ *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* $\longrightarrow$ *card B* = *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* =↓ *W*
shows *V* = *W*
using *assms inner-univalent-down* **by** *blast*

lemma *inner-total-down-closed*:
inner-total T ⟹ *R* ⊆ *T* ⟹ *inner-total R*
by *simp*

lemma *inner-univalent-down-closed*:
inner-univalent T ⟹ *R* ⊆ *T* ⟹ *inner-univalent R*
by *simp*

lemma *inner-deterministic-down-closed*:
inner-deterministic T ⟹ *R* ⊆ *T* ⟹ *inner-deterministic R*
by *blast*

lemma *inner-univalent-convex*:
assumes *inner-univalent R*
shows *R* = *R*↕
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 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 ⟹ *R O converse s-id* = *R O 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* ⊑↓ *S*) ⟷ (*R* ⊆ *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 $(R =\downarrow S) \longleftrightarrow (R = S)$
  by (*meson assms inner-deterministic-lower-eq lower-reflexive*)

lemma *inner-deterministic-upper-eq-iff*:
  assumes *inner-deterministic R*
    and *inner-deterministic S*
  shows $(R =\uparrow S) \longleftrightarrow (R = S)$
  by (*simp add*: *antisym assms inner-deterministic-upper-iff*)

lemma *inner-deterministic-convex-eq-iff*:
  assumes *inner-deterministic R*
    and *inner-deterministic S*
  shows $(R =\updownarrow S) \longleftrightarrow (R = S)$
  by (*metis assms inner-deterministic-lower-eq-iff inner-deterministic-upper-eq-iff*)

lemma
  inner-univalent R $\Longrightarrow$ *inner-univalent S* $\Longrightarrow$ *inner-univalent* $(R \cup\cup S)$
  nitpick[*expect=genuine,card=1,2*]
  oops

lemma *inner-univalent-ii-closed*:
  inner-univalent R $\Longrightarrow$ *inner-univalent S* $\Longrightarrow$ *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 $\Longrightarrow$ *inner-total S* $\Longrightarrow$ *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 $\Longrightarrow$ *inner-total S* $\Longrightarrow$ *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*:
  $R \sqsubseteq\downarrow$ *fus* R
  apply (*clarsimp 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 (*clarsimp 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 $\forall\, R$. *deterministic* $(f\, R)$
    and $\forall\, R\; S \;.\; R \sqsubseteq\downarrow S \wedge$ *deterministic* $S \longrightarrow f\, R \sqsubseteq\downarrow S$
  shows $f\, T =$ *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*:
  $($*fus* $R)\downarrow = -((-(R\downarrow) \cap A_{\cup\cup})\uparrow)$
proof
  show $($*fus* $R)\downarrow \subseteq -((-(R\downarrow) \cap A_{\cup\cup})\uparrow)$
    apply (*clarsimp simp*: *mr-simp*)
    by *blast*
next

show $-((-(R\downarrow) \cap A_{\cup\cup})\uparrow) \subseteq (\mathit{fus}\ R)\downarrow$
proof (*clarsimp 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** $x \in C \wedge (a,C) \in R$
using *1* **by** (*metis IntD1 insert-Diff insert-is-Un singletonI*)
thus $x \in \bigcup\{\ C\ .\ (a,C) \in R\ \}$
by *blast*
qed
thus $\exists D\ .\ A = \bigcup\{\ C\ .\ (a,C) \in R\ \} \cap D$
by *auto*
qed
qed

lemma *fusion-up-char*:
$(\mathit{fus}\ R)\uparrow = -((\sim(R\downarrow) \cap A_{\cap\cap})\downarrow)$
proof
show $(\mathit{fus}\ R)\uparrow \subseteq -((\sim(R\downarrow) \cap A_{\cap\cap})\downarrow)$
apply (*clarsimp simp*: *mr-simp*)
by *blast*
next
show $-((\sim(R\downarrow) \cap A_{\cap\cap})\downarrow) \subseteq (\mathit{fus}\ R)\uparrow$
proof (*clarsimp simp*: *mr-simp*)
fix *a A*
assume *1*: $\forall C\ .\ (\forall D\ .\ A \neq C \cap D) \vee (\forall B\ .\ (\forall D\ .\ -\ C \neq B \cap D) \vee (a,B) \notin R) \vee (\forall b\ .\ C \neq \mathit{UNIV} - \{b\})$
have $\bigcup\{\ C\ .\ (a,C) \in R\ \} \subseteq A$
proof
fix *x*
assume $x \in \bigcup\{\ C\ .\ (a,C) \in R\ \}$
from *this* **obtain** *C* **where** $x \in C \wedge (a,C) \in R$
by *blast*
thus $x \in A$
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 $\exists C\ .\ A = \bigcup\{\ C\ .\ (a,C) \in R\ \} \cup C$
by *auto*
qed
qed

lemma *fusion-up-char-2*:
$(\mathit{fus}\ R)\uparrow = -(((R\downarrow \cap A_{\cup\cup}) * \sim 1)\downarrow)$
by (*simp add*: *atoms-sp-cp co-prod fusion-up-char ic-atoms ic-dist-oi*)

lemma *fusion-char*:
fus R = $-((-(R\downarrow) \cap A_{\cup\cup})\uparrow) \cap -((\sim(R\downarrow) \cap A_{\cap\cap})\downarrow)$
by (*metis deterministic-def fusion-deterministic fusion-down-char fusion-up-char inf-commute univalent-convex*)

lemma *fusion-char-2*:
fus R = $-((-(R\downarrow) \cap A_{\cup\cup})\uparrow) \cap -(((R\downarrow \cap A_{\cup\cup}) * \sim 1)\downarrow)$
using *fusion-char fusion-up-char fusion-up-char-2* **by** *blast*

lemma *fusion-lower-isotone*:
$R \sqsubseteq\downarrow S \Longrightarrow$ *fus R* $\sqsubseteq\downarrow$ *fus S*
by (*meson fusion-deterministic fusion-least fusion-lower-increasing lower-transitive*)

lemma *fusion-iu-idempotent*:
fus R $\cup\cup$ *fus R* = *fus R*
using *deterministic-def fusion-deterministic univalent-iu-idempotent* **by** *blast*

lemma *fusion-down*:
fus R = *fus* $(R\downarrow)$
by (*simp add*: *fusion-char*)

lemma *fusion-iu-total*:
total T $\Longrightarrow$ $T \cup\cup$ *fus T* = *fus T*
by (*meson deterministic-def fusion-deterministic fusion-lower-increasing total-univalent-lower-iu*)

lemma *fusion-deterministic-fixpoint*:
deterministic R $\longleftrightarrow$ *R* = *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*) *mrel* ⇒ (*'a*,*'b*) *mrel* (*ne* - [*100*] *100*)
where *ne R* ≡ $R \cap -1_{\cup\cup}$

lemma *non-empty*:
ne R = { (*a*,*B*) | *a B* . (*a*,*B*) ∈ *R* ∧ *B* ≠ {} }
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* ∪ *S*) = *ne R* ∪ *ne S*
by (*fact inf-sup-distrib2*)

lemma *ne-down-idempotent*:

ne ((*ne* (*R*↓))↓) = *ne* (*R*↓)
by (*auto simp*: *mr-simp*)

lemma *ne-up*:
(*ne R*)↑ = *ne R* ∗ *1*↑
proof
show (*ne R*)↑ ⊆ *ne R* ∗ *1*↑
apply (*clarsimp simp*: *mr-simp*)
by (*metis UN-constant Un-insert-left insert-absorb*)
next
show *ne R* ∗ *1*↑ ⊆ (*ne R*)↑
proof (*clarsimp 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 (*clarsimp 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 (*clarsimp 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* ∧ *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*) ∧ *x* ∈ *R* ∧ *B* ≠ {}
by (*metis Int-iff Int-lower2 inner-total-2 surj-pair*)
from *this* **obtain** *C* **where** *5*: *B* ⊆ *C* ∧ (*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*) ∧ *x* ∈ *S* ∧ *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* ∧ (*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 (*clarsimp simp*: *mr-simp*)
from *this* **obtain** *D* **where** *10*: *b* ∈ *D* ∧ (*a*,*D*) ∈ *R*
using *9* **by** *blast*
from *this* **obtain** *E* **where** *11*: *D* ⊆ *E* ∧ (*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 $x \in ne\ R$
using *6 7 8 10* **by** *blast*
qed
qed
qed
next
assume *12*: $\forall R . fus\ R = fus\ S \wedge R \sqsubseteq\downarrow S \longrightarrow ne\ R = ne\ S$
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*: $fus\ ?R = fus\ S$
apply (*clarsimp simp*: *mr-simp*)
by *blast*
have $?R \sqsubseteq\downarrow S$
using *lower-less-eq* **by** *fastforce*
hence $ne\ ?R = ne\ S$
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*:
$ne\ (\bigcup X) = \bigcup (non\text{-}empty\ `\ X)$
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 (*clarsimp 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 $S \sqsubseteq\downarrow$ *fis* R
proof (*clarsimp 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 $\forall R$. *inner-deterministic* $(f\ R)$
    and $\forall R\ S . S \sqsubseteq\downarrow R \wedge$ *inner-deterministic* $S \longrightarrow S \sqsubseteq\downarrow f\ R$
  shows $f\ T =$ *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*:
  $R \sqsubseteq\downarrow S \Longrightarrow$ *fis* $R \sqsubseteq\downarrow$ *fis* S
  by (*meson fission-greatest fission-inner-deterministic fission-lower-decreasing lower-transitive*)

lemma *fission-idempotent*:
  fis (*fis* R) $=$ *fis* R
  by (*metis comp-apply fis-fis*)

lemma *fission-top*:
  fis $U = A_{\cup\cup}$
  using *fission-var top-down top-upper-least* **by** *fastforce*

lemma *fission-down*:
  fis $R =$ *fis* $(R\downarrow)$

by (*simp add*: *fission-var*)

lemma *fission-ne-fixpoint*:
fis R = *ne* (*fis R*)
using *fission-inner-deterministic* **by** *blast*

lemma *fission-down-ne-fixpoint*:
fis R = *ne* ((*fis R*)↓)
by (*metis fission-inner-deterministic fission-ne-fixpoint fusion-down inner-univalent-char lower-ii-decreasing*)

lemma *fission-inner-deterministic-fixpoint*:
inner-deterministic R ⟷ *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*) ⊆ *fis R* ∗ *fis S*
proof
fix *x*
assume *x* ∈ *fis* (*R* ∗ *S*)
from *this* **obtain** *a b B* **where** *1*: *x* = (*a*,{*b*}) ∧ (*a*,*B*) ∈ *R* ∗ *S* ∧ *b* ∈ *B*
by (*smt CollectD fission-set*)
from *this* **obtain** *C f* **where** *2*: (*a*,*C*) ∈ *R* ∧ (∀ *c* ∈ *C* . (*c*,*f c*) ∈ *S*) ∧ *B* = ⋃{ *f c* | *c* . *c* ∈ *C* }
by (*simp add*: *mr-simp*) *blast*
from *this* **obtain** *c* **where** *3*: *b* ∈ *f c* ∧ *c* ∈ *C*
using *1* **by** *blast*
let *?B* = {*c*}
let *?f* = λ*x* . {*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*) ∧ {*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 $(R \cup S)$ = *fis* $R \cup$ *fis* S
by (*simp add*: *down-dist-ou fission-var inf-sup-distrib2*)

lemma *fission-sp-iu-unit*:
fis $(R * 1_{\cup\cup}) = \{\}$
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) $\sqsubseteq\downarrow R$
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *fusion-fission-lower-increasing*:
$R \sqsubseteq\downarrow$ *fus* (*fis* R)
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *fission-fusion-galois*:
fis $R \sqsubseteq\downarrow S \longleftrightarrow R \sqsubseteq\downarrow$ *fus* S
apply (*rule iffI*)
apply (*meson fusion-fission-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* $S \Longrightarrow$ *fis* $R \sqsubseteq\downarrow S$
by (*metis fission-fusion-galois fusion-lower-increasing*)

lemma *fission-below-ne-down-fusion*:
fis $R \subseteq$ *ne* ((*fus* R)$\downarrow$)
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* ⇒ (*'a*,*'b*) *mrel* (⨅⨅ - [*80*] *80*) **where**
⨅⨅ *R* ≡ { (*a*,*B*) . *B* = $\bigcap$ { *C* . (*a*,*C*) ∈ *R* } }

declare *co-fusion-def* [*mr-simp*]

lemma *co-fusion-upper-decreasing*:
⨅⨅ *R* ⊑↑ *R*
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *co-fusion-deterministic*:
deterministic (⨅⨅ *R*)
by (*simp add*: *deterministic-set co-fusion-def*)

lemma *co-fusion-greatest*:
assumes *S* ⊑↑ *R*
and *deterministic S*
shows *S* ⊑↑ ⨅⨅ *R*
proof (*clarsimp simp*: *mr-simp*)
fix *a*
from *assms*(*2*) **obtain** *B* **where** *1*: (*a*,*B*) ∈ *S*
by (*meson deterministic-set*)
hence *B* ⊆ $\bigcap$ { *C* . (*a*,*C*) ∈ *R* }
using *assms deterministic-upper* **by** (*smt* (*verit*, *ccfv-threshold*) *Inter-greatest mem-Collect-eq*)
thus ∃ *B* . (∃ *D* . $\bigcap$ { *C* . (*a*,*C*) ∈ *R* } = *B* ∪ *D*) ∧ (*a*,*B*) ∈ *S*
using *1* **by** *blast*
qed

lemma *co-fusion-unique*:
  assumes $\forall R . f\ R \sqsubseteq\uparrow R$
    and $\forall R$. *deterministic* $(f\ R)$
    and $\forall R\ S . S \sqsubseteq\uparrow R \wedge$ *deterministic* $S \longrightarrow S \sqsubseteq\uparrow f\ R$
  shows $f\ T = \bigsqcap\bigsqcap T$
  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*:
  $(\bigsqcap\bigsqcap R)\uparrow = -((-(R\uparrow) \cap A_{\cap\cap})\downarrow)$
proof
  show $(\bigsqcap\bigsqcap R)\uparrow \subseteq -((-(R\uparrow) \cap A_{\cap\cap})\downarrow)$
    apply (*clarsimp simp*: *mr-simp*)
    by *blast*
next
  show $-((-(R\uparrow) \cap A_{\cap\cap})\downarrow) \subseteq (\bigsqcap\bigsqcap R)\uparrow$
  proof (*clarsimp simp*: *mr-simp*)
    fix *a A*
    assume *1*: $\forall B . (\forall C . A \neq B \cap C) \vee (\exists C . (\exists D . B = C \cup D) \wedge (a,C) \in R) \vee (\forall b . B \neq UNIV - \{b\})$
    have $\bigcap\{ C . (a,C) \in R \} \subseteq A$
    proof
      fix x
      assume $x \in \bigcap\{ C . (a,C) \in R \}$
      hence $\forall C . A \neq (UNIV - \{x\}) \cap C$
        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*:
  $(\bigsqcap\bigsqcap R)\downarrow = -((\sim(R\uparrow) \cap A_{\cup\cup})\uparrow)$
proof
  show $(\bigsqcap\bigsqcap R)\downarrow \subseteq -((\sim(R\uparrow) \cap A_{\cup\cup})\uparrow)$
    apply (*clarsimp simp*: *mr-simp*)
    by *blast*
next
  show $-((\sim(R\uparrow) \cap A_{\cup\cup})\uparrow) \subseteq (\bigsqcap\bigsqcap R)\downarrow$
  proof (*clarsimp 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*:
$R \sqsubseteq\uparrow S \Longrightarrow \prod\prod R \sqsubseteq\uparrow \prod\prod S$
by (*meson co-fusion-deterministic co-fusion-greatest co-fusion-upper-decreasing upper-transitive*)

lemma *co-fusion-ii-idempotent*:
$\prod\prod R \cap\cap \prod\prod R = \prod\prod R$
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 $T \Longrightarrow T \cap\cap \prod\prod T = \prod\prod T$
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* ⇒ ($'a$,$'b$) *mrel* ($at_{\cap\cap}$ - [*80*] *80*) **where**
  $at_{\cap\cap}$ R ≡ R↑ ∩ $A_{\cap\cap}$

lemma *co-fission*:
  $at_{\cap\cap}$ R = { (a,B) | a B . (∃ b . −B = {b}) ∧ (∃ C . (a,C) ∈ R ∧ C ⊆ B) }
  apply (*clarsimp simp*: *mr-simp*)
  by *blast*

declare *co-fission* [*mr-simp*]

lemma *co-fission-upper-increasing*:
  R ⊑↑ $at_{\cap\cap}$ R
  by (*fact semilattice-inf-class.inf-le1*)

lemma *co-fission-ic-inner-deterministic*:
  inner-deterministic (∼$at_{\cap\cap}$ R)
  by (*simp add*: *ic-co-atoms ic-dist-oi inner-deterministic-atoms*)

lemma *co-fission-least*:
  assumes R ⊑↑ S
    and *inner-deterministic* (∼S)
  shows $at_{\cap\cap}$ R ⊑↑ S
proof (*clarsimp simp*: *mr-simp*)
  fix a B
  assume *1*: (a,B) ∈ S
  hence (a,−B) ∈ ∼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 ⊆ B ∧ (a,C) ∈ R
    using *assms*(*1*) **by** (*meson upper-less-eq*)
  thus ∃ C . (∃ D . B = C ∪ D) ∧ (∃ E . (∃ D . C = E ∪ D) ∧ (a,E) ∈ R) ∧ (∃ 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 ∀ R . R ⊑↑ f R
    and ∀ R . *inner-deterministic* (∼f R)
    and ∀ R S . R ⊑↑ S ∧ *inner-deterministic* (∼S) ⟶ f R ⊑↑ 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*:
$R \sqsubseteq\uparrow S \Longrightarrow at_{\cap\cap}\ R \sqsubseteq\uparrow at_{\cap\cap}\ S$
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*:
$at_{\cap\cap}\ U = A_{\cap\cap}$
using *top-lower-greatest U-par-idem top-down* **by** *blast*

lemma *co-fission-up*:
$at_{\cap\cap}\ R = at_{\cap\cap}\ (R\uparrow)$
by *simp*

lemma *co-fission-ic-inner-deterministic-fixpoint*:
inner-deterministic $(\sim R) \longleftrightarrow R = at_{\cap\cap}\ R$
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*:
$\prod\prod(at_{\cap\cap}\ R) \sqsubseteq\uparrow R$
proof (*clarsimp simp*: *mr-simp*)
fix $a\ B$
assume *1*: $(a,B) \in R$
have $\bigcap\{\ D\ .\ (\exists E\ .\ (\exists F\ .\ D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b\ .\ D = UNIV - \{b\})\ \} \subseteq B$
proof
fix x
assume *2*: $x \in \bigcap\{\ D\ .\ (\exists E\ .\ (\exists F\ .\ D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b\ .\ D = UNIV - \{b\})\ \}$
show $x \in B$
proof (*rule ccontr*)
let $?D = -\{x\}$
assume *3*: $x \notin B$
hence $B \subseteq ?D$
by *simp*
hence $\bigcap\{\ D\ .\ (\exists E\ .\ (\exists F\ .\ D = E \cup F) \wedge (a,E) \in R) \wedge (\exists b\ .\ D = UNIV - \{b\})\ \} \subseteq ?D$
using *1* **by** (*smt CollectI Compl-eq-Diff-UNIV Inf-lower subset-Un-eq*)
thus *False*
using *2* **by** *auto*
qed

qed
thus $\exists\, C\ .\ B = \bigcap\{\ D\ .\ (\exists\, E\ .\ (\exists\, F\ .\ D = E \cup F) \wedge (a,E) \in R) \wedge (\exists\, b\ .\ D = UNIV - \{b\})\ \} \cup C$
by *auto*
qed

lemma *co-fission-co-fusion-upper-increasing*:
$R \sqsubseteq\uparrow at_{\cap\cap}\ (\prod\prod R)$
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*:
$\prod\prod R \sqsubseteq\uparrow S \longleftrightarrow R \sqsubseteq\uparrow at_{\cap\cap}\ S$
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*:
$\prod\prod R = \prod\prod S \Longrightarrow S \sqsubseteq\uparrow at_{\cap\cap}\ R$
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* ⇒ *bool* **where**
test R ≡ *R* ⊆ *1*

lemma *test*:
test R ⟷ (∀ *a B* . (*a*,*B*) ∈ *R* ⟶ *B* = {*a*})
by (*force simp*: *s-id-def*)

lemma *test-fix*: *test R* ≡ *R* ∩ 1_σ = *R*
by (*simp add*: *le-iff-inf*)

lemma *test-ou-closed*:
test p ⟹ *test q* ⟹ *test* (*p* ∪ *q*)
by (*fact sup-least*)

lemma *test-oi-closed*:
test p ⟹ *test* (*p* ∩ *q*)
by *blast*

abbreviation *test-complement* :: ('*a*,'*a*) *mrel* ⇒ ('*a*,'*a*) *mrel* (≀ - [*80*] *80*) **where**
≀ *R* ≡ −*R* ∩ *1*

lemma *test-complement-closed*:
test (≀ *p*)
by *simp*

lemma *test-double-complement*:
test p ⟷ *p* = ≀ ≀ *p*
by *blast*

lemma *test-complement*:
(*a*,{*a*}) ∈ ≀ *p* ⟷ ¬ (*a*,{*a*}) ∈ *p*
by (*simp add*: *s-id-def*)

declare *test-complement* [*mr-simp*]

lemma *test-complement-antitone*:
assumes *test p*
shows *p* ⊆ *q* ⟷ ≀ *q* ⊆ ≀ *p*
using *assms*(*1*) **by** *blast*

lemma *test-complement-huntington*:
test p ⟹ *p* = ≀ (≀ *p* ∪ ≀ *q*) ∪ ≀ (≀ *p* ∪ *q*)
by *blast*

abbreviation *test-implication* :: ('*a*,'*a*) *mrel* ⇒ ('*a*,'*a*) *mrel* ⇒ ('*a*,'*a*) *mrel*
(**infixl** → *65*) **where**

$p \to q \equiv \wr\ p \cup q$

lemma *test-implication-closed*:
$test\ q \Longrightarrow test\ (p \to q)$
by *simp*

lemma *test-implication*:
$(a,\{a\}) \in p \to q \longleftrightarrow ((a,\{a\}) \in p \longrightarrow (a,\{a\}) \in q)$
by (*simp add*: *s-id-def*)

declare *test-implication* [*mr-simp*]

lemma *test-implication-left-antitone*:
assumes *test p*
shows $p \subseteq r \Longrightarrow r \to q \subseteq p \to q$
by *blast*

lemma *test-implication-right-isotone*:
assumes *test p*
shows $q \subseteq r \Longrightarrow p \to q \subseteq p \to r$
by *blast*

lemma *test-sp-idempotent*:
$test\ p \Longrightarrow p * p = p$
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 (*clarsimp simp*: *mr-simp*)
apply *safe*
apply *blast*
using *assms subid-aux2* **by** *fastforce+*

lemma *sp-test*:
$test\ p \Longrightarrow R * p = R \cap (U * p)$
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 subrelI*)

lemma *sp-test-dist-oi*:
$test\ p \Longrightarrow (R \cap S) * p = (R * p) \cap (S * p)$
by (*smt Int-left-commute semilattice-inf-class.inf.assoc semilattice-inf-class.inf.right-idem sp-test*)

lemma *sp-test-dist-oi-left*:
$test\ p \Longrightarrow (R \cap S) * p = (R * p) \cap S$
by (*smt Int-commute semilattice-inf-class.inf.left-commute sp-test*)

lemma *sp-test-dist-oi-right*:
test p $\Longrightarrow$ $(R \cap S) * p = R \cap (S * p)$
by (*metis semilattice-inf-class.inf.commute sp-test-dist-oi-left*)

lemma *sp-test-sp-oi-left*:
test p $\Longrightarrow$ $(R \cap (U * p)) * T = R * p * T$
by (*metis sp-test*)

lemma *sp-test-sp-oi-right*:
test p $\Longrightarrow$ $R * ((p * U) \cap T) = R * p * T$
by (*metis inf.orderE test-assoc1 test-sp*)

lemma *test-sp-ne*:
test p $\Longrightarrow$ $p *$ *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 $\Longrightarrow$ *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 $(U * p)\downarrow = \{(a,A).\ (a,A) \in U \wedge (\forall a \in A.\ (a,\{a\}) \in p)\} \cap\cap U$
by (*smt* (*verit*) *Collect-cong case-prodI2 case-prod-conv s-prod-test-var*)
also have ... $= \{(a,A).\ \forall a \in A.\ (a,\{a\}) \in p\} \cap\cap U$
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 $\Longrightarrow$ $-(U * p) = (-(U * p))\uparrow$
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 $\Longrightarrow$ $(a,B) \in U * p \longleftrightarrow (\forall b{\in}B\ .\ (b,\{b\}) \in p)$
using *test-fix* **by** (*metis IntE UNIV-I s-prod-test sp-test*)

lemma *oc-top-sp-test*:
  test p ⟹ (*a*,*B*) ∈ −(*U* ∗ *p*) ⟷ (∃ *b*∈*B* . (*b*,{*b*}) ∉ *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*:
  $-1_{\cup\cup}$ ∗ ≀ *p* = *ne* (*U* ∗ ≀ *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* ∗ ≀ *p*))↑
proof (*rule antisym*)
  show −(*U* ∗ *p*) ⊆ (*ne* (*U* ∗ ≀ *p*))↑
  proof
    fix *x*::′*c* × ′*a set*
    assume *1*: *x* ∈ −(*U* ∗ *p*)
    from *this* **obtain** *a B* **where** *2*: *x* = (*a*,*B*)
      by *force*
    from *this* **obtain** *c* **where** *3*: *c* ∈ *B* ∧ (*c*,{*c*}) ∉ *p*
      using *1* **by** (*meson assms oc-top-sp-test*)
    hence *4*: (*a*,{*c*}) ∈ *U* ∗ ≀ *p*
      by (*metis singletonD test-complement test-complement-closed top-sp-test*)
    have (*a*,{*c*}) ∈ $-1_{\cup\cup}$
      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*}) ∈ *ne* (*U* ∗ ≀ *p*)
      using *4* **by** *simp*
    thus *x* ∈ (*ne* (*U* ∗ ≀ *p*))↑
      using *2 3* **by** (*metis* (*no-types*, *lifting*) *U-par-st singletonD subset-eq*)
  qed
next
  have (*U* ∗ *p*)↓ = *U* ∗ *p*
    using *assms top-sp-test-down-closed* **by** *auto*
  also have ... ⊆ −($-1_{\cup\cup}$ ∗ ≀ *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* ∗ ≀ *p*)
    by (*simp add*: *oc-top-sp-test-0*)
  finally have *U* ∗ *p* ⊆ −((*ne* (*U* ∗ ≀ *p*))↑)
    by (*simp add*: *down-double-complement-up*)
  thus (*ne* (*U* ∗ ≀ *p*))↑ ⊆ −(*U* ∗ *p*)
    by *auto*
qed

lemma *oc-top-sp-test-2*:
  test p $\Longrightarrow -(U * p) = (-1_{\cup\cup} * \wr\ p)\uparrow$
  by (*simp add*: *oc-top-sp-test-1 oc-top-sp-test-0*)

lemma *split-sp-test*:
  assumes *test p*
  shows $R = (R * p) \cup (ne\ R \cap (ne\ (R\downarrow * \wr\ p))\uparrow)$
proof (*rule antisym*)
  show $R \subseteq (R * p) \cup (ne\ R \cap (ne\ (R\downarrow * \wr\ p))\uparrow)$
  proof
    fix *x*
    assume *1*: $x \in R$
    from *this* **obtain** *a B* **where** *2*: $x = (a,B)$
      by *force*
    show $x \in (R * p) \cup (ne\ R \cap (ne\ (R\downarrow * \wr\ p))\uparrow)$
    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 $(a,\{b\}) \in R\downarrow * \wr\ p$
        using *3* **by** (*metis s-prod-test-aux2 singletonD test-complement*)
      hence $(a,\{b\}) \in ne\ (R\downarrow * \wr\ p)$
        by (*simp add*: *non-empty*)
      hence $(a,B) \in (ne\ (R\downarrow * \wr\ p))\uparrow$
        using *3* **by** (*meson U-par-st*)
      thus *?thesis*
        using *1 2 3 non-empty* **by** *auto*
    qed
  qed
next
  show $(R * p) \cup (ne\ R \cap (ne\ (R\downarrow * \wr\ p))\uparrow) \subseteq R$
    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 $\Longrightarrow ne\ p = p$

using *inner-deterministic-sp-unit* **by** *blast*

lemma *ne-test-up*:
test p $\Longrightarrow$ *ne* $(p\uparrow) = p\uparrow$
by (*metis atoms-solution ne-equality test-ne up-idempotent up-isotone*)

lemma *ne-sp-test-up*:
test p $\Longrightarrow$ $(ne\ (R * p))\uparrow = ne\ R * p\uparrow$
using *test-fix* **by** (*smt ne-up sp-test-dist-oi-left test-assoc1 test-ne*)

lemma *ne-down-sp-test-up*:
test p $\Longrightarrow$ *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 $\Longrightarrow$ $p\uparrow = p * 1\uparrow$
by (*metis ne-up test-ne*)

lemma *top-test-oi-top-complement*:
test p $\Longrightarrow$ $(U * p) \cap (U * \wr\ p) = 1_{\cup\cup}$
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 $\Longrightarrow$ $(R * p) \cap (R * \wr\ p) = R \cap 1_{\cup\cup}$
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* $(U * p) * \wr\ p = \{\}$
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 $-(p * U) = \wr\ p * U$
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 $\ldots = -\{(a,A).\ (a,\{a\}) \in p\}$
using *top-upper-least* **by** *auto*
also have $\ldots = \{(a,A).\ (a,\{a\}) \notin p\}$
by *force*
also have $\ldots = \{(a,A).\ (a,\{a\}) \in \wr\ p\}$
by (*meson test-complement*)
also have $\ldots = \{(a,A).\ (a,\{a\}) \in \wr\ p \wedge (a,A) \in U\}$
using *U-par-idem top-upper-least* **by** *auto*

```
  also have ... = ≀ p ∗ U
    by (simp add: test-s-prod-var)
  finally show ?thesis
    .
qed

lemma top-sp-test-shunt:
  assumes test p
  shows R ⊆ U ∗ p ⟶ R ∗ ≀ p ⊆ 1∪∪
  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↓ ⊆ U ∗ p ⟷ R↓ ∗ ≀ p ⊆ 1∪∪
proof
  assume R↓ ⊆ U ∗ p
  thus R↓ ∗ ≀ p ⊆ 1∪∪
    using assms top-sp-test-shunt by blast
next
  assume 1: R↓ ∗ ≀ p ⊆ 1∪∪
  have R ⊆ U ∗ p
  proof
    fix x
    assume x ∈ R
    from this obtain a B where 2: x = (a,B) ∧ x ∈ R
      by force
    have ∀ b∈B . (b,{b}) ∈ p
    proof
      fix b
      assume 3: b ∈ B
      have (b,{b}) ∉ ≀ p
      proof
        assume 4: (b,{b}) ∈ ≀ p
        have (a,{b}) ∈ R↓
          using 2 3 down by fastforce
        hence (a,{b}) ∈ R↓ ∗ ≀ 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}) ∈ p
        by (meson test-complement)
    qed
    thus x ∈ U ∗ p
      using 2 by (simp add: assms top-sp-test)
  qed
  thus R↓ ⊆ U ∗ p
```

using *assms top-sp-test-down-iff-1* **by** *blast*
qed

lemma *top-sp-test-down-iff-3*:
$R\downarrow * \wr\ p \subseteq 1_{\cup\cup} \longleftrightarrow ne\ (R\downarrow) * \wr\ p \subseteq \{\}$
by (*simp add*: *disjoint-eq-subset-Compl ne-sp-test*)

lemma *top-sp-test-down-iff-4*:
assumes *test p*
shows $R\downarrow \cap (U * \wr\ p) \subseteq 1_{\cup\cup} \longleftrightarrow R\downarrow \subseteq 1_{\cup\cup} \cup (U * p)$
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*:
$t \subseteq 1 \cup 1_{\cup\cup} \longleftrightarrow (\exists p\ q\ .\ p \subseteq 1 \wedge q \subseteq 1_{\cup\cup} \wedge t = p \cup q)$
by (*meson subset-UnE sup.mono*)

lemma *test-iu-test-sp-assoc-1*:
$t \subseteq 1 \cup 1_{\cup\cup} \Longrightarrow t * (R * S) = (t * R) * S$
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*:
$t \subseteq 1_{\cup\cup} \Longrightarrow R * (t * S) = (R * t) * S$
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 $\ldots = (R * t) * \{\}$
by (*metis cl5 s-prod-idl*)
also have $\ldots \subseteq (R * t) * S$
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 $?g = \lambda b\ .\ if\ (b,\{b\}) \in t \wedge (b,\{\}) \notin t\ then\ \{b\}\ else\ \{\}$
  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 $\forall b{\in}B\ .\ \exists D\ .\ (b,D) \in t \wedge (\exists g\ .\ (\forall e{\in}D\ .\ (e,g\ e) \in S) \wedge f\ b = \bigcup\{\ g\ e \mid e\ .\ e \in D\ \})$
      by (*simp add*: *mr-simp Setcompr-eq-image*)
    hence $\exists Db\ .\ \forall b{\in}B\ .\ (b,Db\ b) \in t \wedge (\exists g\ .\ (\forall e{\in}Db\ b\ .\ (e,g\ e) \in S) \wedge f\ b = \bigcup\{\ g\ e \mid e\ .\ e \in Db\ b\ \})$
      by (*rule bchoice*)
    from *this* **obtain** Db **where** *2*: $\forall b{\in}B\ .\ (b,Db\ b) \in t \wedge (\exists g\ .\ (\forall e{\in}Db\ b\ .\ (e,g\ e) \in S) \wedge f\ b = \bigcup\{\ g\ e \mid e\ .\ e \in Db\ b\ \})$
      by *auto*
    let $?D = \bigcup\{\ Db\ b \mid b\ .\ b \in B\ \}$
    have $\forall b{\in}B\ .\ (b,Db\ b) \in t$
      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** (*clarsimp 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*:
$t \subseteq 1_{\cup\cup} \Longrightarrow R * (S * t) = (R * S) * t$
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*: $x = (a,C) \wedge (a,B) \in R \wedge (\forall\ b{\in}B\ .\ (b,f\ b) \in S * t) \wedge C = \bigcup\{\ f\ b \mid b\ .\ b{\in}B\ \}$
by (*clarsimp 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 (*clarsimp simp*: *mr-simp Setcompr-eq-image*)

hence ∃ *Db* . ∀ *b*∈*B* . (*b*,*Db b*) ∈ *S* ∧ (∃ *g* . (∀ *e*∈*Db b* . (*e*,*g e*) ∈ *t*) ∧ *f b* = ⋃{ *g e* | *e* . *e* ∈ *Db b* })
by (*rule bchoice*)
from *this* **obtain** *Db* **where** *2*: ∀ *b*∈*B* . (*b*,*Db b*) ∈ *S* ∧ (∃ *g* . (∀ *e*∈*Db b* . (*e*,*g e*) ∈ *t*) ∧ *f b* = ⋃{ *g e* | *e* . *e* ∈ *Db b* })
by *auto*
hence ∃ *gb* . ∀ *b*∈*B* . (∀ *e*∈*Db b* . (*e*,*gb b e*) ∈ *t*) ∧ *f b* = ⋃{ *gb b e* | *e* . *e* ∈ *Db b* }
by (*auto intro*: *bchoice*)
from *this* **obtain** *gb* **where** *3*: ∀ *b*∈*B* . (∀ *e*∈*Db b* . (*e*,*gb b e*) ∈ *t*) ∧ *f b* = ⋃{ *gb b e* | *e* . *e* ∈ *Db b* }
by *auto*
let *?g* = λ*x* . ⋃{ *gb b x* | *b* . *b* ∈ *B* ∧ *x* ∈ *Db b* }
have *4*: ∀ *b*∈*B* . ∀ *e*∈*Db b* . *gb b e* = {} ∨ *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 ∀ *e* . *?g e* ⊆ {*e*}
by *auto*
hence *5*: ∀ *e* . *?g e* = {} ∨ *?g e* = {*e*}
by *auto*
let *?D* = ⋃{ *Db b* | *b* . *b* ∈ *B* }
have *6*: (*a*,*?D*) ∈ *R* ∗ *S*
apply (*unfold s-prod-def*)
using *1 2* **by** *blast*
have *7*: ∀ *b*∈*?D* . (*b*,*?g b*) ∈ *t*
proof
fix *e*
assume *e* ∈ *?D*
from *this* **obtain** *b* **where** *8*: *b* ∈ *B* ∧ *e* ∈ *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* ∈ *B* ∧ *e* ∈ *Db bb* ∧ *e* ∈ *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 = ⋃{ ?g e | e . e ∈ ?D }
proof
  show C ⊆ ⋃{ ?g e | e . e ∈ ?D }
  proof
    fix y
    assume y ∈ C
    from this obtain b where 12: b ∈ B ∧ y ∈ f b
      using 1 by auto
    from this 3 obtain e where e ∈ Db b ∧ y ∈ gb b e
      by auto
    thus y ∈ ⋃{ ?g e | e . e ∈ ?D }
      using 4 12 by auto
  qed
next
  show ⋃{ ?g e | e . e ∈ ?D } ⊆ C
  proof
    fix y
    assume y ∈ ⋃{ ?g e | e . e ∈ ?D }
    from this obtain b e where 13: b ∈ B ∧ e ∈ Db b ∧ y ∈ gb b e
      by auto
    hence y ∈ f b
      using 3 by auto
    thus y ∈ C
      using 1 13 by auto
  qed
qed
have (a,C) ∈ (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 ∈ (R ∗ S) ∗ t
  using 1 by simp
qed
next
  show (R ∗ S) ∗ t ⊆ 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) ⊆ (t ∗ R) ∗ S
  proof
    fix x
    assume x ∈ t ∗ (R ∗ S)
    from this obtain a B D f where 1: x = (a,D) ∧ (a,B) ∈ t ∧ (∀ b∈B . (b,f b)
∈ R ∗ S) ∧ D = ⋃{ f b | b . b∈B }
      by (simp add: mr-simp) blast
    have (a,D) ∈ (t ∗ R) ∗ S
    proof (cases (a,B) ∈ 1∪∪)
      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) ∈ t ∗ (R ∗ {})
        using 1 ‹(B::'b::type set) = {}› s-prod-def by fastforce
      hence (a,D) ∈ (t ∗ R) ∗ {}
        by (metis cl5 s-prod-idl)
      thus ?thesis
        using s-prod-isor by auto
    next
      case False
      hence (a,B) ∈ A∪∪
        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 ∧ (b,f b) ∈ R ∗ S
        using 1 by auto
      from this obtain C g where 3: (b,C) ∈ R ∧ (∀ c∈C . (c,g c) ∈ S) ∧ D =
⋃{ g c | c . c∈C }
        by (clarsimp simp: mr-simp) blast
      have (a,C) ∈ 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 ∈ (t ∗ R) ∗ S
      using 1 by auto
  qed
next
  show (t ∗ R) ∗ S ⊆ 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*:
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 $R \cap -(U * S) = R \cap -(U * S)$
using *iu-unit-below-top-sp-test* **by** *auto*

lemma *schroeder-test*:
assumes *test p*
shows $R * p \subseteq S \longleftrightarrow -S * p \subseteq -R$
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 $\Longrightarrow -p * p \subseteq -1$
by (*simp add*: *schroeder-test*)

lemma *test-sp-commute*:
test p $\Longrightarrow$ *test q* $\Longrightarrow p * q = q * p$
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 $p * q \subseteq r \longleftrightarrow p \subseteq r \cup \wr\, q$
proof
assume *1*: $p * q \subseteq r$
have $p = p * q \cup p * \wr\, q$
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 ... $\subseteq r \cup \wr\, q$
using *1* **by** (*metis assms*(*1*) *inf-le2 sup.mono test-sp*)
finally show $p \subseteq r \cup \wr\, q$
.
next
assume $p \subseteq r \cup \wr\, q$
hence $p * q \subseteq p * r$
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 ... $\subseteq r$
using *assms*(*1*) *test-sp* **by** *auto*
finally show $p * q \subseteq r$

.
qed

lemma *test-sp-is-iu*:
test p ⟹ *test q* ⟹ *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 (*clarsimp simp*: *mr-simp*)
apply (*rule Int-greatest*)
apply (*clarsimp simp*: *mr-simp*)
by (*clarsimp 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*:
  $\forall p{\in}X$. *test* $p \Longrightarrow$ *test* $(\bigcup X)$
  by *blast*

lemma *test-oI-closed*:
  $\exists p{\in}X$. *test* $p \Longrightarrow$ *test* $(\bigcap X)$
  by *blast*

lemma *sp-test-dist-oI*:
  assumes *test* p
    and $X \neq \{\}$
  shows $(\bigcap X) * p = (\bigcap R{\in}X\ .\ R * p)$
  apply (*rule antisym*)
   apply (*clarsimp simp*: *mr-simp*)
   apply *blast*
  apply (*clarsimp simp*: *mr-simp*)
  subgoal for a C **proof** −
    assume *1*: $\forall R{\in}X\ .\ \exists B\ .\ (a,B) \in R \wedge (\exists f\ .\ (\forall b{\in}B\ .\ (b,f\ b) \in p) \wedge C = \bigcup(f$
$`\ B))$
    have *2*: $(\forall R{\in}X\ .\ (a,C) \in R \wedge (\forall b{\in}C\ .\ (b,\{b\}) \in p))$
    proof
      fix R
      assume $R \in X$
      from *this* **obtain** B **where** *3*: $(a,B) \in R \wedge (\exists f\ .\ (\forall b{\in}B\ .\ (b,f\ b) \in p) \wedge C$
$= \bigcup(f\ `\ B))$
        using *1* **by** *force*
      from *this* **obtain** f **where** *4*: $\forall b{\in}B\ .\ (b,f\ b) \in p$ **and** *5*: $C = \bigcup(f\ `\ B)$
        by *auto*
      hence *6*: $\forall b{\in}B\ .\ f\ b = \{b\}$
        using *assms*(*1*) *test* **by** *blast*
      hence *7*: $C = B$
        using *5* **by** *auto*
      hence $(a,C) \in R$
        using *3* **by** *auto*
      thus $(a,C) \in R \wedge (\forall b{\in}C\ .\ (b,\{b\}) \in p)$
        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* - $\lambda x\ .\ \{x\}$])
      apply (*rule conjI*)
      using *2* *assms*(*2*) **apply** *blast*
      by *simp*
  qed
  done

lemma *test-iU-is-iI*:
  assumes $\forall\, i{\in}I\ .\ test\ (X\ i)$
    and $I \neq \{\}$
  shows $\bigcup\bigcup X|I = \bigcap\bigcap X|I$
  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 (*clarsimp simp*: *mr-simp*)
  by (*metis* (*no-types*, *opaque-lifting*) *INF-eq-const SUP-eq-const assms test*)

lemma *test-iU-is-oI*:
  assumes $\forall\, i{\in}I\ .\ test\ (X\ i)$
    and $I \neq \{\}$
  shows $\bigcup\bigcup X|I = \bigcap(X\ `\ I)$
  apply (*rule antisym*)
   apply (*clarsimp simp*: *mr-simp*)
   apply (*metis* (*no-types*, *opaque-lifting*) *SUP-eq-const assms test*)
  apply (*clarsimp 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**
  $aDom\ R \equiv \wr\ Dom\ R$

lemma *ad-set*: $aDom\ R = \{(a,\{a\})|\ a.\ \neg(\exists\, A.\ (a,A) \in R)\}$
  by (*clarsimp 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 $\ldots \subseteq -((R * 1_{\cup\cup}) \cup\cup\ 1) \cap 1$
by (*metis c6 convex-increasing iu-commute iu-isotone iu-unit sp-unit-convex test-complement-antitone upper-iu-increasing*)
also have $\ldots = aDom\ R$
by (*simp add*: *d-def-expl*)
finally show $-((R * 1_{\cup\cup})\uparrow) \cap 1 \subseteq aDom\ R$
using $\langle\wr\ ((R::('a::type \times 'b::type\ set)\ set) * 1_{\cup\cup})\uparrow \subseteq \wr\ (R * 1_{\cup\cup} \cup\cup\ 1)\rangle$ **by** *blast*
have $aDom\ R = \{(a,\{a\})\ |a.\ \neg(\exists B.\ (a,B) \in R)\}$
by (*simp add*: *ad-set*)
also have $\ldots \subseteq \{(a,\{a\})\ |a.\ (a,\{a\}) \notin (R * (p\text{-}id::('b,'a)\ mrel))\uparrow\}$
by (*simp add*: *U-par-st domain-pointwise*)
also have $\ldots = \{(a,\{a\})\ |a.\ (a,\{a\}) \in -(R * (p\text{-}id::('b,'a)\ mrel))\uparrow \cap 1\}$
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 $Dom\ (R \cap -(U * p)) = Dom\ (ne\ (R\downarrow) * \wr\ p)$
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 (*clarsimp 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 (*clarsimp 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 ⊆ *S* ⟹ *Dom R* ⊆ *Dom S*
unfolding *Dom-def* **by** *blast*

lemma *ad-antitone*:
R ⊆ *S* ⟹ *aDom S* ⊆ *aDom R*
by (*simp add*: *Int-commute d-isotone semilattice-inf-class.le-infI2*)

lemma *d-dist-ou*:
Dom (*R* ∪ *S*) = *Dom R* ∪ *Dom S*
unfolding *Dom-def* **by** *blast*

lemma *d-dist-iu*:
Dom (*R* ∪∪ *S*) = *Dom R* ∗ *Dom S*
by (*clarsimp simp*: *mr-simp*) *auto*

lemma *d-dist-ii*:
Dom (*R* ∩∩ *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* ∗ *Dom S*) = *Dom* (*R* ∗ *S*)
apply (*clarsimp simp*: *mr-simp*)
apply *safe*
apply *metis*
by (*metis UN-singleton*)

lemma *ad-loc*:
aDom (*R* ∗ *Dom S*) = *aDom* (*R* ∗ *S*)
by *simp*

lemma *d-ne-down*:
Dom (*ne* (*R*↓)) = *Dom* (*ne R*)
by (*metis atoms-solution d-down-oi-up-1 d-down-oi-up-2*)

lemma *ne-sp-iu-unit-up*:
ne R = *R* ⟹ (*R* ∗ $1_{\cup\cup}$)↑ = *R* ∗ *U*
apply (*clarsimp simp*: *mr-simp*)
apply *safe*
apply (*metis* (*no-types*, *lifting*) *Compl-iff IntE Inter-iff UNIV-I UN-simps*(*2*) *image-eqI singletonI*)
apply *clarsimp*
by (*metis SUP-bot sup-bot-left*)

lemma *ne-d-expl*:
ne R = *R* ⟹ *Dom R* = *R* ∗ *U* ∩ *1*
by (*metis cl8-var d-def-expl d-test ne-sp-iu-unit-up test-sp*)

lemma *ne-a-expl*:
ne R = *R* ⟹ *aDom R* = −(*R* ∗ *U*) ∩ *1*

by (*simp add*: *ad-expl-2 ne-sp-iu-unit-up*)

lemma *d-dist-oU*:
Dom (⋃ *X*) = ⋃(*Dom* ‘ *X*)
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *d-dist-iU-iI*:
Dom (⋃⋃ *X*|*I*) = *Dom* (⋂⋂ *X*|*I*)
by (*clarsimp simp*: *mr-simp*)

lemma *d-dist-iU-oI*:
assumes *I* ≠ {}
shows *Dom* (⋃⋃ *X*|*I*) = ⋂(*Dom* ‘ *X* ‘ *I*)
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply *blast*
apply (*clarsimp 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 $x \in Q \oslash R$
using *2 sp-lres-def* **by** *auto*
qed
next
assume *3*: $S \subseteq Q \oslash R$
have $(Q \oslash R) * R \subseteq Q$
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** $(a,C) \in Q \oslash R \wedge (\exists g . (\forall c{\in}C . (c,g\ c) \in R) \wedge D = \bigcup\{ g\ c \mid c . c \in C \})$
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*:
$\{\} \oslash R = \{\} \oslash Dom\ R$
by (*metis d-sp-strict order-refl sp-lres-galois subset-antisym subset-empty*)

lemma *bot-sp-lres-expl*:
$\{\} \oslash R = -(U * Dom\ R)$
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*:
$Q \subseteq S \Longrightarrow Q \oslash R \subseteq S \oslash R$
by (*meson dual-order.refl sp-lres-galois subset-trans*)

lemma *sp-lres-right-antitone*:
$S \subseteq R \Longrightarrow Q \oslash R \subseteq Q \oslash S$
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 (*clarsimp 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*:
$aDom\ R = -(U * Dom\ R) \cap 1$
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 ... $\longleftrightarrow ne\ (S{\downarrow}) * Dom\ (ne\ (R{\downarrow}) * \wr\ p) = \{\}$
by (*simp add*: *assms d-ne-down-dp-complement-test*)
also have ... $\longleftrightarrow ne\ (S{\downarrow}) * (ne\ (R{\downarrow}) * \wr\ p) = \{\}$
using *d-sp-strict* **by** *auto*
also have ... $\longleftrightarrow ne\ (S{\downarrow}) * ne\ (R{\downarrow}) * \wr\ p = \{\}$
by (*simp add*: *test-assoc3*)
also have ... $\longleftrightarrow ne\ ((S{\downarrow} * R){\downarrow}) * \wr\ p = \{\}$
by (*simp add*: *down-dist-sp ne-dist-down-sp*)
also have ... $\longleftrightarrow S{\downarrow} * R \subseteq U * p$
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 ... $\longleftrightarrow S{\downarrow} \subseteq (U * p) \oslash R$
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 $(((U * p) \oslash R) \cap 1)\downarrow * R = (((U * p) \oslash R) \cap 1) * (1_{\cup\cup} \cup R)$
    using *down-sp-sp* **by** *blast*
  also have ... $= (((U * p) \oslash R) \cap 1) * 1_{\cup\cup} \cup (((U * p) \oslash R) \cap 1) * R$
    by (*simp add*: *s-distl-test*)
  also have ... $\subseteq 1_{\cup\cup} \cup (((U * p) \oslash R) \cap 1) * R$
    using *c6* **by** *auto*
  also have ... $\subseteq 1_{\cup\cup} \cup ((U * p) \oslash R) * R$
    by (*metis Un-Int-eq*(*4*) *inf-le2 iu-unit-convex iu-unit-down sp-oi-subdist sup.mono*)
  also have ... $\subseteq 1_{\cup\cup} \cup U * p$
    using *sp-lres-sp-below* **by** *auto*
  also have ... $= U * p$
    by (*simp add*: *iu-unit-below-top-sp-test sup.absorb2*)
  finally have $(((U * p) \oslash R) \cap 1)\downarrow \subseteq (U * p) \oslash R$
    by (*simp add*: *sp-lres-galois*)
  hence $((U * p) \oslash R) \cap 1 \subseteq U * aDom\ (R \cap -(U * p))$
    by (*metis assms top-test-sp-lres-total-expl-1*)
  thus $((U * p) \oslash R) \cap 1 \subseteq aDom\ (R \cap -(U * p))$
    by (*metis* (*no-types*, *lifting*) *inf.idem inf.orderE inf-commute sp-oi-subdist sp-test test-s-prod-is-meet*)
next
  have $aDom\ (R \cap -(U * p)) = U * aDom\ (R \cap -(U * p)) \cap 1$
    by (*metis* (*no-types*, *lifting*) *inf.absorb-iff2 inf.idem inf-commute inf-le2 sp-test test-s-prod-is-meet*)
  thus $aDom\ (R \cap -(U * p)) \subseteq ((U * p) \oslash R) \cap 1$
    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 $aDom\ (ne\ (R\downarrow) * \wr\ p) = ((U * p) \oslash R) \cap 1$
  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 $(R \cap -(U * p)) * U = (R\downarrow \cap -(U * p)) * U$
proof (*rule antisym*)
  show $(R \cap -(U * p)) * U \subseteq (R\downarrow \cap -(U * p)) * U$
    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 $R\downarrow \cap -(U * p) \subseteq (R \cap -(U * p))\downarrow$

by (*metis oc-top-sp-test-up-closed down-oi-up-closed assms*)
also have ... $\subseteq (R \cap -(U * p)) * U$
by (*simp add*: *down-below-sp-top*)
finally show $(R{\downarrow} \cap -(U * p)) * U \subseteq (R \cap -(U * p)) * U$
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 $(R{\downarrow} \cap -(U * p)) * U = ne\ (R{\downarrow}) * \wr\ p * U$
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 ... $= Dom\ (ne\ (R{\downarrow}) * \wr\ p) * U$
by (*simp add*: *assms d-ne-down-dp-complement-test*)
also have ... $= Dom\ (ne\ (R{\downarrow} * \wr\ p)) * U$
by (*simp add*: *ne-sp-test*)
also have ... $= ne\ (R{\downarrow} * \wr\ p) * U$
by (*simp add*: *cl8-var ne-sp-iu-unit-up*)
also have ... $= ne\ (R{\downarrow}) * \wr\ p * U$
by (*simp add*: *ne-sp-test*)
finally show $(R{\downarrow} \cap -(U * p)) * U \subseteq ne\ (R{\downarrow}) * \wr\ p * U$
.
next
have $ne\ (R{\downarrow}) * \wr\ p \subseteq -(U * p)$
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 $ne\ (R{\downarrow}) * \wr\ p * U \subseteq (R{\downarrow} \cap -(U * p)) * U$
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 $\Longrightarrow R \subseteq R * p \cup ne\ (R{\downarrow}) * (\wr\ p){\uparrow}$
proof –
assume *test p*
hence $R \subseteq R * p \cup (ne\ (R{\downarrow} * \wr\ p)){\uparrow}$
by (*smt* (*verit*, *best*) *IntE UnCI UnE split-sp-test subsetI*)
thus $R \subseteq R * p \cup ne\ (R{\downarrow}) * (\wr\ p){\uparrow}$

by (*simp add*: *ne-sp-test-up*)
qed

lemma *split-sp-test-3*:
test p $\Longrightarrow R \subseteq R * p \cup R{\downarrow} * (\wr\ p){\uparrow}$
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 $R * (p \cup q) \subseteq R * p \cup ne\ (R{\downarrow}) * q{\uparrow}$
proof −
have *1*: $(p \cup q) * p = p$
by (*metis Un-Int-eq*(*1*) *assms sp-test-dist-oi-left test-ou-closed test-sp-commute test-sp-idempotent*)
have $(R * (p \cup q)){\downarrow} * \wr\ p \subseteq R{\downarrow} * q$
proof −
have $(R * (p \cup q)){\downarrow} * \wr\ p = R * (p \cup q) * (1_{\cup\cup} \cup \wr\ p)$
using *down-sp-sp* **by** *blast*
also have ... $= R * ((p \cup q) * 1_{\cup\cup} \cup (p \cup q) * \wr\ p)$
by (*smt* (*verit*) *assms inf.orderE s-distl-test test-assoc1 test-ou-closed*)
also have ... $\subseteq R * (1_{\cup\cup} \cup (p \cup q) * \wr\ p)$
by (*meson c6 s-prod-isor subset-refl sup.mono*)
also have ... $= R{\downarrow} * ((p \cup q) * \wr\ p)$
using *down-sp-sp* **by** *blast*
also have ... $= R{\downarrow} * (p * \wr\ p \cup q * \wr\ p)$
by (*metis assms ii-right-dist-ou inf-commute inf-le1 test-ou-closed test-sp-is-ii*)
also have ... $= R{\downarrow} * (q * \wr\ p)$
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 ... $\subseteq R{\downarrow} * q$
by (*metis inf-le1 inf-le2 s-prod-isor sp-test*)
finally show *?thesis*
.
qed
hence *2*: $ne\ ((R * (p \cup q)){\downarrow}) * \wr\ p \subseteq ne\ (R{\downarrow}) * q$
by (*metis assms*(*2*) *inf-le2 ne-dist-ou ne-sp-test subset-Un-eq*)
have $R * (p \cup q) \subseteq R * (p \cup q) * p \cup ne\ ((R * (p \cup q)){\downarrow}) * (\wr\ p){\uparrow}$
by (*simp add*: *assms split-sp-test-2*)
also have ... $= R * p \cup ne\ ((R * (p \cup q)){\downarrow}) * (\wr\ p){\uparrow}$
using *1* **by** (*metis assms*(*1*) *inf.orderE test-assoc3*)
also have ... $\subseteq R * p \cup ne\ (R{\downarrow}) * q{\uparrow}$
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 $R * (p \cup q) \subseteq R * p \cup R{\downarrow} * q{\uparrow}$
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 $Dom\ (R * (p \cup q)) \subseteq Dom\ (R * p \cup ne\ (R{\downarrow}) * q)$
proof −
  have $Dom\ (R * (p \cup q)) \subseteq Dom\ (R * p \cup ne\ (R{\downarrow}) * q{\uparrow})$
    by (*simp add*: *assms d-isotone split-sp-test-4*)
  also have ... $= Dom\ (R * p \cup (ne\ (R{\downarrow}) * q){\uparrow})$
    by (*simp add*: *assms*(*2*) *ne-sp-test ne-sp-test-up*)
  also have ... $\subseteq Dom\ (R * p \cup ne\ (R{\downarrow}) * q)$
    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 $Dom\ (ne\ (R{\downarrow}) * (p \cup q)) = Dom\ (ne\ (R{\downarrow}) * p \cup ne\ (R{\downarrow}) * q)$
  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*:
  $test\ p \Longrightarrow p * (R \cup\cup S) = p * R \cup\cup S$
  by (*metis cl8-var inf.orderE p-prod-assoc s-subid-iff2*)

lemma *test-sp-left-dist-iu-2*:
  $test\ p \Longrightarrow p * (R \cup\cup S) = R \cup\cup p * S$
  by (*metis iu-commute test-sp-left-dist-iu-1*)

lemma *d-sp-below-iu-down*:
  $Dom\ R * S \subseteq (R \cup\cup S){\downarrow}$
  by (*simp add*: *cl8-var iu-lower-left-isotone sp-iu-unit-lower*)

lemma *d-sp-ne-down-below-ne-iu-down*:
  Dom R $*$ *ne* $(S\downarrow) \subseteq$ *ne* $((R \cup\cup S)\downarrow)$
proof –
  have *Dom R* $* S\downarrow \subseteq (R \cup\cup S)\downarrow$
    by (*simp add*: *cl8-var iu-lower-isotone sp-iu-unit-lower*)
  hence *ne* (*Dom R* $* S\downarrow) \subseteq$ *ne* $((R \cup\cup S)\downarrow)$
    by *blast*
  thus *?thesis*
    by (*smt d-test test-sp-ne*)
qed

lemma *top-test*:
  test p $\Longrightarrow U * p = \{ (a,B) . (\forall b{\in}B . (b,\{b\}) \in p) \}$
  apply (*unfold test*)
  apply (*clarsimp simp*: *mr-simp*)
  by *fastforce*

lemma *iu-oi-complement-top-test-ou-up*:
  test p $\Longrightarrow (R \cup\cup S) \cap -(U * p) \subseteq ((R \cup S) \cap -(U * p))\uparrow$
  apply (*unfold top-test*)
  apply (*clarsimp simp*: *mr-simp*)
  by *blast*

lemma *d-ne-iu-down-sp-test-ou*:
  assumes *test p*
  shows *Dom* (*ne* $((R \cup\cup S)\downarrow) * p) \subseteq$ *Dom* ((*ne* $(R\downarrow) \cup$ *ne* $(S\downarrow)) * p)$
proof –
  have *Dom* (*ne* $((R \cup\cup S)\downarrow) * p) =$ *Dom* $((R \cup\cup S) \cap -(U * \wr p))$
    by (*metis assms d-ne-down-dp-complement-test test-double-complement*)
  also have ... $\subseteq$ *Dom* $((R \cup S) \cap -(U * \wr p))$
    by (*metis iu-oi-complement-top-test-ou-up d-isotone d-up*
semilattice-inf-class.inf-le2)
  also have ... $=$ *Dom* (*ne* $((R \cup S)\downarrow) * p)$
    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 $I \neq \{\}$
  shows $p * (\bigcup\bigcup X|I) = \bigcup\bigcup (\lambda i . p * X\ i)|I$
  apply (*rule antisym*)
   apply (*clarsimp simp*: *mr-simp*)
  subgoal for *a B f* **proof** –
    assume *1*: $(a,B) \in p$
    hence *2*: $B = \{a\}$
      by (*metis assms*(*1*) *test*)

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** *3*: $f\ a = \bigcup(g\ `\ I) \wedge (\forall i{\in}I\ .\ (a,g\ i) \in X\ i)$
using *2* **by** *auto*
have $\bigcup(f\ `\ B) = \bigcup(g\ `\ I) \wedge (\forall i{\in}I\ .\ \exists B\ .\ (a,B) \in p \wedge (\exists f\ .\ (\forall b{\in}B\ .\ (b,f\ b) \in X\ i) \wedge g\ i = \bigcup(f\ `\ B)))$
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* - $\lambda b\ .\ g\ i$])
using *2 3* **by** *blast*
done
thus *?thesis*
by *auto*
qed
apply (*clarsimp simp*: *mr-simp*)
subgoal for $a\ f$ **proof** −
assume *4*: $\forall i{\in}I\ .\ \exists B\ .\ (a,B) \in p \wedge (\exists g\ .\ (\forall b{\in}B\ .\ (b,g\ b) \in X\ i) \wedge f\ i = \bigcup(g\ `\ B))$
have $(a,\{a\}) \in p \wedge (\exists g\ .\ (\forall b{\in}\{a\}\ .\ \exists f\ .\ g\ b = \bigcup(f\ `\ I) \wedge (\forall i{\in}I\ .\ (b,f\ i) \in X\ i)) \wedge \bigcup(f\ `\ I) = \bigcup(g\ `\ \{a\}))$
apply (*rule conjI*)
using *4* **apply** (*metis assms equals0I test*)
apply (*rule exI*[*of* - $\lambda a\ .\ \bigcup(f\ `\ I)$])
apply *clarsimp*
apply (*rule exI*[*of* - f])
using *4 assms*(*1*) *test* **by** *fastforce*
thus *?thesis*
by *auto*
qed
done

8.4 Modal operations

definition *adia* :: $('a,'b)\ mrel \Rightarrow ('b,'b)\ mrel \Rightarrow ('a,'a)\ mrel$ ($|$ - $\rangle$ - [*50*,*90*] *95*)
where
$|R\rangle p \equiv \{\ (a,\{a\})\ |\ a\ .\ \exists B\ .\ (a,B) \in R \wedge (\forall b{\in}B\ .\ (b,\{b\}) \in p)\ \}$

definition *abox* :: $('a,'b)\ mrel \Rightarrow ('b,'b)\ mrel \Rightarrow ('a,'a)\ mrel$ ($|$ -] - [*50*,*90*] *95*)
where
$|R]p \equiv \{\ (a,\{a\})\ |\ a\ .\ \forall B\ .\ (a,B) \in R \longrightarrow (\forall b{\in}B\ .\ (b,\{b\}) \in p)\ \}$

definition *edia* :: $('a,'b)\ mrel \Rightarrow ('b,'b)\ mrel \Rightarrow ('a,'a)\ mrel$ ($|$ - $\rangle\rangle$ - [*50*,*90*] *95*)
where
$|R\rangle\rangle p \equiv \{\ (a,\{a\})\ |\ a\ .\ \exists B\ .\ (a,B) \in R \wedge (\exists b{\in}B\ .\ (b,\{b\}) \in p)\ \}$

```
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 (clarsimp 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 ¬(∃ B . (a,B) ∈ R ∩ −(U ∗ p))
    proof
      assume ∃ B . (a,B) ∈ R ∩ −(U ∗ p)
      from this obtain B where (a,B) ∈ R ∧ (a,B) ∉ U ∗ p
        by auto
      thus False
        using 1 by (metis (no-types, lifting) assms top-sp-test)
    qed
    thus x ∈ aDom (R ∩ −(U ∗ p))
      using 1 aDom by blast
  qed
next
  show aDom (R ∩ −(U ∗ p)) ⊆ |R]p
  proof
    fix x
    assume x ∈ aDom (R ∩ −(U ∗ p))
    from this obtain a where 2: x = (a,{a}) ∧ ¬(∃ B . (a,B) ∈ R ∩ −(U ∗ p))
      by (smt aDom surj-pair mem-Collect-eq)
    hence ∀ B . (a,B) ∈ R ⟶ (∀ b∈B . (b,{b}) ∈ p)
      using assms by (metis (no-types, lifting) IntI oc-top-sp-test)
    thus x ∈ |R]p
      using 2 abox-def by blast
  qed
qed

lemma abox:
  assumes test p
  shows |R]p = aDom (ne (R↓) ∗ ≀ p)
  by (simp add: abox-1 assms d-ne-down-dp-complement-test)

lemma edia-1:
  assumes test p
  shows |R⟩⟩p = Dom (R ∩ −(U ∗ ≀ p))
proof
  show |R⟩⟩p ⊆ Dom (R ∩ −(U ∗ ≀ p))
  proof
    fix x
    assume x ∈ |R⟩⟩p
    from this obtain a b B where 1: x = (a,{a}) ∧ (a,B) ∈ R ∧ b ∈ B ∧
(b,{b}) ∈ p
      by (smt edia-def surj-pair mem-Collect-eq)
    hence (a,B) ∈ −(U ∗ ≀ p)
```

by (*metis* (*no-types*, *lifting*) *lattice-class.inf-sup-ord*(*2*) *oc-top-sp-test test-complement*)
thus $x \in Dom\ (R \cap -(U * \wr\ p))$
using *1 Dom-def* **by** *auto*
qed
next
show $Dom\ (R \cap -(U * \wr\ p)) \subseteq |R\rangle\rangle p$
proof
fix x
assume $x \in Dom\ (R \cap -(U * \wr\ p))$
from *this* **obtain** $a\ B$ **where** *2*: $x = (a,\{a\}) \wedge (a,B) \in R \wedge (a,B) \in -(U * \wr\ p)$
by (*smt Dom-def surj-pair mem-Collect-eq IntE*)
hence $\exists\, b{\in}B\ .\ (b,\{b\}) \in p$
by (*meson oc-top-sp-test test-complement test-complement-closed*)
thus $x \in |R\rangle\rangle p$
using *2 edia-def* **by** *blast*
qed
qed

lemma *edia*:
assumes *test p*
shows $|R\rangle\rangle p = Dom\ (ne\ (R{\downarrow}) * p)$
by (*metis assms d-ne-down-dp-complement-test edia-1 test-double-complement*)

lemma *ebox*:
assumes *test p*
shows $|R]]p = aDom\ (R * \wr\ p)$
proof
show $|R]]p \subseteq aDom\ (R * \wr\ p)$
proof
fix x
assume $x \in |R]]p$
from *this* **obtain** a **where** *1*: $x = (a,\{a\}) \wedge (\forall B\ .\ (a,B) \in R \longrightarrow (\exists\, b{\in}B\ .\ (b,\{b\}) \in p))$
by (*smt ebox-def surj-pair mem-Collect-eq*)
hence $\neg(\exists B\ .\ (a,B) \in R * \wr\ p)$
by (*metis* (*no-types*, *lifting*) *s-prod-test test-complement*)
thus $x \in aDom\ (R * \wr\ p)$
using *1 aDom* **by** *blast*
qed
next
show $aDom\ (R * \wr\ p) \subseteq |R]]p$
proof
fix x
assume $x \in aDom\ (R * \wr\ p)$
from *this* **obtain** a **where** *2*: $x = (a,\{a\}) \wedge \neg(\exists B\ .\ (a,B) \in R * \wr\ p)$
by (*smt aDom surj-pair mem-Collect-eq*)
have $\forall B\ .\ (a,B) \in R \longrightarrow (\exists\, b{\in}B\ .\ (b,\{b\}) \in p)$

```
    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 ∈ |R]]p
      using 2 ebox-def by blast
  qed
qed

lemma abox-2:
  assumes test p
  shows |R]p = −((R ∩ −(U ∗ p)) ∗ U) ∩ 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↓) ∗ ≀ p ∗ U) ∩ 1
  by (simp add: abox assms ne-a-expl ne-sp-test)

lemma abox-4:
  assumes test p
  shows |R]p = ((U ∗ p) ⊘ R) ∩ 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↓)]]p
  by (simp add: abox assms ebox)

lemma abox-edia:
  assumes test p
  shows |R]p = ≀ |R⟩⟩(≀ p)
  by (simp add: abox assms edia)

lemma abox-adia:
  assumes test p
  shows |R]p = ≀ |ne (R↓)⟩(≀ p)
  by (simp add: abox adia assms)

lemma edia-adia:
  assumes test p
  shows |R⟩⟩p = |ne (R↓)⟩⟩p
  by (simp add: adia assms edia)

lemma edia-abox:
```

assumes *test p*
shows $|R\rangle\rangle p = \wr\ |R](\wr\ p)$
by (*metis abox-1 assms d-complement-ad edia-1 semilattice-inf-class.inf.cobounded2*)

lemma *edia-ebox*:
assumes *test p*
shows $|R\rangle\rangle p = \wr\ |ne\ (R\downarrow)]](\wr\ p)$
by (*simp add*: *abox assms ebox edia-abox*)

lemma *abox-ne-down*:
assumes *test p*
shows $|R]p = |ne\ (R\downarrow)]p$
by (*simp add*: *abox assms ne-down-idempotent*)

lemma *edia-ne-down*:
assumes *test p*
shows $|R\rangle\rangle p = |ne\ (R\downarrow)\rangle\rangle p$
by (*simp add*: *assms edia ne-down-idempotent*)

lemma *adia-up*:
assumes *test p*
shows $|R\rangle p = |R\uparrow\rangle p$
proof −
have $|R\uparrow\rangle p = Dom\ (R\uparrow \cap U * p)$
by (*metis adia assms iu-assoc iu-unit-up up-dist-iu-oi*)
also have ... $= Dom\ (R \cap U * p)$
by (*metis assms d-def-expl domain-up-down-conjugate sp-test-dist-oi-right top-sp-test-down-closed*)
also have ... $= |R\rangle p$
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 $|R]]p = |R\uparrow]]p$
by (*metis Int-commute adia adia-up assms ebox semilattice-inf-class.inf-le1*)

lemma *adia-ebox*:
assumes *test p*
shows $|R\rangle p = \wr\ |R]](\wr\ p)$
by (*metis* (*no-types*, *lifting*) *adia assms d-complement-ad ebox test-double-complement*)

lemma *ebox-adia*:
assumes *test p*
shows $|R]]p = \wr\ |R\rangle(\wr\ p)$

by (*simp add*: *adia assms ebox*)

lemma *abox-down*:
assumes *test p*
shows $|R]p = |R{\downarrow}]p$
by (*simp add*: *abox assms*)

lemma *edia-down*:
assumes *test p*
shows $|R\rangle\rangle p = |R{\downarrow}\rangle\rangle p$
by (*simp add*: *assms edia*)

lemma *fusion-oi-complement-top-test-up*:
$test\ p \Longrightarrow fus\ R \cap -(U * p) \subseteq (R \cap -(U * p))\uparrow$
apply (*unfold top-test*)
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *adia-left-isotone*:
$test\ p \Longrightarrow R \subseteq S \Longrightarrow |R\rangle p \subseteq |S\rangle p$
by (*metis adia d-isotone inf.absorb-iff1 sp-test-dist-oi*)

lemma *adia-right-isotone*:
$test\ p \Longrightarrow test\ q \Longrightarrow p \subseteq q \Longrightarrow |R\rangle p \subseteq |R\rangle q$
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*:
$test\ p \Longrightarrow R \subseteq S \Longrightarrow |S]p \subseteq |R]p$
apply (*clarsimp simp*: *mr-simp*) **by** *force*

lemma *abox-right-isotone*:
$test\ p \Longrightarrow test\ q \Longrightarrow p \subseteq q \Longrightarrow |R]p \subseteq |R]q$
by (*smt* (*verit*, *ccfv-threshold*) *IntE abox-def inf.orderE mem-Collect-eq subsetI*)

lemma *edia-left-isotone*:
$test\ p \Longrightarrow R \subseteq S \Longrightarrow |R\rangle\rangle p \subseteq |S\rangle\rangle p$
by (*metis Int-mono adia-left-isotone down-isotone edia-adia order-refl*)

lemma *edia-right-isotone*:
$test\ p \Longrightarrow test\ q \Longrightarrow p \subseteq q \Longrightarrow |R\rangle\rangle p \subseteq |R\rangle\rangle q$
by (*simp add*: *adia-right-isotone edia-adia*)

lemma *ebox-left-antitone*:
$test\ p \Longrightarrow R \subseteq S \Longrightarrow |S]]p \subseteq |R]]p$
by (*metis* (*no-types*, *lifting*) *adia-ebox adia-left-isotone ebox-adia test-complement-antitone test-double-complement*)

lemma *ebox-right-isotone*:

test p $\Longrightarrow$ *test q* $\Longrightarrow$ $p \subseteq q$ $\Longrightarrow$ $|R]]p \subseteq |R]]q$
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 $|fus\ R\rangle\rangle p = Dom\ (fus\ R \cap -(U * \wr\ p))$
using *assms edia-1* **by** *blast*
also have ... $\subseteq Dom\ (R \cap -(U * \wr\ p))$
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*:
$fis\ R \subseteq S \longleftrightarrow (\forall\ a\ b\ B\ .\ (a,B) \in R \land b \in B \longrightarrow (a,\{b\}) \in S)$
apply *standard*
apply (*simp add*: *basic-trans-rules*(*31*) *fission-set*)
apply (*clarsimp simp*: *mr-simp*)
by *blast*

lemma *below-fission-up*:
$S \subseteq (fis\ R)\uparrow \longleftrightarrow (\forall\ a\ B\ .\ (a,B) \in S \longrightarrow (\exists\ C\ .\ (a,C) \in R \land C \cap B \neq \{\}))$
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 (*clarsimp 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 $S \subseteq (fis\ R)\uparrow$
  proof
    fix x
    assume $x \in S$
    from *this* **obtain** $a\ B$ **where** *2*: $x = (a,B) \wedge (a,B) \in S$
      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** *3*: $(a,C) \in R \wedge b \in C \wedge b \in B$
      by *auto*
    hence $(a,\{b\}) \in fis\ R$
      using *fission-set* **by** *blast*
    thus $x \in (fis\ R)\uparrow$
      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 $S \subseteq (fis\ R)\uparrow$
  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 $S \subseteq (fis\ R)\uparrow$
  shows $|R]p = |S]]p$
  by (*simp add*: *abox-below-ebox assms ebox-below-abox subset-antisym*)

lemma *abox-eq-ebox-sufficient*:
  $S = fis\ R \vee S = ne\ (R\downarrow) \vee S = (ne\ (R\downarrow))\uparrow \longrightarrow fis\ R \subseteq S \wedge S \subseteq (fis\ R)\uparrow$
  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 $\Longrightarrow$ $|R]p = |fis\ R]]p$
by (*metis abox abox-fission ebox fission-down-ne-fixpoint*)

lemma *ebox-down-ne-up-abox*:
test p $\Longrightarrow$ $|R]p = |(ne\ (R\downarrow))\uparrow]]p$
using *abox-ebox ebox-up* **by** *blast*

lemma *same-fusion*:
assumes *fis R* $\sqsubseteq\downarrow$ *S*
and *S* $\sqsubseteq\downarrow$ *fus R*
shows *fis R* = *fis S*
by (*metis assms fission-down fission-fusion fission-fusion-galois subset-antisym*)

lemma *same-abox*:
assumes *fis R* $\sqsubseteq\downarrow$ *S*
and *S* $\sqsubseteq\downarrow$ *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 $|R\rangle p = |R\rangle\rangle p$
by (*metis assms edia-adia fission-down-ne-fixpoint fission-inner-deterministic-fixpoint*)

lemma *abox-adia-deterministic*:
assumes *test p*
and *deterministic R*
shows $|R]p = |R\rangle p$
proof
show $|R]p \subseteq |R\rangle p$
proof
fix x

assume $x \in |R]p$
from *this* **obtain** a **where** *1*: $x = (a,\{a\}) \wedge (\forall B . (a,B) \in R \longrightarrow (\forall b{\in}B . (b,\{b\}) \in p))$
using *abox-def* **by** *force*
from *assms*(*2*) **obtain** B **where** $(a,B) \in R$
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** *2*: $x = (a,\{a\}) \wedge (a,B) \in R \wedge (\forall b{\in}B . (b,\{b\}) \in p)$
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 (*clarsimp simp*: *mr-simp*) **by** *force*

lemma *ou-unit-abox*:
$test\ p \Longrightarrow |\{\}]p = 1$
by (*metis abox-1 abox-sp-unit disjoint-eq-subset-Compl empty-subsetI inf.absorb-iff2 test-complement-closed*)

lemma *ou-unit-test-implication*:
$test\ p \Longrightarrow \{\} \rightarrow p = 1$
by *blast*

lemma *sp-unit-abox*:
$test\ p \Longrightarrow |1]p = p$
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*:
$test\ p \Longrightarrow 1 \rightarrow p = p$
by *simp*

lemma *test-abox-ebox*:
$test\ p \Longrightarrow test\ q \Longrightarrow |q]p = |q]]p$
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*:
$test\ p \Longrightarrow test\ q \Longrightarrow |q]p = q \rightarrow p$
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 $|R]p \cup |R\rangle 1 = 1$
apply (*rule antisym*)
apply (*simp add*: *assms abox adia-ebox*)

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

lemma *d-test-sp*:
test p $\Longrightarrow$ *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 $\Longrightarrow$ *aDom* $(p * R) = \wr\ p \cup$ *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 $\Longrightarrow$ *test q* $\Longrightarrow$ $|p * R\rangle q = p * |R\rangle q$
by (*metis* (*no-types*, *lifting*) *adia d-test-sp test-assoc3 test-double-complement*)

lemma *ebox-test-sp*:
test p $\Longrightarrow$ *test q* $\Longrightarrow$ $|p * R]]q = \wr\ p \cup |R]]q$
by (*simp add*: *ad-test-sp ebox test-assoc3*)

lemma *abox-test-sp*:
assumes *test p*
and *test q*
shows $|p * R]q = \wr\ p \cup |R]q$
proof $-$
have $|p * R]q =$ *aDom* $((p * R) \cap -(U * q))$
by (*simp add*: *abox-1 assms*(*2*))
also have ... = *aDom* $(p * (R \cap -(U * q)))$
by (*metis Int-assoc assms*(*1*) *test-sp*)
also have ... $= \wr\ p \cup |R]q$
by (*simp add*: *abox-1 ad-test-sp assms*)
finally show *?thesis*
.
qed

lemma *abox-test-sp-2*:
test p $\Longrightarrow$ *test q* $\Longrightarrow$ $p \cup |R]q = |\wr\ p * R]q$
by (*simp add*: *abox-test-sp test-double-complement*)

lemma *abox-test-sp-3*:
test p $\Longrightarrow$ *test q* $\Longrightarrow$ $p \to |R]q = |p * R]q$
by (*simp add*: *abox-test-sp*)

lemma *fission-sp-dist*:
fis $(R * S) =$ *fis* $(R *$ *Dom S*$)$ $*$ *fis S*
proof $-$
have $S =$ *Dom S* $* (S \cup$ *aDom S* $* 1_{\cup\cup})$
by (*auto simp*: *mr-simp*)
hence *fis* $(R * S) =$ *fis* $(R *$ *Dom S* $* (S \cup$ *aDom S* $* 1_{\cup\cup}))$
by (*metis d-s-id-ax sp-test-sp-oi-right test-sp*)

also have ... = *fis* (*R* ∗ *Dom S*) ∗ *fis* (*S* ∪ *aDom S* ∗ $1_{\cup\cup}$)
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* ∗ $1_{\cup\cup}$))
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*↓) ∗ (≀ *p* ∪ ≀ *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*↓) ∗ ≀ *p*) ∗ *aDom* (*ne* (*R*↓) ∗ ≀ *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* ∪ ≀ *q*) ⊆ |*R*⟩*p* ∪ |*ne* (*R*↓)⟩(≀ *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 $|R\rangle(p \to q) * |R]p \subseteq |R\rangle q$
  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 $|R\rangle p * |R](p \to q) \subseteq |R\rangle q$
proof –
  have $p \subseteq p \to q \to q$
    using *assms*(*1*) **by** *blast*
  hence $|R\rangle p \subseteq |R\rangle((p \to q) \to q)$
    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 $|R](p \to q) \subseteq |R\rangle p \to |R\rangle q$
  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 $|R]p \subseteq |R\rangle q \to |R\rangle(p * q)$
proof –
  have $p \subseteq q \to p * q$
    by (*metis assms subset-refl test-galois-1 test-sp-commute*)
  hence $|R]p \subseteq |R](q \to p * q)$
    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 $|R]p * |R](p \to q) \subseteq |R]q$
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 $|R](p \to q) \subseteq |R]p \to |R]q$
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 $|R \cup S]]p = |R]]p * |S]]p$
by (*auto simp*: *mr-simp*)

lemma *abox-left-dist-ou*:
assumes *test p*
shows $|R \cup S]p = |R]p * |S]p$
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 $|R \cup S\rangle p = |R\rangle p \cup |S\rangle p$
by (*auto simp*: *mr-simp*)

lemma *edia-left-dist-ou*:
assumes *test p*
shows $|R \cup S\rangle\rangle p = |R\rangle\rangle p \cup |S\rangle\rangle p$
by (*simp add*: *assms boolean-algebra.conj-disj-distrib2 d-dist-ou edia-1*)

lemma *abox-dist-iu-1*:
assumes *test p*
shows $|R \cup\cup S]p = |Dom\ R * ne\ (S\downarrow)]]p * |Dom\ S * ne\ (R\downarrow)]]p$
proof
have *1*: $|R \cup\cup S]p \subseteq |Dom\ R * ne\ (S\downarrow)]]p$
by (*metis abox-ebox assms d-sp-ne-down-below-ne-iu-down ebox-left-antitone*)
have $|R \cup\cup S]p \subseteq |Dom\ S * ne\ (R\downarrow)]]p$
by (*metis abox-ebox assms d-sp-ne-down-below-ne-iu-down ebox-left-antitone iu-commute*)
thus $|R \cup\cup S]p \subseteq |Dom\ R * ne\ (S\downarrow)]]p * |Dom\ S * ne\ (R\downarrow)]]p$
using *1* **by** (*simp add*: *assms ebox*)
next
have $|Dom\ R * ne\ (S\downarrow)]]p * |Dom\ S * ne\ (R\downarrow)]]p \subseteq |Dom\ R * Dom\ S * ne\ (S\downarrow)]]p * |Dom\ S * ne\ (R\downarrow)]]p$
apply (*clarsimp simp*: *mr-simp*)
by (*metis UN-I singletonI*)
also have ... $\subseteq |Dom\ R * Dom\ S * ne\ (S\downarrow)]]p * |Dom\ R * Dom\ S * ne\ (R\downarrow)]]p$

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*)↓)) ∗ ≀ *p*)
using *assms ebox* **by** *blast*
also have ... ⊆ *aDom* ((*ne* ((*Dom R* ∗ *Dom S* ∗ *S* ∪∪ *Dom R* ∗ *Dom S* ∗ *R*)↓)) ∗ ≀ *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*:
$|R * S]p = \{ (a,\{a\}) \mid a . \forall B . (a,B) \in R \longrightarrow (\forall C . (\exists f . (\forall b \in B . (b,f\ b) \in S) \wedge C = \bigcup \{ f\ b \mid b . b \in B \}) \longrightarrow (\forall c \in C . (c,\{c\}) \in p)) \}$
apply (*unfold abox-def s-prod-def*)
by *blast*

lemma *abox-sp-1*:
assumes *test p*
shows $|R]|S\rangle 1 * |R * S]p \subseteq |R]|S]p$
proof −
have $|R]|S\rangle 1 * |R * S]p = |R]|S\rangle 1 \cap |R * S]p$
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 ... $\subseteq |R]|S]p$
proof
fix *x*
assume $x \in |R]|S\rangle 1 \cap |R * S]p$
from *this* **obtain** *a* **where** *1*: $x = (a,\{a\}) \wedge x \in |R]|S\rangle 1 \wedge x \in |R * S]p$
by (*metis Int-iff abox-test adia-test order-refl subid-aux2 subsetD surj-pair*)
hence *2*: $\forall B . (a,B) \in R \longrightarrow (\forall b \in B . \exists D . (b,D) \in S)$
by (*smt abox-adia-sp-one-set mem-Collect-eq old.prod.inject*)
have *3*: $\forall B . (a,B) \in R \longrightarrow (\forall C . (\exists f . (\forall b \in B . (b,f\ b) \in S) \wedge C = \bigcup \{ f\ b \mid b . b \in B \}) \longrightarrow (\forall c \in C . (c,\{c\}) \in p))$
using *1* **by** (*smt sp-abox-set mem-Collect-eq old.prod.inject*)
have $\forall B . (a,B) \in R \longrightarrow (\forall C . (\exists b \in B . (b,C) \in S) \longrightarrow (\forall c \in C . (c,\{c\}) \in p))$
proof (*rule allI*, *rule impI*)
fix *B*
assume *4*: $(a,B) \in R$
hence $\exists DD . \forall b \in B . (b,DD\ b) \in S$
using *2* **by** (*auto intro*: *bchoice*)
from *this* **obtain** *DD* **where** *5*: $\forall b \in B . (b,DD\ b) \in S$
by *auto*
show $\forall C . (\exists b \in B . (b,C) \in S) \longrightarrow (\forall c \in C . (c,\{c\}) \in p)$
proof (*rule allI*, *rule impI*)
fix *C*
assume $\exists b \in B . (b,C) \in S$
from *this* **obtain** *b* **where** *6*: $b \in B \wedge (b,C) \in S$
by *auto*
let $?f = \lambda x . \textit{if } x = b \textit{ then } C \textit{ else } DD\ x$
let $?C = C \cup \bigcup \{ ?f\ x \mid x . x \in B \wedge x \neq b \}$
have $\exists f . (\forall b \in B . (b,f\ b) \in S) \wedge ?C = \bigcup \{ f\ b \mid b . b \in B \}$
apply (*rule exI*[**where** *?x=?f*])
using *5 6* **by** *auto*
hence $\forall c \in ?C . (c,\{c\}) \in p$
using *3 4* **by** *auto*
thus $\forall c \in C . (c,\{c\}) \in p$
by *blast*

qed
qed
thus $x \in |R]|S]p$
using *1 abox-abox-set* **by** *blast*
qed
finally show *?thesis*
.
qed

lemma *abox-sp-2*:
assumes *test p*
shows $|R]|S]p = |R\downarrow * S]p$
proof −
have $|R]|S]p = aDom\ (ne\ (R\downarrow) * Dom\ (ne\ (S\downarrow) * \wr\ p))$
by (*metis abox abox-test assms d-complement-ad*)
also have ... $= aDom\ (ne\ (R\downarrow) * ne\ (S\downarrow) * \wr\ p)$
by (*simp add*: *test-assoc3*)
also have ... $= aDom\ (ne\ ((R\downarrow * S)\downarrow) * \wr\ p)$
by (*simp add*: *down-dist-sp ne-dist-down-sp*)
also have ... $= |R\downarrow * S]p$
by (*simp add*: *abox assms*)
finally show *?thesis*
.
qed

lemma *abox-sp-3*:
assumes *test p*
shows $|R]|S]p \subseteq |R * S]p$
by (*clarsimp simp*: *mr-simp*) *auto*

lemma *abox-sp-4*:
assumes *test p*
shows $|R * S]p \subseteq |R]|S\rangle 1 \to |R]|S]p$
proof −
have $|R]|S\rangle 1 * |R * S]p \subseteq |R]|S]p$
by (*auto simp*: *assms abox-sp-1*)
hence $|R]|S\rangle 1 \cap |R * S]p \subseteq |R]|S]p$
by (*smt* (*verit*) *abox-test adia-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 $|R]|S\rangle 1 * |R * S]p = |R]|S\rangle 1 * |R]|S]p$
proof (*rule antisym*)
have $|R * S]p \subseteq |R]|S\rangle 1 \to |R]|S]p$
by (*simp add*: *abox-sp-4 assms*)

hence $|R]|S\rangle 1 \cap |R * S]p \subseteq |R]|S]p$
by *blast*
hence $|R]|S\rangle 1 \cap |R * S]p \subseteq |R]|S\rangle 1 \cap |R]|S]p$
by *blast*
thus $|R]|S\rangle 1 * |R * S]p \subseteq |R]|S\rangle 1 * |R]|S]p$
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 $|R]|S\rangle 1 * |R]|S]p \subseteq |R]|S\rangle 1 * |R * S]p$
by (*simp add*: *abox-sp-3 assms s-prod-isor*)
qed

lemma *abox-sp-6*:
assumes *test p*
shows $|R]|S\rangle 1 \rightarrow |R * S]p = |R]|S\rangle 1 \rightarrow |R]|S]p$
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 $|R * S]p = |R]|S]p$
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 $\ldots = |(Q * R) * S\rangle p$
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 $|Q * (R * S)]]p = |(Q * R) * S]]p$
by (*simp add*: *adia-sp-associative assms ebox-adia*)

lemma *edia-sp-associative*:
assumes *test p*
shows $|Q * (R * S)\rangle\rangle p = |(Q * R) * S\rangle\rangle p$
proof −
have $|fis\ (Q * (R * S))\rangle\rangle p = |fis\ (Q * Dom\ (R * S)) * (fis\ (R * Dom\ S) * fis\ S)\rangle\rangle p$

by (*metis fission-sp-dist*)
also have ... = $|(fis\ (Q * Dom\ (R * S)) * fis\ (R * Dom\ S)) * fis\ S\rangle\rangle p$
by (*simp add*: *inner-deterministic-sp-assoc semilattice-inf-class.inf-commute semilattice-inf-class.le-infI1 fission-var*)
also have ... = $|fis\ (Q * Dom\ (R * Dom\ S)) * fis\ (R * Dom\ S) * fis\ S\rangle\rangle p$
by *simp*
also have ... = $|fis\ (Q * (R * Dom\ S)) * fis\ S\rangle\rangle p$
by (*metis fission-sp-dist*)
also have ... = $|fis\ ((Q * R) * Dom\ S) * fis\ S\rangle\rangle p$
by (*metis d-complement-ad test-assoc3*)
also have ... = $|fis\ ((Q * R) * S))\rangle\rangle p$
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 $X \neq \{\}$
shows $|R]\bigcap X = (\bigcap p{\in}X\ .\ |R]p)$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply (*clarsimp simp*: *mr-simp*)
using *assms* **by** *blast*

lemma *ebox-left-dist-oU*:
assumes $X \neq \{\}$
shows $|\bigcup X]]p = (\bigcap R{\in}X\ .\ |R]]p)$
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 $X \neq \{\}$
shows $|\bigcup X]p = (\bigcap R{\in}X\ .\ |R]p)$
apply (*rule antisym*)
apply (*clarsimp simp*: *mr-simp*)
apply *blast*
apply (*clarsimp simp*: *mr-simp*)
using *assms* **by** *blast*

lemma *adia-left-dist-oU*:
$|\bigcup X\rangle p = (\bigcup R{\in}X\ .\ |R\rangle p)$

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

lemma *edia-left-dist-oU*:
$|\bigcup X\rangle\rangle p = (\bigcup R{\in}X\ .\ |R\rangle\rangle p)$
apply (*clarsimp simp*: *mr-simp*)
by *blast*

8.6 Goldblatt's axioms with star

no-notation *rtrancl* ((-*) [*1000*] *999*)
notation *star* (-* [*1000*] *999*)

lemma *star-induct-1*:
assumes $1 \subseteq X$
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 $S \subseteq 1 \cup 1_{\cup\cup}$
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 $1 \subseteq X \oslash S$
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*:
$R^*\downarrow = (R\downarrow)^* \cup 1_{\cup\cup}$
proof
have $R^* * (1 \cup 1_{\cup\cup}) \subseteq (R\downarrow)^* \cup 1_{\cup\cup}$
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 ... $\subseteq (ne\ (R\downarrow))^*$
by (*metis* (*no-types*, *lifting*) *IntE UnCI inf.absorb-iff2 sp-oi-subdist star-unfold-eq subsetI*)
finally show $R\downarrow * ((ne\ (R\downarrow))^* \cup 1_{\cup\cup}) \subseteq ((ne\ (R\downarrow))^* \cup 1_{\cup\cup})$
by *blast*
qed
thus $ne\ ((R\downarrow)^*) \subseteq (ne\ (R\downarrow))^*$
by (*smt Compl-disjoint2 Int-commute Int-left-commute ne-dist-ou semilattice-inf-class.le-iff-inf sup-bot.right-neutral*)
next
show $(ne\ (R\downarrow))^* \subseteq ne\ ((R\downarrow)^*)$
proof (*rule star-induct-1*)
show $1 \subseteq ne\ ((R\downarrow)^*)$
using *star-refl test-ne* **by** *auto*
next
show $ne\ (R\downarrow) * ne\ ((R\downarrow)^*) \subseteq ne\ ((R\downarrow)^*)$
by (*metis IntE IntI UnCI ne-dist-down-sp star-unfold-eq subsetI*)
qed
qed

lemma *abox-star-unfold*:
$test\ p \Longrightarrow |R^*]p = p * |R]|R^*]p$
by (*metis abox-left-dist-ou abox-sp-7 sp-unit-abox star-total star-unfold-eq*)

lemma *star-sp-test-commute*:
assumes $S \subseteq 1 \cup 1_{\cup\cup}$
and $Q * S \subseteq S * R$
shows $Q^* * S \subseteq S * R^*$
proof (*rule star-induct*)
show $S \subseteq 1 \cup 1_{\cup\cup}$
by (*simp add*: *assms*(*1*))
next
show $S \subseteq S * R^*$
by (*metis s-prod-idr s-prod-isor star-refl*)
next
have $Q * (S * R^*) \subseteq S * R * R^*$
by (*metis* (*no-types*, *lifting*) *assms s-prod-distr subset-Un-eq test-iu-test-sp-assoc-3*)
thus $Q * (S * R^*) \subseteq S * R^*$
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 $|R\rangle p \subseteq p \longleftrightarrow |R^*\rangle p \subseteq p$
proof
  assume $|R\rangle p \subseteq p$
  hence $\wr\ p * Dom\ (R * p) = \{\}$
    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 $\wr\ p * (R^* * p) = \{\}$
    by (*meson assms subset-empty test-sp-shunting*)
  hence $\wr\ p * Dom\ (R^* * p) = \{\}$
    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 $|R^*\rangle p \subseteq q$
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 $|R^*](|R\rangle p \to p) \subseteq |R^*\rangle p \to p$
proof −
let $?q = |R^*](|R\rangle p \to p)$
have $|R^*\rangle p \subseteq ?q \to p$
proof (*rule adia-star-induct-1*)
show *test p*
by (*simp add*: *assms*)
next
show *test* $(?q \to p)$
by (*simp add*: *assms*)
next
have $|R\rangle(?q \to p) * |R]?q * (|R\rangle p \to p) \subseteq |R\rangle p * (|R\rangle p \to p)$
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 $|R\rangle(?q \to p) \subseteq (|R\rangle p \to p) * |R]?q \to p$
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 ... $= ?q \to p$
by (*metis abox-star-unfold assms test-implication-closed*)
finally show $p \cup |R\rangle(?q \to p) \subseteq ?q \to p$
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 $(s\text{-}id \cup p\text{-}id) * R = R \cup p\text{-}id$
by (*simp add*: *s-prod-distr*)

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*.a_1

lemma *counter-02*:
∃ *R*::(*Enum.finite-1*,*Enum.finite-1*) *mrel* . ∃ *p* . ¬ (*test p* ⟶ (*R* ∩ −(*U* ∗ *p*)) ∗ *U* = *R* ∗ −(*p* ∗ *U*))
apply (*rule exI*[**where** *?x*={(*a-1*,{})}])
apply (*rule exI*[**where** *?x*={}])
apply (*clarsimp 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*:
∃ *R*::(*Enum.finite-1*,*Enum.finite-1*) *mrel* . ∃ *p* . ¬ (*test p* ⟶ (*R* ∩ −(*U* ∗ *p*)) ∗ $1_{\cup\cup}$ = *R* ∗ (−(*p* ∗ *U*) ∩ $1_{\cup\cup}$))
apply (*rule exI*[**where** *?x*={(*a-1*,{})}])
apply (*rule exI*[**where** *?x*={}])
apply (*clarsimp 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*.a_1
abbreviation *b-2* ≡ *finite-2*.a_2
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*:
∃ *R*::(*Enum.finite-2*,*Enum.finite-2*) *mrel* . ∃ *p q* . ¬ (*test p* ⟶ *test q* ⟶ |*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 (*clarsimp simp*: *s-id-def*)
apply (*subst top-test*)
apply (*clarsimp simp*: *s-id-def*)
apply (*unfold abox-def*)
apply (*clarsimp simp*: *s-id-def*)
by *blast*

lemma *counter-05*:

¬ (∃f . ∀R p . test p ⟶ |R⟩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*:
¬ (∃f . ∀R p . test p ⟶ |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*:
¬ (∃f . mono f ∧ (∀R . fus R = lfp (λX . f R X)))
proof
assume ∃f::('a,'b) mrel ⇒ ('a,'b) mrel ⇒ ('a,'b) mrel . mono f ∧ (∀R . fus R = lfp (λX . f R X))
from *this* **obtain** f :: ('a,'b) mrel ⇒ ('a,'b) mrel ⇒ ('a,'b) mrel **where** mono f ∧ (∀R . fus R = lfp (λX . f R X))
by *auto*
hence fus {} ⊆ 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 ≡ finite-3.a_1
abbreviation c-2 ≡ finite-3.a_2
abbreviation c-3 ≡ finite-3.a_3

lemma *counter-08*:
¬ (∼(1::(Enum.finite-3,Enum.finite-3) mrel) ∗ ∼1 ∈ {1, ∼1})
proof −
let ?c = (c-1,{c-1,c-2,c-3})
have 1: ?c ∈ ∼1 ∗ ∼1
apply (*clarsimp simp*: *mr-simp*)
apply (*rule exI*[**where** ?x={c-2,c-3}])
using *UNIV-finite-3* **by** *auto*
have ?c ∉ 1 ∧ ?c ∉ ∼1
by (*auto simp*: *mr-simp*)
thus *?thesis*
using *1* **by** *auto*
qed

lemma *counter-09*:
¬ (∼(1::(Enum.finite-3,Enum.finite-3) mrel) ⊙ 1 ∈ {1, ∼1})
by (*metis counter-08 co-prod empty-iff ic-involutive insert-iff*)

lemma *ex-2-cases*:
∃b. b = b-1 ∨ 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-1* $\longrightarrow$ *X* = *Y*) $\wedge$ (*b* = *b-2* $\longrightarrow$ *X* = *Z*)} = *Y* $\cup$ *Z*
by *auto*

lemma *ex-2-set-cases*:
($\exists$ *B*::*Enum.finite-2 set* . *P B*) $\longleftrightarrow$ *P* {} $\vee$ *P* {*b-1*} $\vee$ *P* {*b-2*} $\vee$ *P* {*b-1*,*b-2*}
proof −
let *?U* = *UNIV*::*Enum.finite-2 set set*
have *?U* $\subseteq$ {{},{*b-1*},{*b-2*},{*b-1*,*b-2*}}
proof
fix *x*
have *x* $\subseteq$ {*b-1*,*b-2*}
by *auto*
thus *x* $\in$ {{},{*b-1*},{*b-2*},{*b-1*,*b-2*}}
by *auto*
qed
hence *?U* = {{},{*b-1*},{*b-2*},{*b-1*,*b-2*}}
by *auto*
thus *?thesis*
by (*metis UNIV-I empty-iff insertE*)
qed

abbreviation *B-0* $\equiv$ {}::*Enum.finite-2 set*
abbreviation *B-1* $\equiv$ {*b-1*}
abbreviation *B-2* $\equiv$ {*b-2*}
abbreviation *B-3* $\equiv$ {*b-1*,*b-2*}
abbreviation *mkf x y* $\equiv$ λ*z* . *if z* = *b-1 then x else y*

lemma *mkf*:
f = *mkf* (*f b-1*) (*f b-2*)
by *auto*

lemma *mkf2*:
f b-1 = *X* $\wedge$ *f b-2* = *Y* $\Longrightarrow$ *f* = *mkf X Y*
by *auto*

lemma *ex-2-mrel-cases*:
($\exists$ *f*::*Enum.finite-2* $\Rightarrow$ *Enum.finite-2 set* . *P f*) $\longleftrightarrow$
P (*mkf B-0 B-0*) $\vee$ *P* (*mkf B-0 B-1*) $\vee$ *P* (*mkf B-0 B-2*) $\vee$ *P* (*mkf B-0 B-3*) $\vee$
P (*mkf B-1 B-0*) $\vee$ *P* (*mkf B-1 B-1*) $\vee$ *P* (*mkf B-1 B-2*) $\vee$ *P* (*mkf B-1 B-3*) $\vee$
P (*mkf B-2 B-0*) $\vee$ *P* (*mkf B-2 B-1*) $\vee$ *P* (*mkf B-2 B-2*) $\vee$ *P* (*mkf B-2 B-3*) $\vee$
P (*mkf B-3 B-0*) $\vee$ *P* (*mkf B-3 B-1*) $\vee$ *P* (*mkf B-3 B-2*) $\vee$ *P* (*mkf B-3 B-3*)
proof
assume $\exists$ *f*::*Enum.finite-2* $\Rightarrow$ *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*: ⋀*x* . *f x* = *B-0* ∨ *f x* = *B-1* ∨ *f x* = *B-2* ∨ *f x* = *B-3*
  by *auto*
have *f* = *mkf B-0 B-0* ∨ *f* = *mkf B-0 B-1* ∨ *f* = *mkf B-0 B-2* ∨ *f* = *mkf B-0 B-3* ∨
  f = *mkf B-1 B-0* ∨ *f* = *mkf B-1 B-1* ∨ *f* = *mkf B-1 B-2* ∨ *f* = *mkf B-1 B-3* ∨
  f = *mkf B-2 B-0* ∨ *f* = *mkf B-2 B-1* ∨ *f* = *mkf B-2 B-2* ∨ *f* = *mkf B-2 B-3* ∨
  f = *mkf B-3 B-0* ∨ *f* = *mkf B-3 B-1* ∨ *f* = *mkf B-3 B-2* ∨ *f* = *mkf B-3 B-3*
  using *2*[*of b-1*] *2*[*of b-2*] *mkf2*[*of f*] **by** *blast*
thus *P* (*mkf B-0 B-0*) ∨ *P* (*mkf B-0 B-1*) ∨ *P* (*mkf B-0 B-2*) ∨ *P* (*mkf B-0 B-3*) ∨
  P (*mkf B-1 B-0*) ∨ *P* (*mkf B-1 B-1*) ∨ *P* (*mkf B-1 B-2*) ∨ *P* (*mkf B-1 B-3*) ∨
  P (*mkf B-2 B-0*) ∨ *P* (*mkf B-2 B-1*) ∨ *P* (*mkf B-2 B-2*) ∨ *P* (*mkf B-2 B-3*) ∨
  P (*mkf B-3 B-0*) ∨ *P* (*mkf B-3 B-1*) ∨ *P* (*mkf B-3 B-2*) ∨ *P* (*mkf B-3 B-3*)
  using *1* **by** *auto*
next
assume *P* (*mkf B-0 B-0*) ∨ *P* (*mkf B-0 B-1*) ∨ *P* (*mkf B-0 B-2*) ∨ *P* (*mkf B-0 B-3*) ∨
  P (*mkf B-1 B-0*) ∨ *P* (*mkf B-1 B-1*) ∨ *P* (*mkf B-1 B-2*) ∨ *P* (*mkf B-1 B-3*) ∨
  P (*mkf B-2 B-0*) ∨ *P* (*mkf B-2 B-1*) ∨ *P* (*mkf B-2 B-2*) ∨ *P* (*mkf B-2 B-3*) ∨
  P (*mkf B-3 B-0*) ∨ *P* (*mkf B-3 B-1*) ∨ *P* (*mkf B-3 B-2*) ∨ *P* (*mkf B-3 B-3*)
thus ∃*f*::*Enum.finite-2* ⇒ *Enum.finite-2 set* . *P f*
  by *auto*
qed

lemma *counter-10*:
∃*R*::(*Enum.finite-2*,*Enum.finite-2*) *mrel* . ¬ (*U*::(*Enum.finite-2*,*Enum.finite-2*) *mrel*) ∗ (*U* ∗ *R*) ⊆ *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 (*clarsimp simp*: *mr-simp ex-2-cases all-2-cases impl-2-cases*)
by *auto*

lemma *counter-11*:
∃(*R*::(*Enum.finite-2*,*Enum.finite-2*) *mrel*) (*s*::(*Enum.finite-2*,*Enum.finite-2*) *mrel*) (*t*::(*Enum.finite-2*,*Enum.finite-2*) *mrel*) . ¬ (*inner-univalent s* ∧ *inner-univalent t* ⟶ *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 (*clarsimp simp*: *mr-simp ex-2-cases all-2-cases impl-2-cases*)
by (*auto simp*: *times-eq-iff*)

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-ild 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.