Cardinality and Representation of Stone Relation Algebras

Walter Guttmann

September 26, 2023

Abstract

In relation algebras, which model unweighted graphs, the cardinality operation counts the number of edges of a graph. We generalise the cardinality axioms to Stone relation algebras, which model weighted graphs, and study the relationships between various axioms for cardinality. We also give a representation theorem for Stone relation algebras.

Contents

1 Representation of Stone Relation Algebras 2
  1.1 Ideals and Ideal-Points ................................. 4
  1.2 Point Axiom ........................................... 11
  1.3 Ideals, Ideal-Points and Matrices as Types ........... 13
  1.4 Isomorphism ............................................ 18

2 Atoms Below an Element in Partial Orders 27

3 Atoms Below an Element in Stone Relation Algebras 35
  3.1 Atomic .................................................. 41
  3.2 Atom-rectangular ....................................... 44
  3.3 Atomic and Atom-Rectangular ......................... 47
  3.4 Atom-simple ............................................ 48
  3.5 Atomic and Atom-simple ............................... 48
  3.6 Atom-rectangular and Atom-simple .................... 49
  3.7 Atomic, Atom-rectangular and Atom-simple .......... 51
  3.8 Finitely Many Atoms ..................................... 52
  3.9 Atomic and Finitely Many Atoms ...................... 52
  3.10 Atom-rectangular and Finitely Many Atoms .......... 53
  3.11 Atomic, Atom-rectangular and Finitely Many Atoms ... 53
  3.12 Atom-simple and Finitely Many Atoms ............... 54
  3.13 Atomic, Atom-simple and Finitely Many Atoms ....... 54
The theories formally verify results in [1]. See this papers for further details and related work. Stone relation algebras have been introduced in [2] and formalised in [3].

Theories Representation

theory Representation

imports Stone-Relation-Algebras Matrix-Relation-Algebras

begin

1 Representation of Stone Relation Algebras

We show that Stone relation algebras can be represented by matrices if we assume a point axiom. The matrix indices and entries and the point axiom are based on the concepts of ideals and ideal-points. We start with general results about sets and finite suprema.

lemma finite-ne-subset-induct' [consumes 3, case-names singleton insert]:
assumes finite F
  and F ≠ { }
  and F ⊆ S
  and singleton: ∀ x . x ∈ S → P { x }
  and insert: ∀ x F . finite F → F ≠ { } → F ⊆ S → x ∈ S → x /∈ F
⇒ P F ⇒ P ( insert x F )
shows P F
using assms(1-3)
apply (induct rule: finite-ne-induct)
apply (simp add: singleton)
by (simp add: singleton)

context order-bot
begin

abbreviation atom :: 'a ⇒ bool
where atom x ≡ x ≠ bot ∧ (∀ y . y ≠ bot ∧ y ≤ x → y = x)

end
context semilattice-sup
begin

lemma nested-sup-fin:
  assumes finite X
  and X ≠ {} and finite Y and Y ≠ {}
  shows Sup-fin { Sup-fin { f x y | x . x ∈ X } | y . y ∈ Y } =
                Sup-fin { f x y | x y . x ∈ X ∧ y ∈ Y }
proof (rule order.antisym)
  have 1: finite { f x y | x y . x ∈ X ∧ y ∈ Y }
  proof
    have finite (X × Y) by (simp add: assms(1,3))
    hence finite { f (fst z) (snd z) | z . z ∈ X × Y }
      by (metis (mono-tags) Collect-mem_eq finite-image-set)
    thus ?thesis by auto
  qed
  show Sup-fin { Sup-fin { f x y | x . x ∈ X } | y . y ∈ Y } ≤
                Sup-fin { f x y | x y . x ∈ X ∧ y ∈ Y }
  apply (rule Sup-fin.boundedI)
  subgoal by (simp add: assms(3))
  subgoal using assms(4) by blast
  subgoal for a proof
    assume a ∈ { Sup-fin { f x y | x . x ∈ X } | y . y ∈ Y }
    from this obtain y where 2: y ∈ Y ∧ a =
                Sup-fin { f x y | x . x ∈ X }
      by auto
    have Sup-fin { f x y | x . x ∈ X } ≤
                Sup-fin { f x y | x y . x ∈ X ∧ y ∈ Y }
      by (simp add: assms(1))
    apply (rule Sup-fin.boundedI)
    subgoal using assms(2) by blast
    subgoal using Sup-fin.coboundedI 1 2 by blast
    done
    thus ?thesis using 2 by simp
  qed
  done
  show Sup-fin { f x y | x y . x ∈ X ∧ y ∈ Y } ≤
                Sup-fin { Sup-fin { f x y | x . x ∈ X } | y . y ∈ Y }
  apply (rule Sup-fin.boundedI)
  subgoal using 1 by simp
  subgoal using assms(2,4) by blast
  subgoal for a proof
    assume a ∈ { f x y | x y . x ∈ X ∧ y ∈ Y }
    from this obtain x y where 3: x ∈ X ∧ y ∈ Y ∧ a = f x y
  qed
  done


by auto
have \( a \leq \text{Sup-fin} \{ f \ x \ y \ | \ x \cdot x \in X \} \)
  apply (rule Sup-fin.coboundedI)
  apply (simp add: assms(1))
  using 3 by blast
also have \( ... \leq \text{Sup-fin} \{ \text{Sup-fin} \{ f \ x \ y \ | \ x \cdot x \in X \} \ | \ y \cdot y \in Y \} \)
  apply (rule Sup-fin.coboundedI)
  apply (simp add: assms(3))
  using 3 by blast
finally show \( a \leq \text{Sup-fin} \{ \text{Sup-fin} \{ f \ x \ y \ | \ x \cdot x \in X \} \ | \ y \cdot y \in Y \} \)
  .
qed
done
qed

end

context bounded-semilattice-sup-bot
begin

lemma one-point-sup-fin:
  assumes finite X
  and \( y \in X \)
  shows \( \text{Sup-fin} \{ (\text{if } x = y \text{ then } f \ x \text{ else } \text{bot}) \ | \ x \ . \ x \in X \} = f y \)
proof (rule order.antisym)
  show \( \text{Sup-fin} \{ (\text{if } x = y \text{ then } f \ x \text{ else } \text{bot}) \ | \ x \ . \ x \in X \} \leq f y \)
    apply (rule Sup-fin.boundedI)
    apply (simp add: assms(1))
    using assms(2) apply blast
    by auto
  show \( f y \leq \text{Sup-fin} \{ (\text{if } x = y \text{ then } f \ x \text{ else } \text{bot}) \ | \ x \ . \ x \in X \} \)
    apply (rule Sup-fin.coboundedI)
    using assms by auto
qed

end

1.1 Ideals and Ideal-Points

We study ideals in Stone relation algebras, which are elements that are both
a vector and a covector. We include general results about Stone relation
algebras.

context times-top
begin

abbreviation ideal :: 'a \Rightarrow \text{bool} where \( \text{ideal } x \equiv \text{vector } x \land \text{covector } x \)

end
context bounded-non-associative-left-semiring
begin

lemma ideal-fixpoint:
ideal x \iff top \cdot x \cdot top = x
by (metis order.antisym top-left-mult-increasing top-right-mult-increasing)

lemma ideal-top-closed:
ideal top
by simp
end

context bounded-idempotent-left-semiring
begin

lemma ideal-mult-closed:
ideal x = \Rightarrow ideal y = \Rightarrow ideal (x \cdot y)
by (metis mult-assoc)
end

context bounded-idempotent-left-zero-semiring
begin

lemma ideal-sup-closed:
ideal x = \Rightarrow ideal y = \Rightarrow ideal (x \sqcup y)
by (simp add: covector-sup-closed vector-sup-closed)
end

context idempotent-semiring
begin

lemma sup-fin-sum:
fixes f :: 'b::finite \Rightarrow 'a
shows \( \text{Sup-fin } \{ f x \mid x . x \in UNIV \} = (\bigsqcup x f x) \)
proof (rule order.antisym)
show \( \text{Sup-fin } \{ f x \mid x . x \in UNIV \} \leq (\bigsqcup x f x) \)
apply (rule Sup-fin.boundedI)
apply (metis (mono-tags) finite finite-image-set)
apply blast
using ub-sum by auto
next
show \( (\bigsqcup x f x) \leq \text{Sup-fin } \{ f x \mid x . x \in UNIV \} \)
apply (rule lub-sum, rule allI)
apply (rule Sup-fin.coboundedI)
apply (metis (mono-tags) finite finite-image-set)
by auto
end
lemma dedekind-univalent:
  assumes anivalent y
  shows x ∩ y ≤ (x ∩ y ≤ y) ∩ y
proof (rule order.antisym)
  show x ∩ y ≤ (x ∩ y ≤ y) ∩ y
    by (simp add: dedekind-2)
next
  have (x ∩ y ≤ y) ∩ y ≤ x ∩ y
    using comp-left-subdist-inf by auto
  also have ... ≤ x ∩ y
    by (metis assms comp-associative comp-inf.mult-right-isotone comp-right-one
        mult-right-isotone)
  finally show (x ∩ y ≤ y) ∩ y ≤ x ∩ y
  qed

lemma dedekind-injective:
  assumes injective x
  shows x ∩ y ∩ z = x ∩ (y ∩ x ∩ z)
proof (rule order.antisym)
  show x ∩ y ∩ z ≤ x ∩ (y ∩ x ∩ z)
    by (simp add: dedekind-1)
next
  have x ∩ (y ∩ x ∩ z) ≤ x ∩ y ∩ x ∩ z
    using comp-associative comp-right-subdist-inf by auto
  also have ... ≤ x ∩ y ∩ z
    by (metis assms coreflexive-comp-top-inf inf.boundedE inf.boundedI
        inf.cobounded2 inf.le1)
  finally show x ∩ (y ∩ x ∩ z) ≤ x ∩ y ∩ z
  qed

lemma domain-vector-conv:
  1 ∩ x * top = 1 ∩ x * x^T
  by (metis conv-dist-comp one-inf-cone symmetric-top-closed)

lemma domain-vector-covector:
  1 ∩ x * top = 1 ∩ top * x^T
  by (metis conv-dist-comp one-inf-cone symmetric-top-closed)

lemma domain-covector-conv:
1 \cap \top * x^T = 1 \cap x * x^T

using domain-vector-conv domain-vector-covector by auto

lemma ideal-bot-closed:
ideal bot
by simp

lemma ideal-inf-closed:
ideal x \Rightarrow ideal y \Rightarrow ideal (x \cap y)
by (simp add: covector-comp-inf vector-inf-comp)

lemma ideal-comp-closed:
ideal x \Rightarrow ideal (x^T)
using covector-conv-vector vector-cone-covector by blast

lemma ideal-complement-closed:
ideal x \Rightarrow ideal (-x)
by (simp add: covector-complement-closed vector-complement-closed)

lemma ideal-cone-id:
ideal x \Rightarrow x = x^T
by (metis covector-comp-inf-1 inf.sup-monoid.add-commute inf-top.right-neutral mult-left-one vector-inf-comp)

lemma ideal-mult-inf:
ideal x \Rightarrow ideal y \Rightarrow x * y = x \cap y
by (metis inf-top-right vector-inf-comp)

lemma ideal-mult-import:
ideal x \Rightarrow y * z \cap x = (y \cap x) * (z \cap x)
using covector-comp-inf inf.sup-monoid.add-commute vector-inf-comp by auto

lemma point-meet-one:
point x \Rightarrow x * x^T = x \cap 1
by (metis domain-vector-conv inf.absorb2 inf.sup-monoid.add-commute)

lemma below-point-eq-domain:
point x \Rightarrow y \leq x \Rightarrow y = x * x^T * y
by (metis inf.absorb2 vector-export-comp-unit point-meet-one)

lemma covector-mult-vector-ideal:
vector x \Rightarrow vector z \Rightarrow ideal (x^T * y * z)
by (metis comp-associative vector-conv-covector)

abbreviation ideal-point :: 'a => bool where ideal-point x \equiv point x \land (\forall y z. point y \land ideal z \land z \neq bot \land y * z \leq x \Rightarrow y \leq x)

lemma different-ideal-points-disjoint:
assumes ideal-point p
and ideal-point q
and p \neq q
shows p \cap q = \bot

proof (rule ccontr)
  let ?r = p^T * (p \cap q)
  assume 1: p \cap q \neq \bot
  have 2: p \cap q = p * ?r
    by (metis assms(1) comp-associative inf.left-idem vector-export-comp-unit
      point-meet-one)
  have ideal ?r
    by (meson assms assms(1,2) covector-mult-closed vector-conv-covector
      vector-inf-closed vector-mult-closed)
  hence p \leq q
    using 1 2 by (metis assms inf-le2 semiring.mult-not-zero)
  hence False
    by (metis assms dual-order.eq_iff epm-3)
qed

lemma points-disjoint-iff:
assumes vector x
shows x \cap y = \bot \iff x^T \ast y = \bot
by (metis assms inf-top-right Schroeder-1)

lemma different-ideal-points-disjoint-2:
assumes ideal-point p
and ideal-point q
and p \neq q
shows p^T \ast q = \bot
using assms different-ideal-points-disjoint points-disjoint-iff by blast

lemma mult-right-dist-sup-fin:
assumes finite X
and X \neq {}
shows \{ f x \mid x \in X \} \ast y = \{ f x \ast y \mid x \in X \}
proof (rule finite-ne-induct[where F=X])
  show finite X
    using assms(1) by simp
  show X \neq {}
    using assms(2) by simp
  show \bigwedge \{ z \ast \{ f x \mid x \in \{ z \} \} \ast y = \{ f x \ast y \mid x \in \{ z \} \}
    by auto
  fix z F
  assume 1: finite F F \neq {} z \notin F \{ f x \mid x \in F \} \ast y = \{ f x \ast y \mid x \in F \}
  have \{ f x \mid x \in \text{insert} z F \} = \text{insert} (f z) \{ f x \mid x \in F \}
    by auto
  hence \{ f x \mid x \in \text{insert} z F \} \ast y = (f z \cup \{ f x \mid x \in F \}) \ast y
    using Sup-fin.insert 1 by auto
also have ... = \( f z \ast y \sqcup \text{Sup-fin} \{ f x \mid x \in F \} \ast y \)
   using \textit{mult-right-dist-sup} by \textbf{blast}
also have ... = \( f z \ast y \sqcup \text{Sup-fin} \{ f x \ast y \mid x \in F \} \)
   using \textit{1} by \textbf{simp}
also have ... = \text{Sup-fin} \{ f x \ast y \mid x \in \text{insert} \ z \ F \}
   using \textit{1} by \textbf{auto}
also have ... = \text{Sup-fin} \{ f x \ast y \mid x \in \text{insert} \ z \ F \}
   by (rule \textit{arg-cong[where } f = \text{Sup-fin}, \textbf{auto})
finally show \text{Sup-fin} \{ f x \mid x \in \text{insert} \ z \ F \} \ast y = \text{Sup-fin} \{ f x \ast y \mid x \in \text{insert} \ z \ F \}
\)

\textbf{qed}

\textbf{lemma} \textit{mult-left-dist-sup-fin}:
\textbf{assumes} \text{finite} \ X
\textbf{and} \ X \neq \{\}
\textbf{shows} \ y \ast \text{Sup-fin} \{ f x \mid x::'b \ . \ x \in X \} = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in X \}
\textbf{proof} (\textbf{rule} \textit{finite-ne-induct[where } F=\text{X}])
\textbf{show} \text{finite} \ X
   using \textbf{assms}(\textbf{1}) by \textbf{simp}
\textbf{show} \ X \neq \{\}
   using \textbf{assms}(2) by \textbf{simp}
\textbf{show} \bigwedge z . \ y \ast \text{Sup-fin} \{ f x \mid x \ . \ x \in \{z\} \} = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in \{z\} \}
   by \textbf{auto}
\textbf{fix} \ z \ F
\textbf{assume I:} \text{finite} \ F \ F \neq \{\} \ z \notin \ F \ y \ast \text{Sup-fin} \{ f x \mid x \ . \ x \in F \} = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in F \}
\textbf{have} \{ f x \mid x \ . \ x \in \text{insert} \ z \ F \} = \text{insert} (f z) \{ f x \mid x \ . \ x \in F \}
   by \textbf{auto}
\textbf{hence} \ y \ast \text{Sup-fin} \{ f x \mid x \ . \ x \in \text{insert} \ z \ F \} = y \ast (f z \sqcup \text{Sup-fin} \{ f x \mid x \in F \})
   using \text{Sup-fin.insert} \textbf{ 1} by \textbf{auto}
\textbf{also have} ... = y \ast f z \sqcup y \ast \text{Sup-fin} \{ f x \mid x \ . \ x \in F \}
   using \textit{mult-left-dist-sup} by \textbf{blast}
\textbf{also have} ... = y \ast f z \sqcup \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in F \}
   using \textit{1} by \textbf{simp}
\textbf{also have} ... = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in \text{insert} \ z \ F \}
   using \textit{1} by \textbf{auto}
\textbf{also have} ... = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in \text{insert} \ z \ F \}
   by (rule \textit{arg-cong[where } f = \text{Sup-fin}, \textbf{auto})
\textbf{finally show} \ y \ast \text{Sup-fin} \{ f x \mid x \ . \ x \in \text{insert} \ z \ F \} = \text{Sup-fin} \{ y \ast f x \mid x \ . \ x \in \text{insert} \ z \ F \}
\)

\textbf{qed}

\textbf{lemma} \textit{inf-left-dist-sup-fin}:
\textbf{assumes} \text{finite} \ X
\textbf{and} \ X \neq \{\}
\textbf{shows} \ y \sqcap \text{Sup-fin} \{ f x \mid x::'b \ . \ x \in X \} = \text{Sup-fin} \{ y \sqcap f x \mid x \ . \ x \in X \}
proof (rule finite-ne-induct[where F=X])
  show finite X
    using assms(1) by simp
  show X ≠ {}
    using assms(2) by simp
  show \( \forall z . \ y \cap \text{Sup-fin} \{ \ f x \mid x \in \{z\} \} = \text{Sup-fin} \{ \ y \cap f x \mid x \in \{z\} \} \)
    by auto
  fix z F
  assume 1: finite F F ≠ {} z /∈ F y \cap \text{Sup-fin} \{ \ f x \mid x \in F \} = \text{Sup-fin} \{ \ y \cap f x \mid x \in F \}
  have \{ f x \mid x \in \text{insert} z F \} = \text{insert} (f z) \{ f x \mid x \in F \}
    by auto
  hence y \cap \text{Sup-fin} \{ f x \mid x \in \text{insert} z F \} = y \cap (f z \cap \text{Sup-fin} \{ f x \mid x \in F \})
    using Sup-fin.insert 1 by auto
  also have ... = \text{Sup-fin} \{ y \cap f x \mid x \in \text{insert} z F \}
    using inf-top-right by auto
  also have ... = (y \cap f z) \cup (y \cap \text{Sup-fin} \{ f x \mid x \in F \})
    using inf-sup-distrib1 by auto
  also have ... = (y \cap f z) \cup \text{Sup-fin} \{ y \cap f x \mid x \in F \}
    using 1 by simp
  also have ... = \text{Sup-fin} (\text{insert} (y \cap f z) \{ y \cap f x \mid x \in F \})
    using 1 by auto
  also have ... = \text{Sup-fin} \{ y \cap f x \mid x \in \text{insert} z F \}
    by (rule arg-cong[where f = \text{Sup-fin}], auto)
  finally show y \cap \text{Sup-fin} \{ f x \mid x \in \text{insert} z F \} = \text{Sup-fin} \{ y \cap f x \mid x \in F \}
    using inf.sup-monoid.add-commute
  qed

lemma top-one-sup-fin-iff:
assumes finite P
  and P ≠ {} and \( \forall p \in P \). point p
shows top = \text{Sup-fin} P \leftrightarrow 1 = \text{Sup-fin} \{ p * p^T \mid p \in P \}
proof
  assume top = \text{Sup-fin} P
  hence 1 = \text{Sup-fin} P
    using inf-top-right by auto
  also have ... = \text{Sup-fin} \{ 1 \cap p \mid p \in P \}
    using inf-sup1-distrib assms(1,2) by simp
  also have ... = \text{Sup-fin} \{ p * p^T \mid p \in P \}
    by (metis (no-types, opaque-lifting) point-meet-one assms(3)
inf.sup-monoid.add-commute)
  finally show 1 = \text{Sup-fin} \{ p * p^T \mid p \in P \}
    by auto

next
  assume 1 = \text{Sup-fin} \{ p * p^T \mid p \in P \}
  hence top = \text{Sup-fin} \{ p * p^T \mid p \in P \} * top
    using total-one-closed by auto
  also have ... = \text{Sup-fin} \{ 1 \cap p \mid p \in P \} * top

by (metis (no-types, opaque-lifting) point-meet-one assms(3)
inf.sup-monoid.add-commute)
also have ... = Sup-fin { (1 \cap p) * top | p . p \in P }
  using mult-right-dist-sup-fin assms(1,2) by auto
also have ... = Sup-fin { p | p . p \in P }
  by (metis (no-types, opaque-lifting) assms(3) inf.sup-monoid.add-commute
inf.top.right-neutral vector-inf-one-comp)
finally show top = Sup-fin P
  by simp
qed

abbreviation ideals :: 'a set where ideals \equiv \{ x . ideal x \}
abbreviation ideal-points :: 'a set where ideal-points \equiv \{ x . ideal-point x \}

lemma surjective-vector-top:
  surjective x \implies vector x \implies x^T * x = top
  by (metis domain-vector-conv covector-inf-comp-3 ex231a
inf.sup-monoid.add-commute inf-top.left-neutral vector-export-comp-unit)

lemma point-mult-top:
  point x \implies x^T * x = top
  using surjective-vector-top by blast

end

1.2 Point Axiom
The following class captures the point axiom for Stone relation algebras.

class stone-relation-algebra-pa = stone-relation-algebra +
  assumes finite-ideal-points: finite ideal-points
  assumes ne-ideal-points: ideal-points \neq \{\}
  assumes top-sup-ideal-points: top = Sup-fin ideal-points
begin

lemma one-sup-ideal-points:
  1 = Sup-fin \{ p * p^T | p . ideal-point p \}
proof -
  have 1 = Sup-fin \{ p * p^T | p . p \in ideal-points \}
    using top-one-sup-fin-iff finite-ideal-points ne-ideal-points top-sup-ideal-points
  by blast
  also have ... = Sup-fin \{ p * p^T | p . ideal-point p \}
  by simp
  finally show ?thesis
  .
qed

lemma ideal-point-rep-1:
  x = Sup-fin \{ p * p^T * x * q * q^T | p q . ideal-point p \land ideal-point q \}
proof -
let \( ?p = \{ p \ast p^T \mid p . p \in \text{ideal-points} \} \)

have \( x = \text{Sup-fin} \ ?p \ast (x \ast \text{Sup-fin} \ ?p) \)
using one-sup-ideal-points by auto
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} \ ?p \mid p . p \in \text{ideal-points} \)
apply \( \text{rule mult-right-dist-sup-fin} \)
using finite-ideal-points ne-ideal-points by simp-all
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} \ ?p \mid p . p \in \text{ideal-points} \)
using mult-assoc by simp
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast (q \ast q^T) \mid q \ast q \in \text{ideal-points} \mid p . p \in \text{ideal-points} \)
proof
- have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} ?p = \text{Sup-fin} \ { p \ast p^T \ast x \ast q \ast q^T \mid q \ast q \in \text{ideal-points} \mid p . p \in \text{ideal-points} \}
apply \( \text{rule mult-left-dist-sup-fin} \)
using finite-ideal-points ne-ideal-points by simp-all
thus \( \ldots \)thesis
using mult-assoc by simp
qed
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} \ ?p \mid p . p \in \text{ideal-points} \)
apply \( \text{rule nested-sup-fin} \)
using finite-ideal-points ne-ideal-points by simp-all
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} \ ?p \mid p . p \in \text{ideal-points} \)
by meson
also have \( \ldots \) = \( \text{Sup-fin} \ ?p \ast x \ast \text{Sup-fin} \ ?p \mid p . p \in \text{ideal-points} \)
by auto
finally show \( \ldots \)thesis
.
qed

lemma atom-below-ideal-point:
assumes atom a
shows \( \exists p . \text{ideal-point} p \land a \leq p \)
proof
- have a = a \( \cap \) \text{Sup-fin} \ { p \mid p . p \in \text{ideal-points} \}
using top-sup-ideal-points by auto
also have \( \ldots \) = \( \text{Sup-fin} \ { a \cap p \mid p . p \in \text{ideal-points} \}
apply \( \text{rule inf-left-dist-sup-fin} \)
using finite-ideal-points apply blast
using ne-ideal-points by blast
finally have 1: \( \text{Sup-fin} \ { a \cap p \mid p . p \in \text{ideal-points} \} \neq \text{bot} \)
using assms by auto
have \( \exists p \in \text{ideal-points} . a \cap p \neq \text{bot} \)
proof \( \text{rule ccontr} \)
assume \( \neg (\exists p \in \text{ideal-points} . a \cap p \neq \text{bot}) \)
hence \( \forall p \in \text{ideal-points} . a \cap p = \text{bot} \)
by auto

d12
hence \{ a \cap p \mid p \cdot p \in \text{ideal-points} \} = \{ \bot \mid p \cdot p \in \text{ideal-points} \}
by auto

hence Sup-fin \{ a \cap p \mid p \cdot p \in \text{ideal-points} \} = Sup-fin \{ \bot \mid p \cdot p \in \text{ideal-points} \}
by simp
also have ... \leq \bot
apply (rule Sup-fin.boundedI)
apply (simp add: finite-ideal-points)
using ne-ideal-points apply simp
by blast
finally show False
using 1 le-bot by blast
qed

from this obtain p where p \in \text{ideal-points} \land a \cap p \neq \bot
by auto
hence ideal-point p \land a \leq p
using assms inf.absorb-iff1 inf-le1 by blast
thus ?thesis
by auto
qed

end

1.3 Ideals, Ideal-Points and Matrices as Types

Stone relation algebras will be represented by matrices with ideal-points as entries and ideals as indices. To define the type of such matrices, we first derive types for the set of ideals and ideal-points.

typedef (overloaded) 'a ideal = ideals::'a::stone-relation-algebra-pa set
using surjective-top-closed by blast

setup-lifting type-definition-ideal

instantiation ideal :: (stone-relation-algebra-pa) stone-algebra
begin

lift-definition uminus-ideal :: 'a ideal \Rightarrow 'a ideal is uminus
using ideal-complement-closed by blast

lift-definition inf-ideal :: 'a ideal \Rightarrow 'a ideal \Rightarrow 'a ideal is inf
by (simp add: ideal-inf-closed)

lift-definition sup-ideal :: 'a ideal \Rightarrow 'a ideal \Rightarrow 'a ideal is sup
by (simp add: ideal-sup-closed)

lift-definition bot-ideal :: 'a ideal is bot
by (simp add: ideal-bot-closed)

lift-definition top-ideal :: 'a ideal is top
lift-definition less-eq-ideal :: 'a ideal ⇒ 'a ideal ⇒ bool is less-eq.

lift-definition less-ideal :: 'a ideal ⇒ 'a ideal ⇒ bool is less.

instance
  apply intro-classes
  subgoal apply transfer by (simp add: less-le-not-le)
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by simp
  subgoal apply transfer by (simp add: sup-inf-distrib1)
  subgoal apply transfer by (simp add: pseudo-complement)
  subgoal apply transfer by simp
  done

end

instantiation ideal :: (stone-relation-algebra-pa) stone-relation-algebra
 begin

lift-definition conv-ideal :: 'a ideal ⇒ 'a ideal is id
  by simp

lift-definition times-ideal :: 'a ideal ⇒ 'a ideal ⇒ 'a ideal is inf
  by (simp add: ideal-inf-closed)

lift-definition one-ideal :: 'a ideal is top
  by simp

instance
  apply intro-classes
  apply (metis comp-inf.comp-associative inf-ideal-def times-ideal-def)
  apply (metis inf-commute inf-ideal-def inf-sup-distrib1 times-ideal-def)
  apply (metis ( mono-tags , lifting ) comp-inf.mult-left-zero inf-ideal-def
    times-ideal-def)
  apply (metis ( mono-tags , opaque-lifting ) comp-inf.mult-1-left inf-ideal-def
    one-ideal.abs-eq times-ideal-def top-ideal.abs-eq)
  using Rep-ideal-inject conv-ideal.rep-eq apply fastforce
  apply (metis ( mono-tags ) Rep-ideal-inverse conv-ideal.rep-eq)
apply (metis (mono-tags) Rep-ideal-inverse conv-ideal.rep-eq inf-commute inf-ideal-def times-ideal-def)  
apply (metis (mono-tags, opaque-lifting) Rep-ideal-inverse conv-ideal.rep-eq inf-ideal-def le-inf-iff order-refl times-ideal-def)  
apply (metis inf-ideal-def p-dist-inf p-dist-sup times-ideal-def)  
by (metis (mono-tags) one-ideal.abs-eq regular-closed-top top-ideal-def)

end

typedef (overloaded) 'a ideal-point = ideal-points::'a::stone-relation-algebra-pa set  
using ne-ideal-points by blast

instantiation ideal-point :: (stone-relation-algebra-pa) finite
begin
instance
proof
have Abs-ideal-point ' ideal-points = UNIV
  using type-definition.Abs-image type-definition-ideal-point by blast
thus finite (UNIV::'a ideal-point set)
  by (metis (mono-tags, lifting) finite-ideal-points finite-imageI)
qed
end

type-synonym 'a ideal-matrix = ('a ideal-point, 'a ideal)

interpretation ideal-matrix-stone-relation-algebra: stone-relation-algebra where
sup = sup-matrix and inf = inf-matrix and less-eq = less-eq-matrix and less = less-matrix and bot = bot-matrix::'a::stone-relation-algebra-pa ideal-matrix and
top = top-matrix and aminus = aminus-matrix and one = one-matrix and
times = times-matrix and conv = conv-matrix
  by (rule matrix-stone-relation-algebra.stone-relation-algebra-axioms)

lemma ideal-point-rep-2:
  assumes x = Sup-fin { Rep-ideal-point p * Rep-ideal (f p q) * (Rep-ideal-point q)T | p q . True }
  shows f r s = Abs-ideal ((Rep-ideal-point r)T * x * (Rep-ideal-point s))
proof –
  let ?r = Rep-ideal-point r
  let ?s = Rep-ideal-point s
  have ?rT * x * ?s = ?rT * Sup-fin { Rep-ideal-point p * Rep-ideal (f p q) * (Rep-ideal-point q)T | p q . True } * ?s
    using assms by simp
  also have ... = ?rT * Sup-fin { Rep-ideal-point p * Rep-ideal (f p q) * (Rep-ideal-point q)T | p q . p ∈ UNIV ∧ q ∈ UNIV } * ?s
    by simp
  also have ... = ?rT * Sup-fin { Sup-fin { Rep-ideal-point p * Rep-ideal (f p q) *
\[(\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \mid q \cdot q \in \text{UNIV} \} \ast \text{?s}
\]

proof
- have \(\text{Sup-fin} \{ \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \ q \cdot p \in \text{UNIV} \wedge q \in \text{UNIV} \} = \text{Sup-fin} \{ \text{Sup-fin} \{ \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \mid q \cdot q \in \text{UNIV} \}\)
  by (rule nested-sup-fin[symmetric], simp-all)
thus ?thesis
by simp

also have \(\ldots = \text{Sup-fin} \{ \text{Sup-fin} \{ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \mid q \cdot q \in \text{UNIV} \} \ast \text{?s}\)

proof
- have 1: \(?r^T \ast \text{Sup-fin} \{ \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} = \text{Sup-fin} \{ \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \)
  by (rule mult-left-dist-sup-fin, simp-all)
- have 2: \(\bigwedge q \cdot ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T = \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T\)
  by (simp add: mult.assoc)
thus ?thesis
using 1 2 by simp

also have \(\ldots = \text{Sup-fin} \{ \text{Sup-fin} \{ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \ast \text{?s} \mid p \cdot p \in \text{UNIV} \} \mid q \cdot q \in \text{UNIV} \} \)

proof
- have 3: \(\text{Sup-fin} \{ \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \ast \text{?s} = \text{Sup-fin} \{ \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \mid p \cdot p \in \text{UNIV} \} \ast \text{?s} \mid q \cdot q \in \text{UNIV} \} \)
  by (rule mult-right-dist-sup-fin, simp-all)
- have \(\bigwedge q \cdot \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \ast \text{?s} \mid p \cdot p \in \text{UNIV} \} \)
  by (rule mult-right-dist-sup-fin, simp-all)
thus ?thesis
using 3 by simp

also have \(\ldots = \text{Sup-fin} \{ \text{Sup-fin} \{ \text{if } p = r \text{ then } ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal} \ (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \ast \text{?s else bot} \mid p \cdot p \in \text{UNIV} \} \mid q \cdot q \in \text{UNIV} \} \)

proof
- have \(\bigwedge p \cdot ?r^T \ast \text{Rep-ideal-point } p = (\text{if } p = r \text{ then } ?r^T \ast \text{Rep-ideal-point } p \text{ else bot})\)
  proof –
fix $p$

show $?r^T \ast \text{Rep-ideal-point } p = (\text{if } p = r \text{ then } ?r^T \ast \text{Rep-ideal-point } p \text{ else } \text{bot})$

proof (cases $p = r$)
  case True
  thus $?thesis$
    by auto
next
  case False
  have $?r^T \ast \text{Rep-ideal-point } p = \text{bot}$
    apply (rule different-ideal-points-disjoint-2)
    using $\text{Rep-ideal-point}$
    apply blast
    using False by (simp add: $\text{Rep-ideal-point-inject}$)
  thus $?thesis$
    using False by simp
  qed
qed
hence $\bigwedge p \ q . \ ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal } (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \ast ?s = (\text{if } p = r \text{ then } ?r^T \ast \text{Rep-ideal-point } p \ast \text{Rep-ideal } (f \ p \ q) \ast (\text{Rep-ideal-point } q)^T \ast ?s \text{ else } \text{bot})$
  by (metis semiring.mult-zero-left)
  thus $?thesis$
    by simp
  qed
also have $\ldots = \text{Sup-fin } \{ \ ?r^T \ast ?r \ast \text{Rep-ideal } (f \ r \ q) \ast (\text{Rep-ideal-point } q)^T \ast ?s \mid q . \ q \in \text{UNIV} \}$
  by (subst one-point-sup-fin, simp-all)
also have $\ldots = \text{Sup-fin } \{ \ \text{if } q = s \text{ then } ?r^T \ast ?r \ast \text{Rep-ideal } (f \ r \ q) \ast (\text{Rep-ideal-point } q)^T \ast ?s \text{ else } \text{bot} \mid q . \ q \in \text{UNIV} \}$
proof
  have $\bigwedge q . \ (\text{Rep-ideal-point } q)^T \ast ?s = (\text{if } q = s \text{ then } (\text{Rep-ideal-point } q)^T \ast ?s \text{ else } \text{bot})$
    proof
      fix $q$
      show $\text{Sup-fin } \{ \ ?r^T \ast ?r \ast \text{Rep-ideal } (f \ r \ q) \ast (\text{Rep-ideal-point } q)^T \ast ?s \mid q . \ q \in \text{UNIV} \}$
        by (subst one-point-sup-fin, simp-all)
    qed
  qed
also have $\ldots = \text{Sup-fin } \{ \ \text{if } q = s \text{ then } (\text{Rep-ideal-point } q)^T \ast ?s \text{ else } \text{bot} \mid q . \ q \in \text{UNIV} \}$
proof (cases $q = s$)
  case True
  thus $?thesis$
    by auto
next
  case False
  show $\text{Rep-ideal-point } q)^T \ast ?s = (\text{if } q = s \text{ then } (\text{Rep-ideal-point } q)^T \ast ?s \text{ else } \text{bot})$
    proof (cases $q = r$)
      case True
      thus $?thesis$
        by auto
next
  case False
  have $(\text{Rep-ideal-point } q)^T \ast ?s = \text{bot}$
    apply (rule different-ideal-points-disjoint-2)
    using $\text{Rep-ideal-point}$
    apply blast
    using False by (simp add: $\text{Rep-ideal-point-inject}$)
  thus $?thesis$
using False by simp

qed
definition sra-to-mat :: "a::stone-relation-algebra-pa "ideal-matrix" where sra-to-mat x ≡ λ(p, q). Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point q)
definition mat-to-sra :: "a::stone-relation-algebra-pa "ideal-matrix" ⇒ "a where mat-to-sra f ≡ Sup-fin { Rep-ideal-point p * Rep-ideal (f (p,q)) * (Rep-ideal-point q)T | p q . True }

lemma sra-mat-sra:
  mat-to-sra (sra-to-mat x) = x

proof –
  have mat-to-sra (sra-to-mat x) = Sup-fin { Rep-ideal-point p * Rep-ideal (Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point q)) * (Rep-ideal-point q)T | p q . True }
    by (unfold sra-to-mat-def mat-to-sra-def, simp)
  also have ... = Sup-fin { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T | p q . True }
    proof –
      have ∧ p q . ideal ((Rep-ideal-point p)T * x * Rep-ideal-point q)
        using Rep-ideal-point covector-mult-vector-ideal by force

1.4 Isomorphism

The following two functions comprise the isomorphism between Stone relation algebras and matrices. We prove that they are inverses of each other and that the first one is a homomorphism.

lemma sra-mat-sra: 
  mat-to-sra (sra-to-mat x) = x

proof –
  have mat-to-sra (sra-to-mat x) = Sup-fin { Rep-ideal-point p * Rep-ideal (Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point q)) * (Rep-ideal-point q)T | p q . True }
    by (unfold sra-to-mat-def mat-to-sra-def, simp)
  also have ... = Sup-fin { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T | p q . True }
    proof –
      have ∧ p q . ideal ((Rep-ideal-point p)T * x * Rep-ideal-point q)
        using Rep-ideal-point covector-mult-vector-ideal by force
hence \( \bigwedge p \ q \ . \ \text{Rep-ideal} \ ((\text{Abs-ideal} ((\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q)) = ((\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q) \)

using Abs-ideal-inverse by blast

thus \(?thesis\)

by (simp add: mult.assoc)

qed

also have \(... = \sup-\text{fin} \ \{ \ p * p^T * x * q * q^T \ | \ p q \ . \ \text{ideal-point} p \land \text{ideal-point} q \} \)

proof (rule set-eqI)

fix \( z \)

show \( z \in \{ \ \text{Rep-ideal-point} p * (\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q * (\text{Rep-ideal-point} q)^T \ | \ p q . \ True \} \leftrightarrow z \in \{ \ p * p^T * x * q * q^T \ | \ p q . \ \text{ideal-point} p \land \text{ideal-point} q \} \)

proof

assume \( z \in \{ \ \text{Rep-ideal-point} p * (\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q * (\text{Rep-ideal-point} q)^T \ | \ p q . \ True \} \)

from this obtain \( p q \ where \ z = \text{Rep-ideal-point} p * (\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q * (\text{Rep-ideal-point} q)^T \)

by auto

thus \( z \in \{ \ p * p^T * x * q * q^T \ | \ p q . \ \text{ideal-point} p \land \text{ideal-point} q \} \)

using Rep-ideal-point by blast

next

assume \( z \in \{ \ p * p^T * x * q * q^T \ | \ p q . \ \text{ideal-point} p \land \text{ideal-point} q \} \)

from this obtain \( p q \ where \ 1: \ \text{ideal-point} p \land \text{ideal-point} q \land z = p * p^T * x * q * q^T \)

by auto

hence \( \text{Rep-ideal-point} (\text{Abs-ideal-point} p) = p \land \text{Rep-ideal-point} (\text{Abs-ideal-point} q) = q \)

using Abs-ideal-point-inverse by auto

thus \( z \in \{ \ \text{Rep-ideal-point} p * (\text{Rep-ideal-point} p)^T * x * \text{Rep-ideal-point} q * (\text{Rep-ideal-point} q)^T \ | \ p q . \ True \} \)

using 1 by (metis (mono-tags, lifting) mem-Collect-eq)

qed

qed

thus \(?thesis\)

by simp

qed

also have \(... = x \)

by (rule ideal-point-rep-1[ symmetric])

finally show \(?thesis\)

qed

lemma \mat-sra-mat:

\( \text{sra-to-mat} (\text{mat-to-sra} f) = f \)
by (unfold sra-to-mat-def, simp add:
ideal-point-rep-2[symmetric])

lemma sra-to-mat-sup-homomorphism:
  sra-to-mat (x ∪ y) = sra-to-mat x ∪ sra-to-mat y
proof (rule ext, unfold split-paired-all)
 fix p q
 have sra-to-mat (x ∪ y) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * (x ∪ y) *
 Rep-ideal-point q)
   by (unfold sra-to-mat-def, simp)
 also have ... = Abs-ideal ((Rep-ideal-point p)^T * x * Rep-ideal-point q ∪
 (Rep-ideal-point p)^T * y * Rep-ideal-point q)
   by (simp add: comp-right-dist-sup
 idempotent-left-zero-semiring-class,semiring,distrib-left)
 also have ... = Abs-ideal ((Rep-ideal-point p)^T * x * Rep-ideal-point q) \n Abs-ideal ((Rep-ideal-point p)^T * y * Rep-ideal-point q)
 proof (rule sup-ideal,abs-eq[symmetric])
   have 1: \(\forall\ x . \text{ideal-point} ((\text{Rep-ideal-point\ } x :\:\text{'}a)\)
 using Rep-ideal-point by blast
 hence 2: \(\text{covector} ((\text{Rep-ideal-point\ } p)^T\)
     using vector-conv-covector by blast
 thus eq-onp ideal ((\text{Rep-ideal-point\ } p)^T * x * Rep-ideal-point q)
 (\text{Rep-ideal-point\ } p)^T * x * Rep-ideal-point q)
     using 1 by (simp add: comp-associative covector-mult-closed
 eq-onp-same-vars)
   show eq-onp ideal ((\text{Rep-ideal-point\ } p)^T * y * Rep-ideal-point q)
 (\text{Rep-ideal-point\ } p)^T * y * Rep-ideal-point q)
     using 1 2 by (simp add: comp-associative covector-mult-closed
 eq-onp-same-vars)
 qed
 also have ... = sra-to-mat x (p,q) ∪ sra-to-mat y (p,q)
   by (unfold sra-to-mat-def, simp)
 finally show sra-to-mat (x ∪ y) (p,q) = (sra-to-mat x ∪ sra-to-mat y) (p,q)
   by simp
 qed

lemma sra-to-mat-inf-homomorphism:
  sra-to-mat (x ∩ y) = sra-to-mat x ∩ sra-to-mat y
proof (rule ext, unfold split-paired-all)
 fix p q
 have sra-to-mat (x ∩ y) (p,q) = Abs-ideal ((\text{Rep-ideal-point\ } p)^T * (x ∩ y) *
 Rep-ideal-point q)
   by (unfold sra-to-mat-def, simp)
 also have ... = Abs-ideal ((\text{Rep-ideal-point\ } p)^T * x * Rep-ideal-point q ∩
 (\text{Rep-ideal-point\ } p)^T * y * Rep-ideal-point q)
   by (metis (no-types, lifting) Rep-ideal-point conv-involutive
 injective-comp-right-dist-inf mem-Collect-eq univalent-comp-left-dist-inf)
 also have ... = Abs-ideal ((\text{Rep-ideal-point\ } p)^T * x * Rep-ideal-point q) ∩
 Abs-ideal ((\text{Rep-ideal-point\ } p)^T * y * Rep-ideal-point q)
proof (rule inf-ideal.abs-eq[symmetric])
  have 1: ∀x. ideal-point (Rep-ideal-point x::a)
    using Rep-ideal-point by blast
  hence 2: covector ((Rep-ideal-point p)^T)
    using vector-cov-covector by blast
  thus eq-op ideal ((Rep-ideal-point p)^T * x * Rep-ideal-point q)
    ((Rep-ideal-point p)^T * x * Rep-ideal-point q)
    using 1 by (simp add: comp-associative covector-mult-closed
      eq-opn-same-args)
  show eq-op ideal ((Rep-ideal-point p)^T * y * Rep-ideal-point q)
    ((Rep-ideal-point p)^T * y * Rep-ideal-point q)
    using 1 2 by (simp add: comp-associative covector-mult-closed
      eq-opn-same-args)
  qed
also have ...
  = sra-to-mat x (p,q) ∩ sra-to-mat y (p,q)
  by (unfold sra-to-mat-def, simp)
finally show sra-to-mat (x ∩ y) (p,q) = (sra-to-mat x ∩ sra-to-mat y) (p,q)
  by simp
qed

lemma sra-to-mat-conv-homomorphism:
  sra-to-mat (x^T) = (sra-to-mat x)^T
proof (rule ext,unfold split-paired-all)
  fix p q
  have sra-to-mat (x^T) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * (x^T) *
    Rep-ideal-point q)
    by (unfold sra-to-mat-def, simp)
  also have ...
    = Abs-ideal (((Rep-ideal-point q)^T * x * Rep-ideal-point p)^T)
    by (simp add: conv-dist-comp mult.assoc)
  also have ...
    = Abs-ideal ((Rep-ideal-point point q)^T * x * Rep-ideal-point p)
  proof
    have ideal-point (Rep-ideal-point point p) ∧ ideal-point (Rep-ideal-point q)
      using Rep-ideal-point by blast
    thus ?thesis
      by (metis (full-types) covector-mult-vector-ideal ideal-conv-id)
  qed
also have ...
  = (Abs-ideal (((Rep-ideal-point q)^T * x * Rep-ideal-point p))^T)
  by (metis Rep-ideal-inject conv-ideal.rep-eq)
also have ...
  = (sra-to-mat x (q,p))^T
  by (unfold sra-to-mat-def, simp)
finally show sra-to-mat (x^T) (p,q) = ((sra-to-mat x)^T) (p,q)
  by (simp add: conv-matrix-def)
qed

lemma sra-to-mat-complement-homomorphism:
  sra-to-mat (−x) = −(sra-to-mat x)
proof (rule ext,unfold split-paired-all)
  fix p q
  have sra-to-mat (−x) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * −x *
Rep-ideal-point q
by (unfold sra-to-mat-def, simp)
also have ... = Abs-ideal (−((Rep-ideal-point p)^T * x * Rep-ideal-point q))
proof
have 1: (Rep-ideal-point p)^T * −x = −((Rep-ideal-point p)^T * x)
    using Rep-ideal-point comp-mapping-complement surjective-conv-total by force
have −((Rep-ideal-point p)^T * x) * Rep-ideal-point q = −((Rep-ideal-point p)^T * x * Rep-ideal-point q)
    using Rep-ideal-point comp-bijective-complement by blast
thus ?thesis
using 1 by simp
qed
also have ...
also have ... = −Abs-ideal ((Rep-ideal-point p)^T * x * Rep-ideal-point q)
proof (rule uminus-ideal.abs-eq[symmetric])
  have 1: \( \forall x . \ \text{ideal-point} \ (\text{Rep-ideal-point} \ x ::'a) \) 
    using Rep-ideal-point by blast
  hence covector ((Rep-ideal-point p)^T)
    using vector-conv-covector by blast 
  thus eq-onp ideal ((Rep-ideal-point p)^T * x * Rep-ideal-point q)
  ((Rep-ideal-point p)^T * x * Rep-ideal-point q)
    using 1 by (simp add: comp-associative covector-mult-closed 
        eq-onp-same-args)
  qed
also have ...
  by (unfold sra-to-mat-def, simp)
finally show sra-to-mat (−x) (p,q) = (−sra-to-mat x) (p,q)
  by simp
qed

lemma sra-to-mat-bot-homomorphism:
sra-to-mat bot = bot
proof (rule ext, unfold split-paired-all)
  fix p q :: 'a ideal-point
  have sra-to-mat bot (p,q) = Abs-ideal ((Rep-ideal-point p)^T * bot * Rep-ideal-point q)
    by (unfold sra-to-mat-def, simp)
  also have ...
    by (simp add: bot-ideal.abs-eq)
  finally show sra-to-mat bot (p,q) = bot (p,q)
    by simp
qed

lemma sra-to-mat-top-homomorphism:
sra-to-mat top = top
proof (rule ext, unfold split-paired-all)
  fix p q :: 'a ideal-point
  have sra-to-mat top (p,q) = Abs-ideal ((Rep-ideal-point p)^T * top * Rep-ideal-point q)
  qed

22
by (unfold sra-to-mat-def, simp)
also have ... = top
proof –
  have \( \forall x . \text{ideal-point (Rep-ideal-point } x::'a) \)
    using Rep-ideal-point by blast
  thus \( \text{thesis} \)
    by (metis (full-types) conv-dist-comp symmetric-top-closed top-ideal.abs-eq)
qed
finally show sra-to-mat top \( (p,q) \) = top \( (p,q) \)
by simp
qed

lemma sra-to-mat-one-homomorphism:
\( \text{sra-to-mat} \ 1 = \text{one-matrix} \)
proof (rule ext, unfold split-paired-all)
fix \( p \) \( q \) :: 'a ideal-point
have sra-to-mat \( 1 \) \( (p,q) \) = Abs-ideal ((\( \text{Rep-ideal-point } p \))^{T} \ast \text{Rep-ideal-point } q)
  by (unfold sra-to-mat-def, simp)
also have ... = one-matrix \( (p,q) \)
proof (cases \( p = q \))
  case True
    hence \( (\text{Rep-ideal-point } p)^{T} \ast \text{Rep-ideal-point } q = \text{top} \)
      using Rep-ideal-point point-mult-top by auto
    hence Abs-ideal ((\( \text{Rep-ideal-point } p \))^{T} \ast \text{Rep-ideal-point } q) = Abs-ideal top
      by simp
    also have ... = one-matrix \( (p,q) \)
      by (unfold one-matrix-def, simp add: True one-ideal-def)
  finally show \( \text{thesis} \).
  next
  case False
    have \( (\text{Rep-ideal-point } p)^{T} \ast \text{Rep-ideal-point } q = \text{bot} \)
      apply (rule different-ideal-points-disjoint-2)
      using Rep-ideal-point apply blast
      using Rep-ideal-point apply blast
      by (simp add: False Rep-ideal-point-inject)
    also have ... = one-matrix \( (p,q) \)
      by (unfold one-matrix-def, simp add: False)
    finally show \( \text{thesis} \)
      by (simp add: False bot-ideal-def one-matrix-def)
  qed
finally show sra-to-mat \( 1 \) \( (p,q) \) = one-matrix \( (p,q) \)
by simp
qed

lemma Abs-ideal-dist-sup-fin:
assumes finite \( X \)
  and \( X \neq \{ \} \)
  and \( \forall x \in X . \text{ideal (f } x) \)
shows Abs-ideal (Sup-fin \{ f x \mid x . x \in X \}) = Sup-fin \{ Abs-ideal (f x) \mid x . x \in X \}

proof (rule finite-ne-subset-induct \[where F=X\])
  show finite X
    using assms(1) by simp
  show X \neq \{\}
    using assms(2) by simp
  show X \subseteq X
    by simp
  fix y
  assume 1: y \in X
  thus Abs-ideal (Sup-fin \{ f x \mid x . x \in \{y\} \}) = Sup-fin \{ Abs-ideal (f x) \mid x . x \in \{y\} \}
    by auto
  fix F
  assume 2: finite F F \neq \{\} F \subseteq X y \notin F Abs-ideal (Sup-fin \{ f x \mid x . x \in F \}) = Sup-fin \{ Abs-ideal (f x) \mid x . x \in F \}
  have Abs-ideal (Sup-fin \{ f x \mid x . x \in insert y F \}) = Abs-ideal (f y \sqcup Sup-fin \{ f x \mid x . x \in \})
    proof
      have Sup-fin \{ f x \mid x . x \in insert y F \} = f y \sqcup Sup-fin \{ f x \mid x . x \in F \}
        apply (subst Sup-fin.insert[ symmetric])
        using 2 apply simp
        using 2 apply simp
        by (auto intro: arg-cong \[where f=Sup-fin\])
      thus \?thesis
        by simp
    qed
  also have ... = Abs-ideal (f y) \sqcup Abs-ideal (Sup-fin \{ f x \mid x . x \in F \})
    proof (rule sup-ideal.abs-eq[ symmetric])
      show eq-onp ideal (f y) (f y)
        using 1 by (simp add: assms(3) eq-onp-same-args)
      have top \ast Sup-fin \{ f x \mid x . x \in F \} = Sup-fin \{ top \ast f x \mid x . x \in F \}
        using 2 mult-left-dist-sup-fin by fastforce
      hence top \ast Sup-fin \{ f x \mid x . x \in F \} \ast top = Sup-fin \{ top \ast f x \mid x . x \in F \}
        * top
        by simp
      also have ... = Sup-fin \{ top \ast f x \ast top \mid x . x \in F \}
        using 2 mult-right-dist-sup-fin by force
      also have ... = Sup-fin \{ f x \mid x . x \in F \}
        using 2 by (metis assms(3) subset-iff)
      finally have top \ast Sup-fin \{ f x \mid x . x \in F \} \ast top = Sup-fin \{ f x \mid x . x \in F \}
    .
    hence ideal (Sup-fin \{ f x \mid x . x \in F \})
      using ideal-fixpoint by blast
    thus eq-onp ideal (Sup-fin \{ f x \mid x . x \in F \}) (Sup-fin \{ f x \mid x . x \in F \})
      by (simp add: eq-onp-def)
  qed
also have ... = Abs-ideal (f y) ⊔ Sup-fin { Abs-ideal (f x) | x . x ∈ F }

using 2 by simp
also have ... = Sup-fin { Abs-ideal (f x) | x . x ∈ insert y F }
apply (subst Sup-fin.insert[symmetric])
using 2 apply simp
using 2 apply simp
by (auto intro: arg-cong[where f=Sup-fin])
finally show Abs-ideal (Sup-fin { f x | x . x ∈ insert y F }) = Sup-fin { Abs-ideal (f x) | x . x ∈ insert y F }
.
qed

lemma sra-to-mat-mult-homomorphism:
sra-to-mat (x * y) = sra-to-mat x ⊙ sra-to-mat y
proof (rule ext, unfold split-paired-all)
fix p q
have sra-to-mat (x * y) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * (x * y) * Rep-ideal-point q)
  by (unfold sra-to-mat-def, simp)
also have ... = Abs-ideal ((Rep-ideal-point p)^T * x * 1 * y * Rep-ideal-point q)
  by (simp add: mult.assoc)
also have ... = Abs-ideal ((Rep-ideal-point p)^T * x * Sup-fin { r * r^T | r . ideal-point r }) * y * Rep-ideal-point q)
  by (unfold one-sup-ideal-points[symmetric], simp)
also have ... = Abs-ideal ((Rep-ideal-point p)^T * x * Sup-fin { Rep-ideal-point r * (Rep-ideal-point r)^T | r . r ∈ UNIV } * y * Rep-ideal-point q)
proof
  have { r * r^T | r::'a . ideal-point r } = { Rep-ideal-point r * (Rep-ideal-point r)^T | r . r ∈ UNIV }
  proof (rule set-eqI)
    fix x
    show x ∈ { r * r^T | r::'a . ideal-point r } ✴= x ∈ { Rep-ideal-point r * (Rep-ideal-point r)^T | r . r ∈ UNIV }
    proof
      assume x ∈ { r * r^T | r::'a . ideal-point r }
      from this obtain r where 1: ideal-point r ∧ x = r * r^T
      by auto
      hence Rep-ideal-point (Abs-ideal-point r) = r
        using Abs-ideal-point-inverse by auto
      thus x ∈ { Rep-ideal-point r * (Rep-ideal-point r)^T | r . r ∈ UNIV }
        using 1 by (metis (mono-tags, lifting) UNIV-I mem-Collect-eq)
    next
      assume x ∈ { Rep-ideal-point r * (Rep-ideal-point r)^T | r . r ∈ UNIV }
      from this obtain r where x = Rep-ideal-point r * (Rep-ideal-point r)^T
      by auto
      hence Rep-ideal-point (Abs-ideal-point r) = r
        using Rep-ideal-point by blast
      qed
    qed
  qed
thus \?thesis

by simp

qed

also have \ldots = Abs-ideal (\text{Sup-fin} \{ (\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
\ast (\text{Rep-ideal-point } r)^T \mid r \cdot r \in \text{UNIV} \} \ast (y \ast \text{Rep-ideal-point } q))

by (subst mult-left-dist-sup-fin, simp-all add: mult.assoc)

also have \ldots = Abs-ideal (\text{Sup-fin} \{ (\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
\ast (\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q \mid r \cdot r \in \text{UNIV} \})

by (subst mult-right-dist-sup-fin, simp-all add: mult.assoc)

also have \ldots = \text{Sup-fin} \{ Abs-ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
\ast (\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q) \mid r \cdot r \in \text{UNIV} \}

proof -

have 1: \(\forall r. \text{ideal} ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
(\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q)

proof -

fix r :: 'a ideal-point

have \(\forall x. \text{ideal-point} (\text{Rep-ideal-point } x::'a)

using Rep-ideal-point by blast

thus ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r \ast (\text{Rep-ideal-point }
\text{Rep-ideal-point } q)

by (simp add: covector-mult-closed vector-conv-covector vector-mult-closed)

qed

show ?thesis

apply (rule Abs-ideal-dist-sup-fin)

using 1 by simp-all

qed

also have \ldots = (\bigsqcup_r Abs-ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
(\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q)

by (rule sup-fin-sum)

also have \ldots = (\bigsqcup_r Abs-ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r
\ast (\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q)

proof (rule ideal-mult-inf)

fix r :: 'a ideal-point

have 2: \(\forall x. \text{ideal-point} (\text{Rep-ideal-point } x::'a)

using Rep-ideal-point by blast

thus ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r)

by (simp add: covector-mult-closed vector-conv-covector vector-mult-closed)

show ideal ((\text{Rep-ideal-point } r)^T \ast y \ast \text{Rep-ideal-point } q)

using 2 by (simp add: covector-mult-closed vector-conv-covector vector-mult-closed)

qed

thus ?thesis

by (simp add: mult.assoc)

qed

also have \ldots = (\bigsqcup_r Abs-ideal ((\text{Rep-ideal-point } p)^T \ast x \ast \text{Rep-ideal-point } r) \ast
Abs-ideal \(((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q)\)

proof -
have \(\bigwedge r . \text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r \cap \text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q\)

\((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q\) = \(\text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r) * \text{Abs-ideal } ((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q)\)

proof (rule times-ideal.abs-eq[symmetric])
fix \(r::'a \text{ ideal-point}\)
have 3: \(\bigwedge x . \text{ideal-point} (\text{Rep-ideal-point } x::'a)\)
using Rep-ideal-point by blast
hence 4: \(\text{covector} ((\text{Rep-ideal-point } p)^T) \wedge \text{covector} ((\text{Rep-ideal-point } r)^T)\)
using vector-conv-covector by blast
thus eq-onp ideal \(((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r)\)
\(((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r)\)
using 3 by (simp add: comp-associative covector-mult-closed eq-onp-same-args)
show eq-onp ideal \(((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q)\)
\(((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q)\)
using 3 4 by (simp add: comp-associative covector-mult-closed eq-onp-same-args)
qed
thus ?thesis
by simp
qed
also have ...
finally show \(\text{sra-to-mat } (x * y) \circ \text{sra-to-mat } (r, q)\)
by (simp add: times-matrix-def)
qed

end

theory Cardinality

imports List-Infinite.InfiniteSet2 Representation

begin

context uminus
begin

no-notation uminus (' - [81] 80)

end

2 Atoms Below an Element in Partial Orders

We define the set and the number of atoms below an element in a partial order. To handle infinitely many atoms we use \textit{enat}, which are natural numbers with infinity, and \textit{icard}, which modifies \textit{card} by giving a separate
option of being infinite. We include general results about \textit{enat}, \textit{icard}, sets functions and atoms.

\textbf{Lemma} \texttt{enat-mult-strict-mono}:  
\begin{itemize}
  \item \textbf{Assumes} $a < b$ and $c < d$ and $(0::\texttt{enat}) < b$ and $0 \leq c$
  \item \textbf{Shows} $a \times c < b \times d$
\end{itemize}
\textbf{Proof} –
\begin{itemize}
  \item \textbf{Have} $a \neq \infty \wedge c \neq \infty$
  \item \textbf{Using} \texttt{assms(1,2)} \texttt{linorder-not-le} \textbf{by} \texttt{fastforce}
  \item \textbf{Thus} \texttt{?thesis}
  \item \textbf{Using} \texttt{assms} \textbf{by} \texttt{(smt (verit, del-insts) enat-0-less-mult-iff idiff-eq-conv-enat ileI1 imult-ile-mono imult-is-infinity-enat less-eq-idiff-eq-sum less-le-not-le mult-eSuc-right order.strict-trans1 order-le-neq-trans zero-enat-def)}
\end{itemize}
\textbf{Qed}

\textbf{Lemma} \texttt{enat-mult-strict-mono}':  
\begin{itemize}
  \item \textbf{Assumes} $a < b$ and $c < d$ and $(0::\texttt{enat}) \leq a$ and $0 \leq c$
  \item \textbf{Shows} $a \times c < b \times d$
\end{itemize}
\textbf{Using} \texttt{assms} \textbf{by} \texttt{(auto simp add: enat-mult-strict-mono)}

\textbf{Lemma} \texttt{finite-icard-card}:  
\begin{itemize}
  \item \texttt{finite} $A \implies \texttt{icard} A = \texttt{icard} B \implies \texttt{card} A = \texttt{card} B$
\end{itemize}
\textbf{By} \texttt{(metis icard-def icard-eq-enat-imp-card)}

\textbf{Lemma} \texttt{icard-eq-sum}:  
\begin{itemize}
  \item \texttt{finite} $A \implies \texttt{icard} A = \texttt{sum} \lambda x. 1\ A$
\end{itemize}
\textbf{By} \texttt{(simp add: icard-def of-nat-eq-enat)}

\textbf{Lemma} \texttt{icard-sum-constant-function}:  
\begin{itemize}
  \item \texttt{Assumes} $\forall x \in A. f x = c$
  \item \texttt{and} \texttt{finite} $A$
  \item \texttt{Shows} $\texttt{sum} f A = (\texttt{icard} A) \times c$
\end{itemize}
\textbf{By} \texttt{(metis \texttt{asms icard-finite-conv of-nat-eq-enat sum.cong sum-constant)}}

\textbf{Lemma} \texttt{icard-le-finite}:  
\begin{itemize}
  \item \texttt{Assumes} $\texttt{icard} A \leq \texttt{icard} B$
  \item \texttt{and} \texttt{finite} $B$
  \item \texttt{Shows} \texttt{finite} $A$
\end{itemize}
\textbf{By} \texttt{(metis \texttt{asms enat-ord-simps(5) icard-infinite-conv)}}

\textbf{Lemma} \texttt{bij-betw-same-icard}:  
\begin{itemize}
  \item \texttt{bij-betw} $f A B \implies \texttt{icard} A = \texttt{icard} B$
\end{itemize}
\textbf{By} \texttt{(simp add: bij-betw-finite bij-betw-same-card icard-def)}

\textbf{Lemma} \texttt{surj-icard-le}:  
\begin{itemize}
  \item $B \subseteq f\ A \implies \texttt{icard} B \leq \texttt{icard} A$
\end{itemize}
\textbf{By} \texttt{(meson icard-image-le icard-mono preorder-class.order-trans)}

\textbf{Lemma} \texttt{icard-image-part-le}:  
\begin{itemize}
  \item \texttt{Assumes} $\forall x \in A. f x \subseteq B$
  \item \texttt{and} $\forall x \in A. f x \neq \emptyset$
\end{itemize}
and $\forall x \in A \cdot \forall y \in A \cdot x \neq y \longrightarrow f x \cap f y = \{\}$
shows $\text{icard } A \leq \text{icard } B$

proof
have $\forall x \in A \cdot \exists y \cdot y \in f x \cap B$
using assms(1,2) by fastforce
hence $\exists y \cdot \forall x \in A \cdot y \in f x \cap B$
using bchoice by simp
from this obtain $g$ where $1 : \forall x \in A \cdot g x \in f x \cap B$
by auto
hence inj-on $g$ A
by (metis Int-iff assms(3) empty-iff inj-onI)
thus $\text{icard } A \leq \text{icard } B$
using 1 icard-inj-on-le by fastforce
qed

lemma finite-image-part-le:
assumes $\forall x \in A \cdot f x \subseteq B$
and $\forall x \in A \cdot f x \neq \{\}$
and $\forall x \in A \cdot \forall y \in A \cdot x \neq y \longrightarrow f x \cap f y = \{\}$
and finite $B$
shows finite $A$
by (metis assms icard-image-part-le icard-le-finite)

context semiring-1
begin

lemma sum-constant-function:
assumes $\forall x \in A \cdot f x = c$
shows $\text{sum } f A = \text{of-nat } (\text{card } A) * c$

proof (cases finite $A$)
case True
show ?thesis
proof (rule finite-subset-induct)
show finite $A$
using True by simp
show $A \subseteq A$
by simp
show $\text{sum } f \{\} = \text{of-nat } (\text{card } \{\}) * c$
by simp
fix $a$ $F$
assume finite $F$ $a \in A \cdot a \notin F$ $\text{sum } f F = \text{of-nat } (\text{card } F) * c$
thus $\text{sum } f (\text{insert } a F) = \text{of-nat } (\text{card } (\text{insert } a F)) * c$
using assms by (metis sum.insert sum-constant)
qed
next
case False
thus ?thesis
by simp
qed
begin

lemma ne-finite-has-minimal:
  assumes finite \( S \)
  and \( S \neq {} \)
  shows \( \exists m \in S . \forall x \in S . x \leq m \rightarrow x = m \)
proof (rule finite-ne-induct)
  show finite \( S \)
    using assms(1) by simp
  show \( S \neq {} \)
    using assms(2) by simp
  show \( \forall x . \exists m \in \{ x \} . \forall y \in \{ x \} . y \leq m \rightarrow y = m \)
    by auto
  show \( \forall x F . \text{finite } F \rightarrow F \neq {} \rightarrow x \notin F \rightarrow (\exists m \in F . \forall y \in F . y \leq m \rightarrow y = m) \rightarrow (\exists m \in \text{insert } x \ F . \forall y \in \text{insert } x \ F . y \leq m \rightarrow y = m) \)
    by (metis finite-insert insert-not-empty finite-has-minimal)
qed

end

context order
begin

lemma AB-iso:
  \( x \leq y \rightarrow AB x \subseteq AB y \)
by (simp add: Collect-mono dual-orderTrans)

lemma AB-bot:
  \( AB \ bot = {} \)
by (simp add: bot-unique)

lemma nAB-bot:
  \( nAB \ bot = 0 \)
proof
  have \( nAB \ bot = icard (AB \ bot) \)
    by (simp add: num-atoms-below-def)
  also have \( ... = 0 \)
    by (metis mono-tags, lifting AB-bot icard-empty)
finally show \( \text{thesis} \)

end

context order-bot
begin

abbreviation atoms-below :: \( a \Rightarrow \) a set \( AB \)
  where atoms-below \( x \equiv \{ a . \text{atom } a \land a \leq x \} \)

definition num-atoms-below :: \( a \Rightarrow \) enat \( nAB \)
  where num-atoms-below \( x \equiv icard (atoms-below \ x) \)

lemma AB-iso:
  \( x \leq y \rightarrow AB x \subseteq AB y \)
by (simp add: Collect-mono dual-orderTrans)

lemma AB-bot:
  \( AB \ bot = {} \)
by (simp add: bot-unique)

lemma nAB-bot:
  \( nAB \ bot = 0 \)
proof
  have \( nAB \ bot = icard (AB \ bot) \)
    by (simp add: num-atoms-below-def)
  also have \( ... = 0 \)
    by (metis mono-tags, lifting AB-bot icard-empty)
finally show \( \text{thesis} \)

end
lemma AB-atom:
  atom a \iff \text{AB} a = \{a\}
by blast

lemma nAB-atom:
  atom a \implies nAB a = 1
proof -
  assume atom a
  hence \text{AB} a = \{a\}
  using AB-atom by meson
  thus nAB a = 1
  by (simp add: num-atoms-below-def one-eSuc)
qed

lemma nAB-iso:
  x \leq y \implies nAB x \leq nAB y
using icard-mono AB-iso num-atoms-below-def by auto

end

context bounded-semilattice-sup-bot
begin

lemma nAB-iso-sup:
  nAB x \leq nAB (x \sqcap y)
by (simp add: nAB-iso)

end

context bounded-lattice
begin

lemma different-atoms-disjoint:
  atom x \implies atom y \implies x \neq y \implies x \sqcap y = \text{bot}
using inf-le1 le-iff-inf by auto

lemma AB-dist-inf:
  \text{AB} (x \cap y) = \text{AB} x \cap \text{AB} y
by auto

lemma AB-iso-inf:
  \text{AB} (x \cap y) \subseteq \text{AB} x
by (simp add: Collect-mono)

lemma AB-iso-sup:
  \text{AB} x \subseteq \text{AB} (x \sqcup y)
lemma AB-disjoint:
  assumes x ∩ y = bot
  shows AB x ∩ AB y = {}
proof (rule Int-emptyI)
  fix a
  assume a ∈ AB x a ∈ AB y
  hence atom a ∧ a ≤ x ∧ a ≤ y
    by simp
  thus False
  using assms bot-unique by fastforce
qed

lemma nAB-iso-inf:
  nAB (x ∩ y) ≤ nAB x
  by (simp add: nAB-iso)

end

context distrib-lattice-bot
begin

lemma atom-in-sup:
  assumes atom a
      and a ≤ x ⊔ y
  shows a ≤ x ∨ a ≤ y
proof
  have 1: a = (a ⊓ x) ⊔ (a ⊓ y)
    using assms(2) inf-sup-distrib1 le-iff-inf by force
  have a ⊓ x = bot ∨ a ⊓ x = a
    using assms(1) by fastforce
  thus ?thesis
    using 1 le-iff-inf sup-bot-left by fastforce
qed

lemma atom-in-sup-iff:
  assumes atom a
  shows a ≤ x ∨ y ↔ a ≤ x ∧ a ≤ y
using assms atom-in-sup le-supI1 le-supI2 by blast

lemma atom-in-sup-xor:
  atom a =⇒ a ≤ x ∨ y =⇒ x ∧ y = bot =⇒ (a ≤ x ∧ a ≤ y) ∨ (¬ a ≤ x ∧ a ≤ y)
  using atom-in-sup bot-unique le-inf-iff by blast

lemma atom-in-sup-xor-iff:
  assumes atom a
      and x ∩ y = bot
shows $a \leq x \sqcup y \iff (a \leq x \land \neg a \leq y) \lor (\neg a \leq x \land a \leq y)$

using assms atom-in-sup-xor le-supI1 le-supI2 by auto

lemma AB-dist-sup:
$AB (x \sqcup y) = AB x \sqcup AB y$

proof
show $AB (x \sqcup y) \subseteq AB x \sqcup AB y$
  using atom-in-sup by fastforce
next
show $AB x \sqcup AB y \subseteq AB (x \sqcup y)$
  using le-supI1 le-supI2 by fastforce
qed

end

context bounded-distrib-lattice
begin

lemma nAB-add:
$nAB x + nAB y = nAB (x \sqcup y) + nAB (x \sqcap y)$

proof -
  have $nAB x + nAB y = icard (AB x \sqcup AB y) + icard (AB x \sqcap AB y)$
    using num-atoms-below-def icard-Un-Int by auto
  also have ...
    $= nAB (x \sqcup y) + nAB (x \sqcap y)$
    using num-atoms-below-def AB-dist-inf AB-dist-sup by auto
  finally show $\vdash$thesis
  .

qed

lemma nAB-split-disjoint:
  assumes $x \sqcap y = \bot$
  shows $nAB (x \sqcup y) = nAB x + nAB y$
by (simp add: assms nAB-add nAB-bot)

end

context p-algebra
begin

lemma atom-in-p:
atom $a \implies a \leq x \lor a \leq \neg x$
using inf.orderI pseudo-complement by force

lemma atom-in-p-xor:
atom $a \implies (a \leq x \land \neg a \leq \neg x) \lor (\neg a \leq x \land a \leq \neg x)$
by (metis atom-in-p le-iff-inf pseudo-complement)

The following two lemmas also hold in distributive lattices with a least element (see above). However, p-algebras are not necessarily distributive, so the following results are independent.
lemma \textit{atom-in-sup'}:
\[\text{atom } a \implies a \leq x \uplus y \implies a \leq x \lor a \leq y\]
by (metis \textit{inf.absorb-iff2 inf.sup-ge2 pseudo-complement sup-least})

lemma \textit{AB-dist-sup'}:
\[\text{AB } (x \uplus y) = \text{AB } x \uplus \text{AB } y\]
proof
show \[\text{AB } (x \uplus y) \subseteq \text{AB } x \uplus \text{AB } y\]
using \textit{atom-in-sup'} by fastforce
next
show \[\text{AB } x \uplus \text{AB } y \subseteq \text{AB } (x \uplus y)\]
using \textit{le-supI1 le-supI2} by fastforce
qed

lemma \textit{AB-split-1}:
\[\text{AB } x = \text{AB } ((x \sqcap y) \uplus (x \sqcap -y))\]
proof
show \[\text{AB } x \subseteq \text{AB } ((x \sqcap y) \uplus (x \sqcap -y))\]
proof
fix \(a\)
assume \(a \in \text{AB } x\)
hence \[\text{atom } a \land a \leq x\]
by simp
hence \[\text{atom } a \land a \leq (x \sqcap y) \uplus (x \sqcap -y)\]
by (metis \textit{atom-in-p-xor inf.boundedI le-supI1 le-supI2})
thus \(a \in \text{AB } ((x \sqcap y) \uplus (x \sqcap -y))\)
by simp
qed
next
show \[\text{AB } ((x \sqcap y) \uplus (x \sqcap -y)) \subseteq \text{AB } x\]
using \textit{atom-in-sup'} \textit{inf.boundedE} by blast
qed

lemma \textit{AB-split-2}:
\[\text{AB } x = \text{AB } (x \sqcap y) \uplus \text{AB } (x \sqcap -y)\]
using \textit{AB-dist-sup'} \textit{AB-split-1} by auto

lemma \textit{AB-split-2-disjoint}:
\[\text{AB } (x \sqcap y) \cap \text{AB } (x \sqcap -y) = \{\}\]
using \textit{atom-in-p-xor} by fastforce

lemma \textit{AB-pp}:
\[\text{AB } (-x) = \text{AB } x\]
by (metis (opaque-lifting) \textit{atom-in-p-xor})

lemma \textit{nAB-pp}:
\[\text{nAB } (-x) = \text{nAB } x\]
using \textit{AB-pp num-atoms-below-def} by auto
lemma \( nAB\)-split-1:
\[ nAB \, x = nAB \, ((x \cap y) \cup (x \cap -y)) \]
using \( AB\)-split-1 \( \text{num-atoms-below-def} \) by simp

lemma \( nAB\)-split-2:
\[ nAB \, x = nAB \, (x \cap y) + nAB \, (x \cap -y) \]
proof 
  have icard \((AB \, (x \cap y)) + icard \,(AB \,(x \cap -y)) = icard \,(AB \,(x \cap y) \cup AB \,(x \cap -y)) + icard \,(AB \,(x \cap y) \cap AB \,(x \cap -y)) \)
  using icard-Un-Int by auto
  also have ... = icard \,(AB \, x) \)
  using \( AB\)-split-2 \( AB\)-split-2-disjoint by auto
finally show \( \text{thesis} \)
  using \( \text{num-atoms-below-def} \) by auto
qed

end

3 Atoms Below an Element in Stone Relation Algebras

We extend our study of atoms below an element to Stone relation algebras. We consider combinations of the following five assumptions: the Stone relation algebra is atomic, atom-rectangular, atom-simple, a relation algebra, or has finitely many atoms. We include general properties of atoms, rectangles and simple elements.

custom context \( \text{stone-relation-algebra} \)
begin

abbreviation rectangle :: \('a ⇒ bool\) where rectangle \(x \equiv x \ast top \ast x \leq x\)
abbreviation simple :: \('a ⇒ bool\) where simple \(x \equiv top \ast x \ast top = top\)

lemma rectangle-eq:
\[ \text{rectangle} \, x \iff \ast x \ast x = x \]
by (simp add: order.eq-iff ex231d)

lemma arc-univalent-injective-rectangle-simple:
\[ \text{arc} \, a \iff \text{univalent} \, a \land \text{injective} \, a \land \text{rectangle} \, a \land \text{simple} \, a \]
by (smt (z3) arc-top-arc comp-associative conv-dist-comp conv-involutive ideal-top-closed surjective-vector-top rectangle-eq)

lemma conv-atom:
\[ \text{atom} \, x \iff \text{atom} \,(x^T) \]
by (metis conv-involutive conv-isotone symmetric-bot-closed)

lemma conv-atom-iff:
\[ \text{atom} \, x \iff \text{atom} \,(x^T) \]
lemma counterexample-different-atoms-top-disjoint:
atom x \implies atom y \implies x \neq y \implies x \top \cap y = \bot
nitpick[expect=genuine,card=4]
oops

lemma counterexample-different-univalent-atoms-top-disjoint:
atom x \implies univalent x \implies atom y \implies univalent y \implies x \neq y \implies x \top \cap y = \bot
nitpick[expect=genuine,card=4]
oops

lemma AB-card-4-1:
a \leq x \land a \leq y \iff a \leq x \uplus y \land a \leq x \cap y
using le-supI1 by auto

lemma AB-card-4-2:
assumes atom a
shows (a \leq x \land \neg a \leq y) \lor (\neg a \leq x \land a \leq y) \iff a \leq x \uplus y \land \neg a \leq x \cap y
using assms atom-in-sup le-supI1 le-supI2 by auto

lemma AB-card-4-3:
assumes atom a
shows \neg a \leq x \land \neg a \leq y \iff \neg a \leq x \uplus y \land \neg a \leq x \cap y
using assms AB-card-4-2 by auto

lemma AB-card-5-1:
assumes atom a
and a \leq x^T \star y \cap z
shows x \star a \cap y \leq x \star z \cap y
and x \star a \cap y \neq bot
proof
show x \star a \cap y \leq x \star z \cap y
using assms(2) comp-inf.mult-left-isotone mult-right-isotone by auto
show x \star a \cap y \neq bot
by (smt assms inf.left-commute inf.left-idem inf-absorb1 schroeder-1)
qed

lemma AB-card-5-2:
assumes univalent x
and atom a
and atom b
and b \leq x^T \star y \cap z
and a \neq b
shows (x \star a \cap y) \cap (x \star b \cap y) = bot
and x \star a \cap y \neq x \star b \cap y
proof
show (x \star a \cap y) \cap (x \star b \cap y) = bot
by (metis assms(1-3,5) comp-inf semiring.mult-zero-left inf.cobounded1 inf.left-commute inf.sup-monoid.add-commute semiring.mult-not-zero univalent-comp-left-dist-inf)
thus \( x \setminus a \cap y \neq x \setminus b \cap y \)
using AB-card-5-1(2) assms(3,4) by fastforce
qed

lemma AB-card-6-0:
assumes univalent x
and atom a
and \( a \leq x \)
and atom b
and \( b \leq x \)
and \( a \neq b \)
shows \( a \ast top \cap b \ast top = \bot \)
proof –
have \( a \setminus y \ast z \leq 1 \)
by (meson assms(1,3,5) comp-isotone conv-isotone dual-order.trans)
hence \( a \ast top \cap b = \bot \)
by (metis assms(2,4,6) comp-inf semiring.mult-zero-left comp-right-one inf.cobounded1 inf.cobounded2 inf.orderE schroeder-1)
thus \( ?thesis \)
using vector-bot-closed vector-export-comp by force
qed

lemma AB-card-6-1:
assumes atom a
and \( a \leq x \setminus y \ast z^T \)
shows \( a \ast z \setminus y \leq x \ast z \cap y \)
and \( a \ast z \setminus y \neq \bot \)
proof –
show \( a \ast z \setminus y \leq x \ast z \cap y \)
using assms(2) inf.sup-left-isotone mult-left-isotone by auto
show \( a \ast z \setminus y \neq \bot \)
by (metis assms inf.absorb2 inf.boundedE schroeder-2)
qed

lemma AB-card-6-2:
assumes univalent x
and atom a
and \( a \leq x \setminus y \ast z^T \)
and atom b
and \( b \leq x \setminus y \ast z^T \)
and \( a \neq b \)
shows \( (a \ast z \setminus y) \cap (b \ast z \setminus y) = \bot \)
and \( a \ast z \setminus y \neq b \ast z \setminus y \)
proof –
have \( (a \ast z \setminus y) \cap (b \ast z \setminus y) \leq a \ast top \cap b \ast top \)
by (meson comp-inf.comp-isotone comp-inf.ex231d inf.boundedE)
also have \( \ldots = \text{bot} \)
using \(\text{AB-card-6-0} \) assms by force
finally show \((a \ast z \sqcap y) \sqcap (b \ast z \sqcap y) = \text{bot} \)
using \(\text{le-bot} \) by blast
thus \(a \ast z \sqcap y \neq b \ast z \sqcap y\)
using \(\text{AB-card-6-1(2)} \) assms\((4,5) \) by fastforce
qed

lemma \(\text{nAB-conv} \): 
\(\text{nAB } x = \text{nAB } (x^T) \)
proof (unfold \(\text{num-atoms-below-def}\), rule bij-betw-same-icard)
show bij-betw conv \((\text{AB } x) (\text{AB } (x^T)) \)
proof (unfold bij-betw-def, rule conjI)
show inj-on conv \((\text{AB } x) \)
by (metis (mono-tags, lifting) inj-onI conv-involutive)
show \(\text{conv} \ ' \ (\text{AB } x) = \text{AB } (x^T) \)
proof
  fix \(y \)
  assume \(y \in \text{AB } (x^T) \)
  hence \(\text{atom } y \wedge y \leq x^T \)
  by auto
  hence \(\text{atom } (y^T) \wedge y^T \leq x^T \)
  using \(\text{conv-atom-iff} \) \(\text{conv-isotone} \) by force
  hence \(y^T \in \text{AB } x \)
  by auto
  thus \(y \in \text{conv} \ ' \ (\text{AB } x) \)
  by (metis \(\text{no-types}, \) lifting \(\text{image-iff} \) conv-involutive)
qed
qed
qed

lemma \(\text{domain-atom} \): 
assumes \(\text{atom } a \)
shows \(\text{atom } (a \ast \text{top} \sqcap 1) \)
proof
show \(\text{a } \ast \text{top} \sqcap 1 \neq \text{bot} \)
by (metis assms \(\text{domain-vector-conv}\) ex231a \(\text{inf-vector-comp}\) \(\text{mult-left-zero}\) \(\text{vector-export-comp-unit}\))
next
show \(\forall y. y \neq \text{bot} \wedge y \leq a \ast \text{top} \sqcap 1 \rightarrow y = a \ast \text{top} \sqcap 1 \)
proof (rule allI, rule \(\text{impl}\))
  fix \(y \)
  assume \(1: y \neq \text{bot} \wedge y \leq a \ast \text{top} \sqcap 1 \)

38
hence \( 2: y = 1 \cap y \ast a \ast \text{top} \)

using dedekind-injective comp-associative coreflexive-idempotent coreflexive-symmetric inf.absorb2 inf.sup-monoid.add-commute by auto

hence \( y \ast a \neq \text{bot} \)

using 1 comp-inf.semiring.mult-zero-right vector-bot-closed by force

hence \( a = y \ast a \)

using 1 by (metis assms comp-right-one coreflexive-comp-top-inf inf.boundedE mult-sub-right-one)

thus \( y = a \ast \text{top} \cap 1 \)

using 2 inf.sup-monoid.add-commute by auto

qed

lemma codomain-atom:

assumes atom a

shows atom \( (\text{top} \ast a \cap 1) \)

proof

have \( \text{top} \ast a \cap 1 = a^T \ast \text{top} \cap 1 \)

by (simp add: domain-vector-covector inf.sup-monoid.add-commute)

thus \(?\text{thesis}\)

using domain-atom conv-atom assms by auto

qed

lemma atom-rectangle-atom-one-rep:

\( (\forall a . \text{atom } a \longrightarrow a \ast \text{top} \ast a \leq a) \longleftrightarrow (\forall a . \text{atom } a \land a \leq 1 \longrightarrow a \ast \text{top} \ast a \leq 1) \)

proof

assume \( \forall a . \text{atom } a \longrightarrow a \ast \text{top} \ast a \leq a \)

thus \( \forall a . \text{atom } a \land a \leq 1 \longrightarrow a \ast \text{top} \ast a \leq 1 \)

by auto

next

assume 1: \( \forall a . \text{atom } a \land a \leq 1 \longrightarrow a \ast \text{top} \ast a \leq 1 \)

show \( \forall a . \text{atom } a \longrightarrow a \ast \text{top} \ast a \leq a \)

proof (rule allI, rule impl)

fix a

assume atom a

hence atom \( (a \ast \text{top} \cap 1) \)

by (simp add: domain-atom)

hence \( (a \ast \text{top} \cap 1) \ast \text{top} \ast (a \ast \text{top} \cap 1) \leq 1 \)

using 1 by simp

hence \( a \ast \text{top} \ast a^T \leq 1 \)

by (smt comp-associative conv-dist-comp coreflexive-symmetric ex231e inf-top.right-neutral symmetric-top-closed vector-export-comp-unit)

thus \( a \ast \text{top} \ast a \leq a \)

by (smt comp-associative conv-dist-comp domain-vector-conv order.eq-iff ex231e inf.absorb2 inf.sup-monoid.add-commute mapping-one-closed symmetric-top-closed top-right-mult-increasing vector-export-comp-unit)

qed

qed
lemma \(AB\text{-}card\text{-}2\text{-}1\):
\[
\begin{align*}
\text{assumes } & a \ast \top \ast a \leq a \\
\text{shows } & (a \ast \top \cap 1) \ast \top \ast (\top \ast a \cap 1) = a \\
\text{by } & (\text{metis assms comp-inf.vector-top-closed covector-comp-inf ex231d order.antisym inf-commute surjective-one-closed vector-export-comp-unit vector-top-closed mult-assoc})
\end{align*}
\]

lemma \(atoms\text{-}simple\text{-}atom1\text{simple}\):
\[
\begin{align*}
(\forall a . \text{atom} a \longrightarrow \top \ast a \ast \top = \top) & \iff (\forall a . \text{atom} a \land a \leq 1 \longrightarrow \top \ast a \ast \top = \top) \\
\text{proof}
\end{align*}
\]
\[
\begin{align*}
\text{assume } & (\forall a . \text{atom} a \longrightarrow \top \ast a \ast \top = \top) \\
\text{thus } & (\forall a . \text{atom} a \land a \leq 1 \longrightarrow \top \ast a \ast \top = \top) \\
\text{by simp}
\end{align*}
\]

next
\[
\begin{align*}
\text{assume } & 1: (\forall a . \text{atom} a \land a \leq 1 \longrightarrow \top \ast a \ast \top = \top) \\
\text{show } & (\forall a . \text{atom} a \longrightarrow \top \ast a \ast \top = \top) \\
\text{proof (rule allI, rule impI)} \\
\text{fix } & a \\
\text{assume } & \text{atom} a \\
\text{hence } & 2: \text{atom} (a \ast \top \cap 1) \\
\text{by } & (\text{simp add: domain-atom}) \\
\text{have } & \top \ast (a \ast \top \cap 1) \ast \top = \top \ast a \ast \top \\
\text{using } & \text{comp-associative vector-export-comp-unit by auto} \\
\text{thus } & \top \ast a \ast \top = \top \\
\text{using } & 1 \ 2 \text{ by auto} \\
\text{qed}
\end{align*}
\]

\\
\]\n
abbreviation \(\text{dom-cod} :: 'a \Rightarrow 'a \times 'a\)
\[
\text{where}
\]
\[
\begin{align*}
\text{dom-cod } a \equiv (a \ast \top \cap 1, \top \ast a \cap 1)
\end{align*}
\]

lemma \(\text{dom-cod-atoms-1}\):
\[
\begin{align*}
\text{dom-cod } AB \top \subseteq AB \ 1 \times AB \ 1
\end{align*}
\]

40
proof
  fix x
  assume x ∈ dom-cod. AB top
  from this obtain a where 1: atom a ∧ x = dom-cod a
    by auto
  hence a * top ∩ 1 ∈ AB 1 ∧ top ∗ a ∩ 1 ∈ AB 1
    using domain-atom codomain-atom by auto
  thus x ∈ AB 1 × AB 1
    using 1 by auto
qed
end

3.1 Atomic

class stone-relation-algebra-atomic = stone-relation-algebra +
  assumes atomic: x ≠ bot → (∃ a . atom a ∧ a ≤ x)
begin

lemma AB-nonempty:
  x ≠ bot → AB x ≠ { }
  using atomic by fastforce

lemma AB-nonempty-iff:
  x ≠ bot ←→ AB x ≠ { }
  using AB-nonempty AB-bot by blast

lemma atomsimple-simple:
  (∀ a . a ≠ bot → top ∗ a ∗ top = top) ←→ (∀ a . atom a → top ∗ a ∗ top = top)
proof
  assume ∀ a . a ≠ bot → top ∗ a ∗ top = top
  thus ∀ a . atom a → top ∗ a ∗ top = top
    by simp
next
  assume 1: ∀ a . atom a → top ∗ a ∗ top = top
  show ∀ a . a ≠ bot → top ∗ a ∗ top = top
  proof (rule allI, rule implI)
    fix a
    assume a ≠ bot
    from this atomic obtain b where 2: atom b ∧ b ≤ a
      by auto
    hence top ∗ b ∗ top = top
      using 1 by auto
    thus top ∗ a ∗ top = top
      using 2 by (metis order.antisym mult-left-isotone mult-right-isotone top.extremum)
  qed
qed

41
lemma AB-card-2-3:
assumes $a \not= \text{bot}$
and $a \leq 1$
and $b \not= \text{bot}$
and $b \leq 1$
and $\forall a . a \not= \text{bot} \rightarrow \text{top} \ast a \ast \text{top} = \text{top}$
shows $a \ast \text{top} \ast b \ast \text{top} \sqcap 1 = a$ and $\text{top} \ast a \ast \text{top} \ast b \sqcap 1 = b$

proof –
show $a \ast \text{top} \ast b \ast \text{top} \sqcap 1 = a$
  using assms(2,3,5) comp-associative coreflexive-comp-top-inf-one by auto
show $\text{top} \ast a \ast \text{top} \ast b \sqcap 1 = b$
  using assms(1,4,5) epm-3 inf.sup-monoid.add-commute by auto

qed

lemma injective-down-closed:
$x \leq y \Longrightarrow \text{injective } y \Longrightarrow \text{injective } x$
using conv-isotone mult-isotone by fastforce

lemma univalent-down-closed:
$x \leq y \Longrightarrow \text{univalent } y \Longrightarrow \text{univalent } x$
using conv-isotone mult-isotone by fastforce

lemma nAB-bot-iff:
$x = \text{bot} \leftrightarrow nAB x = 0$
by (smt (verit, best) icard-0-eq AB-nonempty-iff num-atoms-below-def)

It is unclear if atomic is necessary for the following two results, but it seems likely.

lemma nAB-univ-comp-meet:
assumes univalent $x$
shows $nAB (x^T \ast y \sqcap z) \leq nAB (x \ast z \sqcap y)$

proof (unfold num-atoms-below-def, rule icard-image-part-le)
show $\forall a \in AB (x^T \ast y \sqcap z) . AB (x \ast a \sqcap y) \subseteq AB (x \ast z \sqcap y)$

proof
  fix $a$
  assume $a \in AB (x^T \ast y \sqcap z)$
  hence $x \ast a \sqcap y \leq x \ast z \sqcap y$
    using AB-card-5-1(1) by auto
  thus $AB (x \ast a \sqcap y) \subseteq AB (x \ast z \sqcap y)$
    using AB-iso by blast
qed

next
show $\forall a \in AB (x^T \ast y \sqcap z) . AB (x \ast a \sqcap y) \neq \{\}$

proof
  fix $a$
  assume $a \in AB (x^T \ast y \sqcap z)$
  hence $x \ast a \sqcap y \neq \text{bot}$
    using AB-card-5-1(2) by auto
thus \( AB \ (x * a \cap y) \neq \{\} \)
using atomic by fastforce
qed
next
show \( \forall a \in AB \ (x^T * y \cap z) . \forall b \in AB \ (x^T * y \cap z) . a \neq b \rightarrow AB \ (x * a \cap y) \cap AB \ (x * b \cap y) = \{\} \)
proof (intro ballI, rule impI)
fix \( a \) \( b \)
assume \( a \in AB \ (x^T * y \cap z) \) \( b \in AB \ (x^T * y \cap z) \) \( a \neq b \)
hence \( (x * a \cap y) \cap (x * b \cap y) = bot \)
using assms AB-card-5-2(1) by auto
thus \( AB \ (x * a \cap y) \cap AB \ (x * b \cap y) = \{\} \)
using AB-bot AB-dist-inf by blast
qed
qed

lemma nAB-univ-meet-comp:
assumes univalent \( x \)
shows \( nAB \ (x \cap y * z^T) \leq nAB \ (x * z \cap y) \)
proof (unfold num-atoms-below-def, rule icard-image-part-le)
show \( \forall a \in AB \ (x \cap y * z^T) . AB \ (a * z \cap y) \subseteq AB \ (x * z \cap y) \)
proof
fix \( a \)
assume \( a \in AB \ (x \cap y * z^T) \)
hence \( a * z \cap y \leq x * z \cap y \)
using AB-card-6-1(1) by auto
thus \( AB \ (a * z \cap y) \subseteq AB \ (x * z \cap y) \)
using AB-iso by blast
qed
next
show \( \forall a \in AB \ (x \cap y * z^T) . AB \ (a * z \cap y) \neq \{\} \)
proof
fix \( a \)
assume \( a \in AB \ (x \cap y * z^T) \)
hence \( a * z \cap y \neq bot \)
using AB-card-6-1(2) by auto
thus \( AB \ (a * z \cap y) \neq \{\} \)
using atomic by fastforce
qed
next
show \( \forall a \in AB \ (x \cap y * z^T) . \forall b \in AB \ (x \cap y * z^T) . a \neq b \rightarrow AB \ (a * z \cap y) \cap AB \ (b * z \cap y) = \{\} \)
proof (intro ballI, rule impl)
fix \( a \) \( b \)
assume \( a \in AB \ (x \cap y * z^T) \) \( b \in AB \ (x \cap y * z^T) \) \( a \neq b \)
hence \( (a * z \cap y) \cap (b * z \cap y) = bot \)
using assms AB-card-6-2(1) by auto
thus \( AB \ (a * z \cap y) \cap AB \ (b * z \cap y) = \{\} \)
using AB-bot AB-dist-inf by blast

3.2 Atom-rectangular

class stone-relation-algebra-atomrect = stone-relation-algebra +
  assumes atomrect: atom a \rightarrow \text{rectangle } a

begin

lemma atomrect-eq:
  atom a \Rightarrow a \ast \text{top} \ast a = a
  by (simp add: order.antisym ex231d atomrect)

lemma AB-card-2-4:
  assumes atom a
  shows (a \ast \text{top} \sqcap 1) \ast \text{top} \ast (\text{top} \ast a \sqcap 1) = a
  by (simp add: assms AB-card-2-1 atomrect)

lemma simple-atom-2:
  assumes atom a
  and a \leq 1
  and atom b
  and b \leq 1
  and x \neq \text{bot}
  and x \leq a \ast \text{top} \ast b
  shows x = a \ast \text{top} \ast b

proof
  have 1: x \ast \text{top} \sqcap 1 \neq \text{bot}
    by (metis assms(5) inf-top-right le-bot top-right-mult-increasing
        vector-bot-closed vector-export-comp-unit)
  have x \ast \text{top} \sqcap 1 \leq a \ast \text{top} \ast b \ast \text{top} \sqcap 1
    using assms(6) comp-inf.comp-isotone comp-isotone by blast
  also have ... \leq a \ast \text{top} \sqcap 1
    by (metis comp-associative comp-inf.mult-right-isotone
        inf.sup-monoid.add-commute mult-right-isotone top.extremum)
  also have ... = a
    by (simp add: assms(2) coreflexive-comp-top-inf-one)
  finally have 2: x \ast \text{top} \sqcap 1 = a
    using 1 by (simp add: assms(1) domain-atom)
  have 3: top \ast x \sqcap 1 \neq \text{bot}
    using 1 by (metis schroeder-1 schroeder-2 surjective-one-closed
        symmetric-top-closed total-one-closed)
  have top \ast x \sqcap 1 \leq top \ast a \ast \text{top} \ast b \sqcap 1
    by (metis assms(6) comp-associative comp-inf.comp-isotone mult-right-isotone
        reflexive-one-closed)
  also have ... \leq top \ast b \sqcap 1
    using inf.sup-monoid mult-left-isotone top-greatest by blast
also have ... = b
  using assms(4) epm-3 inf.sup-monoid.add-commute by auto
finally have top * x ⊓ 1 = b
  using 3 by (simp add: assms(3) codomain-atom)
hence a * top * b = x * top * x
  using 2 by (smt abel-semigroup.commute covector-comp-inf
  inf.abel-semigroup-axioms inf-top-right surjective-one-closed
  vector-export-comp-unit vector-top-closed mult-assoc)
also have ... = a * top * b * top * (x ⊓ a * top * b)
  using assms(6) calculation inf-absorb1 by auto
also have ... ≤ (a * top * (x ⊓ a * top * b))
  by (metis comp-associative comp-inf-covector inf.idem inf.order-iff
  mult-right-isotone)
also have ... ≤ a * top * (x ⊓ a * top)
  using comp-associative comp-inf.mult-right-isotone mult-right-isotone by auto
also have ... = a * top * a * x
  by (metis comp-associative comp-inf-vector inf-top.left-neutral)
also have ... = a * top * a * x
  by (simp add: assms(2) coreflexive-symmetric)
also have ... = a * x
  by (simp add: assms(1) atomrect-eq)
also have ... ≤ x
  using assms(2) mult-left-isotone by fastforce
finally show ?thesis
  using assms(6) order.antisym by blast
qed

lemma dom-cod-inj-atoms:
  inj-on dom-cod (AB top)
proof
  fix a b
  assume 1: a ∈ AB top b ∈ AB top dom-cod a = dom-cod b
  have a = a * top * a
    using 1 atomrect-eq by auto
also have ... = (a * top ⊓ 1) * top * (top * a ⊓ 1)
    using calculation AB-card-2-1 by auto
also have ... = (b * top ⊓ 1) * top * (top * b ⊓ 1)
    using 1 by simp
also have ... = b * top * b
    using abel-semigroup.commute comp-inf-covector inf.abel-semigroup-axioms
    vector-export-comp-unit mult-assoc by fastforce
also have ... = b
    using 1 atomrect-eq by auto
finally show a = b
qed

lemma finite-AB-iff:
  finite (AB top) ←→ finite (AB 1)
proof
  have AB 1 ⊆ AB top
    by auto
  thus finite (AB top) ⟹ finite (AB 1)
    by (meson finite-subset)
next
assume 1: finite (AB 1)
show finite (AB top)
proof (rule inj-on-finite)
  show inj-on dom-cod (AB top)
    using dom-cod-inj-atoms by blast
  show dom-cod 1 AB top ⊆ AB 1 × AB 1
    using dom-cod-atoms-1 by blast
  show finite (AB 1 × AB 1)
    using 1 by blast
qed
qed

lemma nAB-top-1:
  nAB top ≤ nAB 1 × nAB 1
proof (unfold num-atoms-below-def icard-cartesian-product[THEN sym], rule icard-inj-on-le)
  show inj-on dom-cod (AB top)
    using dom-cod-inj-atoms by blast
  show dom-cod 1 AB top ⊆ AB 1 × AB 1
    using dom-cod-atoms-1 by blast
qed

lemma atom-vector-injective:
  assumes atom x
  shows injective (x × top)
proof
  have atom (x × top ∩ 1)
    by (simp add: assms domain-atom)
  hence (x × top ∩ 1) × top × (x × top ∩ 1) ≤ 1
    using atom-rectangle-atom-one-rep atomrect by auto
  hence x × top × xT ≤ 1
    by (smt comp-associative conv-dist-comp coreflexive-symmetric ex231e inf-top.right-neutral symmetric-top-closed vector-export-comp-unit)
  thus injective (x × top)
    by (metis comp-associative conv-dist-comp symmetric-top-closed vector-top-closed vector-top-closed)
qed

lemma atom-injective:
  atom x ⟹ injective x
by (metis atom-vector-injective comp-associative cone-dist-comp dual-order.trans mult-right-isotone symmetric-top-closed top-left-mult-increasing)
lemma atom-covector-univalent:
atom x \rightarrow\ univalent (top \ast x)
by (metis comp-associative conv-involutive atom-vector-injective conv-atom-iff
conv-dist-comp symmetric-top-closed)

lemma atom-univalent:
atom x \rightarrow\ univalent x
using atom-injective conv-atom-iff univalent-conv-injective by blast

lemma counterexample-atom-simple:
atom x \rightarrow\ simple x
nitpick[expect=genuine,card=3]
oops

lemma symmetric-atom-below-1:
assumes atom x
and x = x^T
shows x \leq 1
proof –
have x = x \ast top \ast x^T
  using assms atomrect-eq by auto
also have ... \leq 1
  by (metis assms(1) atom-vector-injective conv-dist-comp equivalence-top-closed ideal-top-closed mult-associative)
finally show ?thesis
.
qed

end

3.3 Atomic and Atom-Rectangular

class stone-relation-algebra-atomic-atomrect = stone-relation-algebra-atomic +
stone-relation-algebra-atomrect
begin

lemma point-dense:
assumes x \neq bot
and x \leq 1
shows \exists a. a \neq bot \land a \ast top \ast a \leq 1 \land a \leq x
proof –
from atomic obtain a where 1: atom a \land a \leq x
  using assms(1) by auto
hence a \ast top \ast a \leq a
  by (simp add: atomrect)
also have ... \leq 1
  using 1 assms(2) order-trans by blast
finally show ?thesis
  using 1 by blast
qed

end

3.4 Atom-simple

class stone-relation-algebra-atomsimple = stone-relation-algebra +
  assumes atomsimple: atom a → simple a
begin

lemma AB-card-2-5:
  assumes atom a
  and a ≤ 1
  and atom b
  and b ≤ 1
  shows a * top * b * top ∩ 1 = a and top * a * top * b ∩ 1 = b
  using assms AB-card-2-2 atomsimple by auto

lemma simple-atom-1:
  atom a =⇒ atom b =⇒ a * top * b ≠ bot
  by (metis order.antisym atomsimple bot-least comp-associative mult-left-zero
top-right-mult-increasing)

end

3.5 Atomic and Atom-simple

class stone-relation-algebra-atomic-atomsimple = stone-relation-algebra-atomic +
  stone-relation-algebra-atomsimple
begin

lemma simple:
  a ≠ bot =⇒ top * a * top = top
  using atomsimple atomsimple-simple by blast

lemma AB-card-2-6:
  assumes a ≠ bot
  and a ≤ 1
  and b ≠ bot
  and b ≤ 1
  shows a * top * b * top ∩ 1 = a and top * a * top * b ∩ 1 = b
  using assms AB-card-2-3 simple atomsimple-simple by auto

lemma dom-cod-atoms-2:
  AB 1 × AB 1 ⊆ dom-cod ' AB top
proof
  fix x
  assume x ∈ AB 1 × AB 1
  from this obtain a b where 1: atom a ∧ a ≤ 1 ∧ atom b ∧ b ≤ 1 ∧ x = (a,b)
  by auto
hence \( a * \text{top} * b \neq \text{bot} \)
  by (simp add: simple-atom-1)
from this obtain \( c \) where 2: \( a \leq a * \text{top} * b \)
  using atomic by blast
hence \( c * \text{top} \cap 1 \leq a * \text{top} \cap 1 \)
  by (smt comp-inf.comp-isotone inf.boundedE inf.orderE inf-vector-comp
  reflexive-one-closed top-right-mult-increasing)
also have ... = \( a \)
  using 1 by (simp add: coreflexive-comp-top-inf-one)
finally have 3: \( c * \text{top} \cap 1 = a \)
  using 1 2 domain-atom by simp
have \( \text{top} * c \leq \text{top} * b \)
  using 2 3 by (smt comp-associative comp-inf.reflexive-top-closed
  comp-inf.vector-top-closed comp-inf-covector comp-isotone simple
  vector-export-comp-unit)
  hence \( \text{top} * c \cap 1 \leq b \)
  using 1 by (smt epm-3 inf.coboundedI inf.left-commute inf.orderE
  injective-one-closed reflexive-one-closed)
  hence \( \text{top} * c \cap 1 = b \)
  using 1 2 codomain-atom by simp
thence \( x \in \text{dom-cod} \ ' \ AB \text{top} \)
  using 2 by auto
qed

lemma dom-cod-atoms:
  \( AB 1 \times AB 1 = \text{dom-cod} \ ' AB \text{top} \)
  using dom-cod-atoms-2 dom-cod-atoms-1 by blast

end

3.6 Atom-rectangular and Atom-simple

class stone-relation-algebra-atomrect-atomsimple =
  stone-relation-algebra-atomrect + stone-relation-algebra-atomsimple
begin

lemma simple-atom:
  assumes atom a
  and \( a \leq 1 \)
  and atom b
  and \( b \leq 1 \)
  shows atom (\( a * \text{top} * b \))
  using assms simple-atom-1 simple-atom-2 by auto

lemma nAB-top-2:
  \( nAB 1 * nAB 1 \leq nAB \text{top} \)
proof (unfold num-atoms-below-def icard-cartesian-product[THEN sym], rule
let \( \varphi = \lambda(a, b). a * \top \) \( b \)

show inj-on \( \varphi \) \((\AB \times \AB)\)

proof

fix \( x \) \( y \)

assume \( x \in \AB \times \AB \) \( y \in \AB \times \AB \)

from this obtain \( a \) \( b \) \( c \) \( d \) where

\[
\begin{array}{l}
\text{1: atom } a \land a \leq 1 \land \text{atom } b \land b \leq 1 \land x = (a, b) \\
\text{2: atom } c \land c \leq 1 \land \text{atom } d \land d \leq 1 \land y = (c, d)
\end{array}
\]

by auto

assume \( \varphi x = \varphi y \)

hence \( 2: a * \top \ast b = c * \top \ast d \)

using 1 by auto

hence \( 3: a = c \)

using 1 by (smt atomparsimple comp-associative coreflexive-comp-top-inf-one)

have \( b = d \)

using 1 2 by (smt atomparsimple comp-associative epm-3 injective-one-closed)

thus \( x = y \)

using 1 3 by simp

qed

show \( \varphi' \) \((\AB \times \AB)\) \(\subseteq \AB \top \)

proof

fix \( x \)

assume \( x \in \varphi' \) \((\AB \times \AB)\)

from this obtain \( a \) \( b \) where

\[
\begin{array}{l}
\text{4: atom } a \land a \leq 1 \land \text{atom } b \land b \leq 1 \land x = a * \top
\end{array}
\]

by auto

hence \( a * \top \ast b \in \AB \top \)

using simple-atom by simp

thus \( x \in \AB \top \)

using 4 by simp

qed

qed

lemma nAB-top:

\( n\AB \times n\AB \) = \( n\AB \top \)

using nAB-top-1 nAB-top-2 by auto

lemma atom-covector-mapping:

\( \text{atom } a \Rightarrow \text{mapping } (\top \ast a) \)

using atom-covector-univalent atomparsimple by blast

lemma atom-covector-regular:

\( \text{atom } a \Rightarrow \text{regular } (\top \ast a) \)

by (simp add: atom-covector-mapping mapping-regular)

lemma atom-vector-bijective:

\( \text{atom } a \Rightarrow \text{bijective } (a \ast \top) \)

using atom-vector-jective comp-associative atomparsimple by auto
lemma atom-vector-regular:
atom a \Rightarrow regular (a \ast top)
by (simp add: atom-vector-bijective bijective-regular)

lemma atom-rectangle-regular:
atom a \Rightarrow regular (a \ast top \ast a)
by (smt atom-covector-regular atom-vector-regular comp-associative
pp-dist-comp regular-closed-top)

lemma atom-regular:
atom a \Rightarrow regular a
using atom-rectangle-regular atomrect-eq
by auto

end

3.7 Atomic, Atom-rectangular and Atom-simple

class stone-relation-algebra-atomic-atomrect-atomsimple =
stone-relation-algebra-atomic + stone-relation-algebra-atomrect +
stone-relation-algebra-atomsimple

begin

subclass stone-relation-algebra-atomic-atomrect ..
subclass stone-relation-algebra-atomic-atomsimple ..
subclass stone-relation-algebra-atomrect-atomsimple ..

lemma nAB-atom-iff:
atom a \longleftrightarrow nAB a = 1
proof
assume atom a
thus nAB a = 1
by (simp add: nAB-atom)
next
assume nAB a = 1
from this obtain b where 1: AB a = \{b\}
using icard-1-imp-singleton num-atoms-below-def one-eSuc
by fastforce
hence 2: atom b \land b \leq a
by auto
hence 3: AB (a \cap b) = \{b\}
by fastforce
have AB (a \cap b) \cup AB (a \cap -b) = AB a \land AB (a \cap b) \cap AB (a \cap -b) = \{}
using AB-split-2 AB-split-2-disjoint by simp
hence \{b\} \cup AB (a \cap -b) = \{b\} \land \{b\} \cap AB (a \cap -b) = \{}
using 1 3 by simp
hence AB (a \cap -b) = \{}
by auto
hence a \cap -b = bot
using AB-nonempty-iff by blast
hence a \leq b

51
using 2 atom-regular pseudo-complement by auto
thus atom a
using 2 by auto
qed
end

3.8 Finitely Many Atoms
class stone-relation-algebra-finiteatoms = stone-relation-algebra +
  assumes finiteatoms: finite { a . atom a }
begin

lemma finite-AB:
  finite (AB x)
  using finite-Collect-conjI finiteatoms by force

lemma nAB-top-finite:
  nAB top \neq \infty
  by (smt (verit, best) finite-AB icard-infinite-conv num-atoms-below-def)

end

3.9 Atomic and Finitely Many Atoms
class stone-relation-algebra-atomic-finiteatoms = stone-relation-algebra-atomic +
  stone-relation-algebra-finiteatoms
begin

lemma finite-ideal-points:
  finite { p . ideal-point p }
proof (cases bot = top)
  case True
  hence \land p . ideal-point p \implies p = bot
    using le-bot top.extremum by blast
  hence \{ p . ideal-point p \} \subseteq \{ bot \}
    by auto
  thus ?thesis
    using finite-subset by auto
next
  case False
  let ?p = \{ p . ideal-point p \}
  show 0; finite ?p
proof (rule finite-image-part-le)
  show \forall x \in ?p . AB x \subseteq AB top
    using top.extremum by auto
  have \forall x \in ?p . x \neq bot
    using False by auto
  thus \forall x \in ?p . AB x \neq \{}
    using AB-nonempty by auto

52
∀ x ∈ ?p, ∀ y ∈ ?p . x ≠ y → AB x ∩ AB y = {}

proof (intro ballI, rule impl, rule ccontr)
  fix x y
  assume x ∈ ?p y ∈ ?p x ≠ y
  hence 1: x ∩ y = bot
    by (simp add: different-ideal-points-disjoint)
  assume AB x ∩ AB y ≠ {}
  from this obtain a where atom a ∧ a ≤ x ∧ a ≤ y
    by auto
  thus False
    using 1 by (metis comp-inf semiring.mult-zero-left inf.absorb2
     inf.sup-monoid.add-assoc)
  qed
  show finite (AB top)
    using finite-AB by blast
  qed

qed

end

3.10 Atom-rectangular and Finitely Many Atoms

class stone-relation-algebra-atomrect-finiteatoms =
stone-relation-algebra-atomrect + stone-relation-algebra-finiteatoms

3.11 Atomic, Atom-rectangular and Finitely Many Atoms

class stone-relation-algebra-atomic-atomrect-finiteatoms =
stone-relation-algebra-atomic + stone-relation-algebra-atomrect +
stone-relation-algebra-finiteatoms

begin

subclass stone-relation-algebra-atomic-atomrect ..
subclass stone-relation-algebra-atomic-finiteatoms..
subclass stone-relation-algebra-atomrect-finiteatoms ..

lemma counterexample-nAB-atom-iff:
  atom x ←→ nAB x = 1
  nitpick[expect=genuine, card=3]
  oops

lemma counterexample-nAB-top-iff-eq:
  nAB x = nAB top ←→ x = top
  nitpick[expect=genuine, card=3]
  oops

lemma counterexample-nAB-top-iff-leq:
  nAB top ≤ nAB x ←→ x = top
  nitpick[expect=genuine, card=3]
  oops

53
3.12 Atom-simple and Finitely Many Atoms

class stone-relation-algebra-atomsimple-finiteatoms =
stone-relation-algebra-atomsimple + stone-relation-algebra-finiteatoms

3.13 Atomic, Atom-simple and Finitely Many Atoms

class stone-relation-algebra-atomic-atomsimple-finiteatoms =
stone-relation-algebra-atomic + stone-relation-algebra-atomsimple +
stone-relation-algebra-finiteatoms
begin

subclass stone-relation-algebra-atomic-atomsimple ..
subclass stone-relation-algebra-atomic-finiteatoms ..
subclass stone-relation-algebra-atomsimple-finiteatoms ..

lemma nAB-top-2:
  nAB 1 * nAB 1 ≤ nAB top
proof (unfold num-atoms-below-def icard-cartesian-product[THEN sym], rule
surj-icard-le)
  show AB 1 × AB 1 ⊆ dom-cod ' AB top
    using dom-cod-atoms-2 by blast
qed

lemma counterexample-nAB-atom-iff-2:
  atom x ←→ nAB x = 1
nitpick[expect=genuine,card=6]
oops

lemma counterexample-nAB-top-iff-eq-2:
  nAB x = nAB top ←→ x = top
nitpick[expect=genuine,card=6]
oops

lemma counterexample-nAB-top-iff-leq-2:
  nAB top ≤ nAB x ←→ x = top
nitpick[expect=genuine,card=6]
oops

end

3.14 Atom-rectangular, Atom-simple and Finitely Many Atoms

class stone-relation-algebra-atomrect-atomsimple-finiteatoms =
stone-relation-algebra-atomrect + stone-relation-algebra-atomsimple +
stone-relation-algebra-finiteatoms
subclass stone-relation-algebra-atomrect-atomsimple ..
subclass stone-relation-algebra-atomrect-finiteatoms ..
subclass stone-relation-algebra-atomsimple-finiteatoms ..

end

3.15 Atomic, Atom-rectangular, Atom-simple and Finitely Many Atoms

class stone-relation-algebra-atomic-atomrect-atomsimple-finiteatoms =
stone-relation-algebra-atomic + stone-relation-algebra-atomrect +
stone-relation-algebra-atomsimple + stone-relation-algebra-finiteatoms
begin
subclass stone-relation-algebra-atomic-atomrect-atomsimple ..
subclass stone-relation-algebra-atomic-atomrect-finiteatoms ..
subclass stone-relation-algebra-atomic-atomsimple-finiteatoms ..
subclass stone-relation-algebra-atomrect-atomsimple-finiteatoms ..

lemma all-regular:
regular x
proof (cases x = bot)
case True
thus ?thesis
by simp
next
case False
hence 1: AB x ≠ {}
using AB-nonempty by blast
have 2: finite (AB x)
using finite-AB by blast
have 3: regular (Sup-fin (AB x))
proof −
have −−Sup-fin (AB x) ≤ Sup-fin (AB x)
proof (rule finite-ne-subset-induct'
show finite (AB x)
using 2 by simp
show AB x ≠ {}
using 1 by simp
show AB x ⊆ AB top
by auto
show ∃a . a ∈ AB top ⇒ −−Sup-fin {a} ≤ Sup-fin {a}
using atom-regular by auto
show ∃a F . finite F ⇒ F ≠ {} ⇒ F ⊆ AB top ⇒ a ∈ AB top ⇒ a ∉ F
⇒ −−Sup-fin F ≤ Sup-fin F ⇒ −−Sup-fin (insert a F) ≤ Sup-fin (insert a F)
proof −
\[\text{fix } F\]
\[
\text{assume } 4: \text{finite } F \neq \{\} \subseteq AB \text{ top } a \in AB \text{ top } a \notin F \quad \text{--Sup-fin } F \leq \text{Sup-fin } F
\]
\[
\text{hence } \text{--Sup-fin (insert } a F) = a \sqcup \text{--Sup-fin } F
\]
\[
\text{using } 4 \text{ atom-regular by auto}
\]
\[
\text{also have } ... \leq a \sqcup \text{Sup-fin } F
\]
\[
\text{using } 4 \text{ sup-mono by fastforce}
\]
\[
\text{also have } ... = \text{Sup-fin (insert } a F)
\]
\[
\text{using } 4 \text{ by auto}
\]
\[
\text{finally show } \text{--Sup-fin (insert } a F) \leq \text{Sup-fin (insert } a F)
\]
\[
\text{.}
\]
\[
\text{qed}
\]
\[
\text{qed}
\]
\[
\text{thus } ?\text{thesis}
\]
\[
\text{using } \text{inf.antisym-conv pp-increasing by blast}
\]
\[
\text{qed}
\]
\[
\text{have } x \sqcap -\text{Sup-fin (AB } x) = \text{bot}
\]
\[
\text{proof (rule ccontr)}
\]
\[
\text{assume } x \sqcap -\text{Sup-fin (AB } x) \neq \text{bot}
\]
\[
\text{from this obtain } b \text{ where } 5: \text{atom } b \land b \leq x \sqcap -\text{Sup-fin (AB } x)
\]
\[
\text{using atomic by blast}
\]
\[
\text{hence } b \leq \text{Sup-fin (AB } x)
\]
\[
\text{using Sup-fin.coboundedI 2 by force}
\]
\[
\text{thus False}
\]
\[
\text{using } 5 \text{ atom-in-p-xor by auto}
\]
\[
\text{qed}
\]
\[
\text{hence } 6: x \leq \text{Sup-fin (AB } x)
\]
\[
\text{using } 3 \text{ by (simp add: pseudo-complement)}
\]
\[
\text{have } \text{Sup-fin (AB } x) \leq x
\]
\[
\text{using } 1 2 \text{ Sup-fin.boundedI by fastforce}
\]
\[
\text{thus } ?\text{thesis}
\]
\[
\text{using } 3 6 \text{ order.antisym by force}
\]
\[
\text{qed}
\]
\[
\text{sublocale } ra: \text{relation-algebra where } \text{minus } = \lambda x y . x \sqcap -y
\]
\[
\text{proof}
\]
\[
\text{show } \bigwedge x . x \sqcap -x = \text{bot}
\]
\[
\text{by simp}
\]
\[
\text{show } \bigwedge x . x \sqcup -x = \text{top}
\]
\[
\text{using all-regular pp-sup-p by fast}
\]
\[
\text{show } \bigwedge x y . x \sqcap -y = x \sqcap -y
\]
\[
\text{by simp}
\]
\[
\text{qed}
\]
\[
\text{end}
\]
\[
\text{class } \text{stone-relation-algebra-finite } = \text{stone-relation-algebra } + \text{ finite}
\]
\[
\text{begin}
\]
\[
\text{56}
\]
subclass stone-relation-algebra-atomic-finiteatoms
proof
  show finite { a . atom a } 
    by simp
  show \( \forall x. x \neq \text{bot} \longrightarrow (\exists a. \text{atom } a \land a \leq x) \)
    proof
      fix x
      assume 1: \( x \neq \text{bot} \)
      let \( ?s = \{ y . y \leq x \land y \neq \text{bot} \} \)
      have 2: finite ?s 
        by auto
      have 3: ?s \( \neq \{\} \)
        using 1 by blast
      from ne-finite-has-minimal obtain \( m \) where \( m \in \?) \land (\forall x \in \? . x \leq m \longrightarrow x = m) \)
        using 2 3 by meson
      hence \( \text{atom } m \land m \leq x \)
        using order-trans by blast
      thus \( \exists a. \text{atom } a \land a \leq x \)
        by auto
    qed
qed
end

3.16 Relation Algebra and Atomic

class relation-algebra-atomic = relation-algebra + stone-relation-algebra-atomic
begin

lemma nAB-atom-iff:
  \( \text{atom } a \quad \longleftrightarrow \quad nAB a = 1 \)
proof
  assume \( \text{atom } a \)
  thus \( nAB a = 1 \)
    by (simp add: nAB-atom)
next
  assume \( nAB a = 1 \)
  from this obtain \( b \) where 1: \( AB a = \{ b \} \)
    using icard-1-imp-singleton num-atoms-below-def one-eSuc by fastforce
  hence 2: \( \text{atom } b \land b \leq a \)
    by auto
  hence 3: \( AB (a \cap b) = \{ b \} \)
    by fastforce
  have \( AB (a \cap b) \cup AB (a \cap \neg b) = AB a \land AB (a \cap b) \cap AB (a \cap \neg b) = \{\} \)
    using AB-split-2 AB-split-2-disjoint by simp
  hence \( \{ b \} \cup AB (a \cap \neg b) = \{ b \} \land \{ b \} \cap AB (a \cap \neg b) = \{\} \)
    using 1 3 by simp
  hence \( AB (a \cap \neg b) = \{\} \)

by auto

hence \( a \cap -b = \bot \)

using \( AB\text{-nonempty-iff} \) by blast

hence \( a \leq b \)

by (\text{simp add: shunting-1})

thus atom a

using 2 by auto

qed

end

3.17 Relation Algebra, Atomic and Finitely Many Atoms

class relation-algebra-atomic-finiteatoms = relation-algebra-atomic +
    stone-relation-algebra-atomic-finiteatoms

begin

  \textit{Sup-fin} only works for non-empty finite sets.

lemma atomistic:
  assumes \( x \neq \bot \)
  shows \( x = \text{Sup-fin} (AB \ x) \)

proof (rule order;antisym)
  show \( x \leq \text{Sup-fin} (AB \ x) \)
  proof (rule ccontr)
    assume \( \neg x \leq \text{Sup-fin} (AB \ x) \)
    hence \( x \cap -\text{Sup-fin} (AB \ x) \neq \bot \)
      using shunting-1 by blast
    from this obtain \( a \) where 1: atom a \( \land a \leq x \cap -\text{Sup-fin} (AB \ x) \)
      using atomic by blast
    hence \( a \in AB \ x \)
      by simp
    hence \( a \leq \text{Sup-fin} (AB \ x) \)
      using Sup-fin.coboundedI finite-AB by auto
    thus False
      using 1 atom-in-p-xor by auto
  qed

  show \( \text{Sup-fin} (AB \ x) \leq x \)
  proof (rule Sup-fin.boundedI)
    show finite (\( AB \ x \))
      using finite-AB by auto
    show \( AB \ x \neq \{\} \)
      using \( \text{assms atomic} \) by blast
    show \( \bigwedge a. \ a \in AB \ x \Rightarrow a \leq x \)
      by auto
  qed

  qed

lemma counterexample-nAB-top:
  \( 1 \neq \top \Rightarrow nAB \top = nAB \ 1 * nAB \ 1 \)

nitpick[expect=genuine,card=4]
class relation-algebra-atomic-atomsimple-finiteatoms =
relation-algebra-atomic-finiteatoms +
stone-relation-algebra-atomic-atomsimple-finiteatoms
begin

lemma counterexample-atom-rectangle:
atom x → rectangle x
nitpick[expect=genuine,card=4]
oops

lemma counterexample-atom-univalent:
atom x → univalent x
nitpick[expect=genuine,card=4]
oops

lemma counterexample-point-dense:
assumes x ≠ bot
and x ≤ 1
shows ∃ a . a ≠ bot ∧ a * top * a ≤ 1 ∧ a ≤ x
nitpick[expect=genuine,card=4]
oops

end

class relation-algebra-atomic-atomrect-atomsimple-finiteatoms =
relation-algebra-atomic-atomsimple-finiteatoms +
stone-relation-algebra-atomic-atomrect-atomsimple-finiteatoms

4 Cardinality in Stone Relation Algebras

We study various axioms for a cardinality operation in Stone relation algebras.

class card =
  fixes cardinality :: ′a ⇒ enat (#· [100] 100)

class sra-card = stone-relation-algebra + card
begin

abbreviation card-bot :: ′a ⇒ bool where card-bot = 0
abbreviation card-bot-iff :: ′a ⇒ bool where card-bot-iff · x:′a . #x = 0 ↔ x = bot
abbreviation card-top :: ′a ⇒ bool where card-top = #top = #1 * #1
abbreviation card-conv :: 'a ⇒ bool where card-conv - ≡
∀ x::'a . #(x⇧T) = #x
abbreviation card-add :: 'a ⇒ bool where card-add - ≡ ∀ x
  y::'a . #x + #y = #(x ⊔ y) + #(x ∩ y)
abbreviation card-iso :: 'a ⇒ bool where card-iso - ≡ ∀ x
  y::'a . x ≤ y → #x ≤ #y
abbreviation card-univ-comp-meet :: 'a ⇒ bool where card-univ-comp-meet -
  ≡ ∀ x y z::'a . univalent x → #(x ⊔ y) ≤ #(x * z ∩ y)
abbreviation card-univ-comp-comp :: 'a ⇒ bool where card-univ-comp-comp -
  ≡ ∀ x y z::'a . univalent x → #(x ⊔ y) ≤ #(x * z ∩ y)
abbreviation card-comp-univ :: 'a ⇒ bool where card-comp-univ -
  ≡ ∀ x y::'a . univalent x → #(y * x) ≤ #y
abbreviation card-univ-meet-vector :: 'a ⇒ bool where card-univ-meet-vector -
  ≡ ∀ x y::'a . univalent x → #(x ∩ y * top) ≤ #y
abbreviation card-univ-meet-conv :: 'a ⇒ bool where card-univ-meet-conv -
  ≡ ∀ x y::'a . univalent x → #(x ∩ y * y⇧T) ≤ #y
abbreviation card-domain-sym :: 'a ⇒ bool where card-domain-sym -
  ≡ ∀ x::'a . #(1 ∩ x * x⇧T) ≤ #x
abbreviation card-domain-sym-conv :: 'a ⇒ bool where card-domain-sym-conv -
  ≡ ∀ x::'a . #(1 ∩ x⇧T * x) ≤ #x
abbreviation card-domain :: 'a ⇒ bool where card-domain -
  ≡ ∀ x::'a . #(1 ∩ x * top) ≤ #x
abbreviation card-domain-comp :: 'a ⇒ bool where card-domain-comp -
  ≡ ∀ x::'a . #(1 ∩ x⇧T * top) ≤ #x
abbreviation card-codomain :: 'a ⇒ bool where card-codomain -
  ≡ ∀ x::'a . #(1 ∩ top * x) ≤ #x
abbreviation card-codomain-comp :: 'a ⇒ bool where card-codomain-comp -
  ≡ ∀ x::'a . #(1 ∩ top * x⇧T) ≤ #x
abbreviation card-comp :: 'a ⇒ bool where card-comp -
  ≡ ∀ x::'a . #(1 ∩ x) ≤ #x
abbreviation card-comp-iff :: 'a ⇒ bool where card-comp-iff -
  ≡ ∀ x::'a . #x = #top → x = top
abbreviation card-comp-iff-eq :: 'a ⇒ bool where card-comp-iff-eq -
  ≡ ∀ x::'a . #x = #top ←→ x = top
abbreviation card-top-finite :: 'a ⇒ bool where card-top-finite -
  ≡ #top ≠ ∞

lemma card-domain-iff:
  card-domain - ↔ card-domain-sym -
  by (simp add: domain-vector-cone)

lemma card-codomain-comp-iff:
  card-codomain-comp - ↔ card-domain -
  by (simp add: domain-vector-covector)
lemma card-codomain-iff:
  assumes card-conv: card-conv -
  shows card-codomain - ↔ card-codomain-conv -
  by (metis card-conv conv-involutive)

lemma card-domain-cone-iff:
  card-codomain - ↔ card-domain-cone -
  using domain-vector-covector by auto

lemma card-domain-sym-cone-iff:
  card-domain-cone - ↔ card-domain-sym-cone -
  by (simp add: domain-vector-conv)

lemma card-bot:
  assumes card-bot-iff: card-bot-iff -
  shows card-bot -
  using card-bot-iff by auto

lemma card-comp-univ-implies-card-univ-comp-meet:
  assumes card-conv: card-conv -
  and card-comp-univ: card-comp-univ -
  shows card-univ-comp-meet -
proof (intro allI, rule impI)
  fix x y z
  assume 1: univalent x
  have #((x^T ⊗ y ⊓ z) = #(y^T ⊗ x ⊓ z^T)
    by (metis card-conv conv-dist-comp conv-dist-inf conv-involutive)
  also have ... = #(y^T ⊓ z^T ⊗ x^T) * x)
    using 1 by (simp add: dedekind-univalent)
  also have ... ≤ #(y^T ⊓ z^T ⊗ x^T)
    using 1 card-comp-univ by blast
  also have ... = #((x ⊗ z ⊓ y)
    by (metis card-conv conv-dist-comp conv-dist-inf inf.sup-monoid.add-commute)
  finally show #((x^T ⊗ y ⊓ z) ≤ #((x ⊗ z ⊓ y)
  qed

lemma card-univ-meet-conv-implies-card-domain-sym:
  assumes card-univ-meet-conv: card-univ-meet-conv -
  shows card-domain-sym -
  by (simp add: card-univ-meet-conv)

lemma card-add-disjoint:
  assumes card-bot: card-bot -
  and card-add: card-add -
  and x ⊓ y = bot
  shows #((x ⊔ y) = #x + #y
  by (simp add: assms(3) card-add card-bot)
lemma card-dist-sup-disjoint:
assumes card-bot: card-bot -
and card-add: card-add -
and A ≠ { }
and finite A
and ∀ x∈A . ∀ y∈A . x ≠ y → x ∩ y = bot
shows #Sup-fin A = sum cardinality A

proof (rule finite-ne-subset-induct)
show finite A
  using assms(4) by simp
show A ≠ { } using assms(3) by simp
show A ⊆ A by simp
show \( \forall x . x \in A \rightarrow \#\text{Sup-fin } \{ x \} = \text{sum cardinality } \{ x \} \) by auto
fix x F
assume 1: finite F F ≠ { } F ⊆ A x ∈ A x /∈ F #Sup-fin F = sum cardinality F
have #Sup-fin (insert x F) = # (x ⊔ Sup-fin F) using 1 by simp
also have ...
proof
  have x ⊔ Sup-fin F = Sup-fin \{ x \cap y | y \in F \} 
    using inf-Sup1-distrib by simp
  also have ...
    using 1 assms(5) by (metis (mono_tags, opaque-lifting) subset_iff)
  also have ...
    by (rule Sup-fin.boundedI, simp-all add: 1)
  finally have x ∩ Sup-fin F = bot
    by (simp add: order.antisym)
  thus ??thesis
    using card-add-disjoint assms by auto
qed

also have ...
proof
  have \( \forall x \in A . \forall y \in A . x \neq y \rightarrow x \cap y = bot \)
    using different-atoms-disjoint assms(5) by auto

qesh

lemma card-dist-sup-atoms:
assumes card-bot: card-bot -
and card-add: card-add -
and A ≠ { }
and finite A
and A ⊆ AB top
shows #Sup-fin A = sum cardinality A

proof –
have \( \forall x \in A . \forall y \in A . x \neq y \rightarrow x \cap y = bot \)
  using different-atoms-disjoint assms(5) by auto
thus ?thesis
  using card-dist-sup-disjoint assms(1–4) by auto
qed

lemma card-univ-meet-comp-implies-card-domain-sym:
  assumes card-univ-meet-comp: card-univ-meet-comp -
  shows card-domain-sym -
  by (metis card-univ-meet-comp inf.idem mult-1-left univalent-one-closed)

lemma card-top-greatest:
  assumes card-iso: card-iso -
  shows x ≤ #top
  by (simp add: card-iso)

lemma card-pp-increasing:
  assumes card-iso: card-iso -
  shows x ≤ #(−−x)
  by (simp add: card-iso pp-increasing)

lemma card-top-iff-eq-leq:
  assumes card-iso: card-iso -
  shows card-top-iff-eq - ←→ card-top-iff-leq -
  using card-iso card-top-greatest nle-le by blast

lemma card-univ-comp-meet-implies-card-comp-univ:
  assumes card-iso: card-iso -
  and card-conv: card-conv -
  and card-univ-comp-meet: card-univ-comp-meet -
  shows card-comp-univ -
proof (intro allI, rule impI)
  fix x y
  assume 1: univalent x
  have #(y * x) = #(xT * yT)
    by (metis card-conv conv-dist-comp)
  also have ... = #(top ∩ xT * yT)
    by simp
  also have ... ≤ #(x * top ∩ yT)
    using 1 by (metis card-univ-comp-meet inf.sup-monoid.add-commute)
  also have ... ≤ #(yT)
    using card-iso by simp
  also have ... = #y
    by (simp add: card-conv)
  finally show #(y * x) ≤ #y
.  
qed

lemma card-comp-univ-iff-card-univ-comp-meet:
  assumes card-iso: card-iso -
  and card-conv: card-conv -

63
shows \( \text{card-comp-univ} \leftrightarrow \text{card-univ-comp-meet} \)
using \( \text{card-iso} \), \( \text{card-univ-comp-meet-implies-card-comp-univ} \), \( \text{card-conv} \)
card-comp-univ-implies-card-univ-meet by blast

lemma card-univ-meet-vector-implies-card-univ-meet-comp:
assumes card-iso: card-iso
and card-univ-meet-vector: card-univ-meet-vector
shows card-univ-meet-comp
proof (intro allI, rule impI)
fix \( x, y, z \)
assume 1: univalent \( x \)
have \(#(x \land y \ast z^T) = #(x \land (y \land x \ast z) \ast (z^T \land y^T \ast x))\)
by (metis conv-involutive dedekind-eq inf.sup-monoid.add-commute)
also have \(... \leq #(x \land (y \land x \ast z) \ast \text{top})\)
using card-iso inf.sup-right-isotone mult-isotone by auto
also have \(... \leq #(x \ast z \land y)\)
using 1 by (simp add: card-univ-meet-vector inf.sup-monoid.add-commute)
finally show \(#(x \land y \ast z^T) \leq #(x \ast z \land y)\)
qed

lemma card-univ-meet-comp-implies-card-univ-meet-vector:
assumes card-iso: card-iso
and card-univ-meet-comp: card-univ-meet-comp
shows card-univ-meet-vector
proof (intro allI, rule impI)
fix \( x, y, z \)
assume 1: univalent \( x \)
have \(#(x \land y \ast \text{top}) \leq #(x \ast \text{top} \land y)\)
using 1 by (metis card-univ-meet-comp symmetric-top-closed)
also have \(... \leq \#y\)
using card-iso by auto
finally show \(#(x \land y \ast \text{top}) \leq \#y\)
qed

lemma card-univ-meet-vector-iff-card-univ-meet-comp:
assumes card-iso: card-iso
and card-univ-meet-vector:
shows card-univ-meet-comp
using card-iso card-univ-meet-comp-implies-card-univ-meet-vector
card-univ-meet-vector-implies-card-univ-meet-comp by blast

lemma card-univ-meet-vector-implies-card-univ-meet-conv:
assumes card-iso: card-iso
and card-univ-meet-vector:
shows card-univ-meet-conv
proof (intro allI, rule impI)
fix \( x, y, z \)
assume 1: univalent \( x \)
have \(\#(x \cap y \ast y^T) \leq \#(x \cap y \ast \text{top})\)
using card-iso comp-inf.mult-right-isotone mult-right-isotone by auto
also have \(\ldots \leq \#y\)
using 1 by (simp add: card-univ-meet-vector)
finally show \(\#(x \cap y \ast y^T) \leq \#y\)
.

qed

lemma card-domain-sym-implies-card-univ-meet-vector:
assumes card-comp-univ: card-comp-univ -
and card-domain-sym: card-domain-sym -
shows card-univ-meet-vector -
proof (intro allI, rule impI)
fix x y z
assume 1: univalent x
have \(\#((x \ast \text{top}) \cap I) \ast (x \cap y \ast \text{top}))\)
by (simp add: inf.absorb2 vector-export-comp-unit)
also have \(\ldots \leq \#(y \ast \text{top} \cap I)\)
using 1 by (simp add: card-comp-univ univalent-inf-closed)
also have \(\ldots \leq \#y\)
using card-domain-sym card-domain-iff inf.sup-monoid.add-commute by auto
finally show \(\#(x \ast \text{top}) \leq \#y\)
.

qed

lemma card-domain-sym-iff-card-univ-meet-vector:
assumes card-iso: card-iso -
and card-comp-univ: card-comp-univ -
shows card-domain-sym - \(\longleftrightarrow\) card-univ-meet-vector -

lemma card-univ-meet-conv-iff-card-univ-meet-comp:
assumes card-iso: card-iso -
and card-comp-univ: card-comp-univ -
shows card-univ-meet-comp - \(\longleftrightarrow\) card-univ-meet-vector -

lemma card-domain-sym-iff-card-univ-meet-comp:
assumes card-iso: card-iso -
and card-comp-univ: card-comp-univ -
shows card-domain-sym - \(\longleftrightarrow\) card-univ-meet-comp -

lemma card-univ-meet-conv-iff-card-univ-meet-comp:
assumes card-iso: card-iso -
and card-comp-univ: card-comp-univ -
shows card-domain-sym - \(\longleftrightarrow\) card-univ-meet-comp -

lemma card-domain-sym-iff-card-univ-meet-comp:
assumes card-iso: card-iso -
and card-comp-univ: card-comp-univ -
shows card-domain-sym - \(\longleftrightarrow\) card-univ-meet-comp -
lemma card-univ-comp-mapping:
  assumes card-comp-univ: card-comp-univ -
    and card-univ-meet-comp: card-univ-meet-comp -
    and univalent x
    and mapping y
  shows #(x * y) = #x
proof -
  have #x = #(x ∩ top * g^T)
    using assms(4) total-conv-surjective by auto
  also have ... ≤ #(x * y ∩ top)
    using assms(3) card-univ-meet-comp by blast
  finally have #x ≤ #(x * y)
    by simp
  thus ?thesis
    using assms(4) card-comp-univ nle-le by blast
qed

lemma card-point-one:
  assumes card-comp-univ: card-comp-univ -
    and card-univ-meet-comp: card-univ-meet-comp -
    and card-conv: card-conv -
    and point x
  shows #x = #1
proof -
  have mapping (x^T)
    using assms(4) surjective-conv-total by auto
  thus ?thesis
    by (smt card-univ-comp-mapping card-comp-univ card-conv
    card-univ-meet-comp coreflexive-comp-top-inf inf.absorb2 reflexive-one-closed
    top-right-mult-increasing total-one-closed univalent-one-closed)
qed

end

4.1 Cardinality in Relation Algebras

class ra-card = sra-card + relation-algebra
begin

lemma card-iso:
  assumes card-bot: card-bot -
    and card-add: card-add -
  shows card-iso -
proof (intro allI, rule impI)
  fix x y
  assume x ≤ y
  hence #y = #(x ⊔ (−x ∩ y))
    by (simp add: sup-absorb2)
also have \ldots = \#(x \sqcup \neg x \cap y) + \#(x \cap \neg x \cap y)
  by (simp add: card-bot)
also have \ldots = \#x + \#(\neg x \cap y)
  by (metis card-add)
finally show \#x \leq \#y
  using le-iff-add by blast
qed

lemma card-top-iff-eq:
  assumes card-bot-iff: card-bot-iff -
    and card-add: card-add -
    and card-top-finite: card-top-finite -
  shows card-top-iff-eq -
proof (rule allI, rule iffI)
  fix x
  assume 1: \#x = \#top
  have \#top = \#(x \sqcup \neg x)
    by simp
  also have \ldots = \#x + \#(\neg x)
    using card-add card-bot-iff card-add-disjoint inf-p by blast
  also have \ldots = \#top + \#(\neg x)
    using 1 by simp
finally have \#(\neg x) = 0
  by (simp add: card-top-finite)
hence \neg x = bot
  using card-bot-iff by blast
thus x = top
  using comp-inf.pp-total by auto
next
  fix x
  assume x = top
  thus \#x = \#top
    by simp
qed

class ra-card-atomic-finiteatoms = ra-card + relation-algebra-atomic-finiteatoms
begin

lemma card-nAB:
  assumes card-bot: card-bot -
    and card-add: card-add -
    and card-atom: card-atom -
  shows \#x = nAB x
proof (cases x = bot)
  case True
  thus ?thesis
    by (simp add: card-bot nAB-bot)
next
  case False
  have 1: finite $(AB \ x)$
    using finite-AB by blast
  have 2: $AB \ x \not= \{\}$
    using False AB-nonempty-iff by blast
  have $\#x = \#\text{Sup-fin} \ (AB \ x)$
    using atomistic False by auto
  also have $\ldots = \text{sum cardinality} \ (AB \ x)$
    using 1 2 card-bot card-add card-dist-sup-disjoint different-atoms-disjoint by force
  also have $\ldots = \text{sum} \ (\lambda x \ . \ 1) \ (AB \ x)$
    using card-atom by simp
  also have $\ldots = \text{i.card} \ (AB \ x)$
    by (metis (mono-tags, lifting) icard-eq-sum finite-AB)
  also have $\ldots = n\text{AB} \ x$
    by (simp add: num-atoms-below-def)
  finally show $\ldots$.

qed

end

class card-ab = sra-card +
  assumes card-nAB': $\#x = n\text{AB} \ x$

class sra-card-ab-atomsimple-finiteatoms = sra-card + card-ab +
  stone-relation-algebra-atomsimple-finiteatoms +
  assumes card-bot-iff: card-bot-iff -
  assumes card-top: card-top -
begin

subclass stone-relation-algebra-atomic-atomsimple-finiteatoms
proof
  show $\bigwedge x . x \not= \text{bot} \longrightarrow (\exists a . \text{atom} a \land a \leq x)$
  proof
    fix $x$
    assume $x \not= \text{bot}$
    hence $\#x \not= 0$
      using card-bot-iff by auto
    hence $n\text{AB} \ x \not= 0$
      by (simp add: card-nAB')
    hence $AB \ x \not= \{\}$
      by (metis (mono-tags, lifting) icard-empty num-atoms-below-def)
    thus $\exists a . \text{atom} a \land a \leq x$
      by auto
  qed
qed
lemma dom-cod-inj-atoms:
inj-on dom-cod (AB top)
proof (rule eq-card-imp-inj-on)
  show 1: finite (AB top)
    using finite-AB by blast
  have icard (dom-cod ' AB top) = icard (AB 1 × AB 1)
    using dom-cod-atoms by auto
  also have ... = icard (AB 1) * icard (AB 1)
    using icard-cartesian-product by blast
  also have ... = #1 * #1
    by (simp add: card-nAB' num-atoms-below-def)
  also have ... = #top
    by (simp add: card-top)
  also have ... = icard (AB top)
    by (simp add: card-nAB' num-atoms-below-def)
  finally have icard (dom-cod ' AB top) = icard (AB top)
  thus card (dom-cod ' AB top) = card (AB top)
    using 1 by (smt (z3) finite-icard-card)
qed

subclass stone-relation-algebra-atomic-atomrect-atomsimple-finiteatoms
proof
  have ∨ a . atom a ∧ a ≤ 1 −→ a * top * a ≤ 1
    proof
      fix a
      let ?ca = top * a ∩ 1
      assume 1: atom a ∧ a ≤ 1
      have aT * top * a ≤ 1
        proof (rule ccontr)
          assume ¬ aT * top * a ≤ 1
          hence aT * top * a ∩ −1 ≠ bot
            by (simp add: pseudo-complement)
          from this obtain b where 2: atom b ∧ b ≤ aT * top * a ∩ −1
            using atomic by blast
          hence b * top ≤ aT * top
            by (metis comp-associative dual-order.trans inf.boundedE mult-left-isotone
                mult-right-isotone top.extremum)
          hence b * top ∩ 1 ≤ ?ca
            by (metis comp-inf.comp-isotone conv-dist-comp conv-dist-inf
                coreflexive-symmetric inf.cobounded2 reflexive-one-closed symmetric-top-closed)
          hence 3: b * top ∩ 1 = ?ca
            using 1 2 domain-atom codomain-atom by simp
          hence top * b ≤ top * a
            using 2 by (metis comp-associative comp-inf.vertex-top-closed
                comp-inf-vertex inf.boundedE mult-right-isotone vector-export-comp-unit
                vector-top-closed)
          hence top * b ∩ 1 ≤ ?ca
            using inf-mono by blast
hence \( \top \ast b \sqcap I = ?ca \)
  using 1 2 codomain-atom by simp
hence 4: \( \text{dom-cod} \ b = \text{dom-cod} \ ?ca \)
  using 3 by (metis comp-inf-covector comp-right-one
  inf.sup-monoid.add-commute inf-top.left-neutral vector-export-comp-unit)
have \( b \in AB \top \land ?ca \in AB \top \)
  using 1 2 codomain-atom by simp
hence \( b = ?ca \)
  using inj-onD dom-cod-inj-atoms 2 4 by smt
thus False
  using 2 by (metis comp-inf
  .mult-right-isotone inf
  .boundedE inf
  .idem inf
  .left-commute inf-p le-bot)
qed

thus \( \ast \top \ast \ast \leq 1 \)
  using 1 by (simp add: coreflexive-symmetric)
qed

lemma atom-rectangle-card:
assumes atom a
shows \( \#(a \ast \top \ast a) = 1 \)
by (simp add: assms atomrect-eq card-nAB' nAB-atom)

lemma atom-regular-rectangle:
assumes atom a
shows \( \lnot a = a \ast \top \ast a \)
proof (rule order.antisym)
show \( \lnot a \leq a \ast \top \ast a \)
  using assms atom-rectangle-regular ex231d pp-dist-comp by auto
show \( a \ast \top \ast a \leq \lnot a \)
proof (rule ccontr)
  assume \( \neg (a \ast \top \ast a) \leq \lnot a \)
  hence \( a \ast \top \ast a \sqcap \lnot a \neq \bot \)
    by (simp add: pseudo-complement)
from this obtain \( b \) where 1: \( a \ast \top \ast a \sqcap b \leq a \ast \top \ast a \sqcap \lnot a \)
  using atomic by blast
hence 2: \( b \neq a \)
  using inf.absorb2 by fastforce
have 3: \( a \in AB \ast \top \ast a \sqcap b \in AB \ast \top \ast a \)
  using 1 assms ex231d by auto
from atom-rectangle-card obtain \( c \) where \( AB \ast \top \ast a \equiv \{ c \} \)
  using card-nAB' num-atoms-below-def assms icard-1-imp-singleton one-eSuc
by fastforce
thus False
  using 2 3 by auto
qed
sublocale ra-atom: relation-algebra-atomic where

    minus = \lambda x y . x \sqcap y ..

end

class ra-card-atomic-atomsimple-finiteatoms = ra-card +
    relation-algebra-atomic-atomsimple-finiteatoms +
    assumes card-bot: card-bot -
    assumes card-add: card-add -
    assumes card-atom: card-atom -
    assumes card-top: card-top -
begin

subclass ra-card-atomic-finiteatoms ..

subclass sra-card-ab-atomsimple-finiteatoms
    apply unfold-locales
    using card-add card-atom card-bot card-nAB apply blast
    using card-add card-atom card-bot card-nAB nAB-bot-iff apply presburger
    using card-top by auto

subclass relation-algebra-atomic-atomrect-atomsimple-finiteatoms ..
end

4.2 Counterexamples

class ra-card-notop = ra-card +
    assumes card-bot-iff: card-bot-iff -
    assumes card-conv: card-conv -
    assumes card-add: card-add -
    assumes card-atom-iff: card-atom-iff -
    assumes card-univ-comp-meet: card-univ-comp-meet -
    assumes card-univ-meet-comp: card-univ-meet-comp -

class ra-card-all = ra-card-notop +
    assumes card-top: card-top -
    assumes card-top-finite: card-top-finite -

class ra-card-notop-atomic-finiteatoms = ra-card-atomic-finiteatoms +
a-card-notop

class ra-card-all-atomic-finiteatoms = ra-card-notop-atomic-finiteatoms +
a-card-all

abbreviation r0000 :: bool \Rightarrow bool \Rightarrow bool where
    r0000 x y \equiv \text{False}
abbreviation r1000 :: bool \Rightarrow bool \Rightarrow bool where
    r1000 x y \equiv \neg x \land \neg y
abbreviation r0001 :: bool ⇒ bool ⇒ bool where r0001 x y ≡ x ∧ y
abbreviation r1001 :: bool ⇒ bool ⇒ bool where r1001 x y ≡ x = y
abbreviation r0110 :: bool ⇒ bool ⇒ bool where r0110 x y ≡ x ≠ y
abbreviation r1111 :: bool ⇒ bool ⇒ bool where r1111 x y ≡ True

lemma r-all-different:
  r0000 ≠ r1000 r0000 ≠ r0001 r0000 ≠ r1001 r0000 ≠ r0110
  r1000 ≠ r1111
  r1000 ≠ r0000 r1000 ≠ r1001 r1000 ≠ r1001
  r0001 ≠ r1111
  r0001 ≠ r0000 r0001 ≠ r1000 r0001 ≠ r0001 ≠ r0110
  r1001 ≠ r1111
  r1001 ≠ r0000 r1001 ≠ r1000 r1001 ≠ r0001
  r0110 ≠ r1111
  r0110 ≠ r0000 r0110 ≠ r1000 r0110 ≠ r0001 r0110 ≠ r1001
  r1111 ≠ r1111
  r1111 ≠ r0000 r1111 ≠ r1000 r1111 ≠ r0001 r1111 ≠ r1001 r1111 ≠ r0110
  by metis+

typedef (overloaded) ra1 = {r0000,r1001,r0110,r1111}
  by auto

typedef (overloaded) ra2 = {r0000,r1000,r0001,r1001}
  by auto

setup-lifting type-definition-ra1
setup-lifting type-definition-ra2
setup-lifting type-definition-prod

instantiation Enum.finite-4 :: ra-card-atomic-finiteatoms
begin

definition one-finite-4 :: Enum.finite-4 where one-finite-4 = finite-4.a1
definition conv-finite-4 :: Enum.finite-4 ⇒ Enum.finite-4 where conv-finite-4 x = x
definition times-finite-4 :: Enum.finite-4 ⇒ Enum.finite-4 ⇒ Enum.finite-4
  where times-finite-4 x y = (case (x,y) of (finite-4.a1,-) ⇒ finite-4.a1 | (-,finite-4.a1) ⇒ finite-4.a1 | (finite-4.a2,y) ⇒ y | (x,finite-4.a2) ⇒ x | - ⇒ finite-4.a4)
definition cardinality-finite-4 :: Enum.finite-4 ⇒ enat
  where cardinality-finite-4 x = (case x of finite-4.a1 ⇒ 0 | finite-4.a4 ⇒ 2 | - ⇒ 1)

instance
  apply intro-classes
  subgoal by (simp add: times-finite-4-def split: finite-4.splits)
  subgoal by (simp add: times-finite-4-def sup-finite-4-def split: finite-4.splits)
  subgoal by (simp add: times-finite-4-def)
  subgoal by (simp add: times-finite-4-def one-finite-4-def split: finite-4.splits)
  subgoal by (simp add: conv-finite-4-def)
subgoal by (simp add: sup-finite-4-def conv-finite-4-def)

subgoal by (simp add: times-finite-4-def conv-finite-4-def split: finite-4.splits)

subgoal by (simp add: times-finite-4-def inf-finite-4-def conv-finite-4-def
less-eq-finite-4-def split: finite-4.splits)

subgoal by (simp add: times-finite-4-def)

subgoal by simp

subgoal by (auto simp add: less-eq-finite-4-def split: finite-4.splits)

subgoal by simp

subgoal by simp

subgoal by simp

subgoal by simp

subgoal by simp

subgoal by simp

subgoal by simp

done

end

instantiation Enum.finite-4 :: ra-card-notop-atomic-finiteatoms
begin

instance

apply intro-classes

subgoal 1

apply (clarsimp simp: cardinality-finite-4-def split: finite-4.splits)

by (metis enat-0 one-neq-zero zero-neq_numeral)

subgoal 2 by (simp add: conv-finite-4-def)

subgoal 3 by (simp add: cardinality-finite-4-def sup-finite-4-def inf-finite-4-def
split: finite-4.splits)

subgoal 4 using 1 3 4 by (auto simp add: cardinality-finite-4-def less-eq-finite-4-def
split: finite-4.splits)

subgoal 5 using 1 3 4 by (metis (no_types, lifting) card-nAB
nAB-univ-comp-meet)

subgoal 6 using 1 3 4 by (metis (no_types, lifting) card-nAB
nAB-univ-meet-comp)

done

end

instantiation ra1 :: ra-card-atomic-finiteatoms
begin

lift-definition bot-ra1 :: ra1 is r0000 by simp

lift-definition one-ra1 :: ra1 is r1001 by simp

lift-definition top-ra1 :: ra1 is r1111 by simp

lift-definition conv-ra1 :: ra1 ⇒ ra1 is id by simp

lift-definition uminus-ra1 :: ra1 ⇒ ra1 is λr x y . ¬ r x y by auto

lift-definition sup-ra1 :: ra1 ⇒ ra1 ⇒ ra1 is λq r x y . q x y ∨ r x y by auto

lift-definition inf-ra1 :: ra1 ⇒ ra1 ⇒ ra1 is λq r x y . q x y ∧ r x y by auto

lift-definition times-ra1 :: ra1 ⇒ ra1 ⇒ ra1 is λq r x y . ∃z . q x z ∧ r z y by

fastforce

lift-definition minus-ra1 :: ra1 ⇒ ra1 ⇒ ra1 is λq r x y . q x y ∧ ¬ r x y by

auto

lift-definition less-eq-ra1 :: ra1 ⇒ ra1 ⇒ bool is λq r . ∀x y . q x y → r x y

. auto

lift-definition less-ra1 :: ra1 ⇒ ra1 ⇒ bool is λq r . (∀ x y . q x y → r x y) ∧
\( q \neq r \).

**lift-definition** cardinality-\( ra1 :: \text{ra1} \Rightarrow \text{enat} \) \( \text{is } \lambda q . \) if \( q = r0000 \) then \( 0 \) else if \( q = r1111 \) then \( 2 \) else \( 1 \).

**instance**

apply intro-classes
subgoal apply transfer by blast
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by meson
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by fastforce
subgoal apply transfer by auto
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by blast
subgoal apply transfer by simp
done

**end**

**lemma** four-cases:

assumes \( P x1 P x2 P x3 P x4 \)

shows \( \forall y \in \{ x . x \in \{ x1, x2, x3, x4 \} \} . P y \)

using \( \text{assms} \) by auto

**lemma** \( r\text{-aux} \):

\( (\lambda y . \ r1001 \ x \ y \lor \ r0110 \ x \ y) = r1111 \) \( (\lambda x . y . \ r1001 \ x \ y \land \ r0110 \ x \ y) = r0000 \)

\( (\lambda x . r0110 \ x \ y \lor \ r1001 \ x \ y) = r1111 \) \( (\lambda x . y . \ r0110 \ x \ y \land \ r1001 \ x \ y) = r0000 \)

\( (\lambda x . r1000 \ x \ y \lor \ r0001 \ x \ y) = r1001 \) \( (\lambda x . y . \ r1000 \ x \ y \land \ r0001 \ x \ y) = r0000 \)

74
\[
(\lambda x y. r_{1000} \land r_{1001} x y) = r_{1001} \quad (\lambda x y. r_{1000} x y \land r_{1001} x y) = r_{1000} \\
(\lambda x y. r_{0001} x y \land r_{1000} x y) = r_{1001} \quad (\lambda x y. r_{0001} x y \land r_{1001} x y) = r_{0000} \\
(\lambda x y. r_{0001} x y \land r_{1001} x y) = r_{1001} \quad (\lambda x y. r_{0001} x y \land r_{1001} x y) = r_{0001} \\
(\lambda x y. r_{1001} x y \land r_{1000} x y) = r_{1001} \quad (\lambda x y. r_{1001} x y \land r_{1000} x y) = r_{1000} \\
(\lambda x y. r_{1001} x y \land r_{0001} x y) = r_{1001} \quad (\lambda x y. r_{1001} x y \land r_{0001} x y) = r_{0001}
\]
subgoal apply transfer by auto
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, metis (full-types))
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, metis (full-types))
subgoal apply transfer by auto

end

instantiation ra2 :: ra-card-notop-atomic-finiteatoms
begin

instance
apply intro-classes
subgoal 1 apply transfer by (metis one-neq-zero zero-neq-numeral)
subgoal 2 apply transfer by simp
subgoal 3 apply transfer
apply (rule four-cases)
subgoal using r-all-different by auto
subgoal apply (rule four-cases) using r-aux r-all-different by auto
subgoal apply (rule four-cases) using r-aux r-all-different by auto
subgoal using r-aux r-all-different by auto
done
subgoal 4 apply transfer using r-all-different zero-one-enat-neq(1) by auto
subgoal 5 using 1 3 4 by (metis (no-types, lifting) card-nAB nAB-univ-comp-meet)
subgoal 6 using 1 3 4 by (metis (no-types, lifting) card-nAB

76
\textbf{nAB-univ-meet-comp) done}

\textbf{end}

\textbf{instantiation prod :: (stone-relation-algebra,stone-relation-algebra)}

\textbf{stone-relation-algebra}

\textbf{begin}

lift-definition \texttt{bot-prod :: 'a × 'b is (bot::'a,bot::'b) .}
lift-definition \texttt{one-prod :: 'a × 'b is (1::'a,1::'b) .}
lift-definition \texttt{top-prod :: 'a × 'b is (top::'a,top::'b) .}
lift-definition \texttt{conv-prod :: 'a × 'b ⇒ 'a × 'b is λ(u,v) . (conv u,conv v) .}
lift-definition \texttt{uminus-prod :: 'a × 'b ⇒ 'a × 'b is λ(u,v) . (uminus u,uminus v) .}
lift-definition \texttt{sup-prod :: 'a × 'b ⇒ 'a × 'b ⇒ 'a × 'b is λ(u,v) (w,x) . (u ⊔ w,v ⊔ x) .}
lift-definition \texttt{inf-prod :: 'a × 'b ⇒ 'a × 'b ⇒ 'a × 'b is λ(u,v) (w,x) . (u ⊓ w,v ⊓ x) .}
lift-definition \texttt{times-prod :: 'a × 'b ⇒ 'a × 'b ⇒ 'a × 'b is λ(u,v) (w,x) . (u * w,v * x) .}
lift-definition \texttt{less-eq-prod :: 'a × 'b ⇒ 'a × 'b ⇒ bool is λ(u,v) (w,x) . u ≤ w ∧ v ≤ x .}
lift-definition \texttt{less-prod :: 'a × 'b ⇒ 'a × 'b ⇒ bool is λ(u,v) (w,x) . u ≤ w ∧ v ≤ x ∧ ¬(u = w ∧ v = x) .}

\textbf{instance}

apply intro-classes
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal by (unfold less-eq-prod-def, clarsimp)
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, simp add: sup-inf-distrib1)
subgoal apply transfer by (clarsimp, simp add: pseudo-complement)
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, simp add: mult.assoc)
subgoal apply transfer by (clarsimp, simp add: mult-right-dist-sup)
subgoal apply transfer by simp
subgoal apply transfer by simp
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, simp add: conv-dist-sup)
subgoal apply transfer by (clarsimp, simp add: conv-dist-comp)
subgoal apply transfer by (clarsimp, simp add: dedekind-1)
subgoal apply transfer by (clarsimp, simp add: pp-dist-comp)
subgoal apply transfer by simp
done

end

instantiation prod :: (relation-algebra,relation-algebra) relation-algebra
begin

lift-definition minus-prod :: "'a × 'b ⇒ 'a × 'b ⇒ 'a × 'b" is λ(u,v) (w,x). (u − w, v − x).

instance
apply intro-classes
subgoal apply transfer by auto
subgoal apply transfer by auto
subgoal apply transfer by (clarsimp, simp add: diff-eq)
done

end

instantiation prod ::
(relation-algebra-atomic-finiteatoms,relation-algebra-atomic-finiteatoms)
relation-algebra-atomic-finiteatoms
begin

instance
apply intro-classes
subgoal apply transfer by (clarsimp, metis atomic bot.extremum inf.antisym-conv)
subgoal
proof −
  have 1: ∀ a::'a . ∀ b::'b . atom (a,b) ⇒ (a = bot ∧ atom b) ∨ (atom a ∧ b = bot)
  proof (intro allI, rule impl)
    fix a :: 'a and b :: 'b
    assume 2: atom (a,b)
    show (a = bot ∧ atom b) ∨ (atom a ∧ b = bot)
    proof (cases a = bot)
      case 3: True
      show ?thesis
      proof (cases b = bot)
        case True
        thus ?thesis
        using 2 3 by (simp add: bot-prod.abs-eq)
      next
      case False
from this obtain $c$ where 4: $\text{atom } c \land c \leq b$
using atomic by auto
hence $(\text{bot},c) \leq (a,b) \land (\text{bot},c) \neq \text{bot}$
by (simp add: less-eq-prod-def bot-prod.abs-eq)
hence $(\text{bot},c) = (a,b)$
using 2 by auto
thus $?thesis$
using 4 by auto
qed

next
case False
from this obtain $c$ where 5: $\text{atom } c \land c \leq a$
using atomic by auto
hence $(c,\text{bot}) \leq (a,b) \land (c,\text{bot}) \neq \text{bot}$
by (simp add: less-eq-prod-def bot-prod.abs-eq)
hence $(c,\text{bot}) = (a,b)$
using 2 by auto
thus $?thesis$
using 5 by auto
qed

have 6: \{ \langle a,b \rangle \mid a . b . \text{atom } (a,b) \} \subseteq \{ \langle \text{bot},b \rangle \mid b::'b . \text{atom } b \} \cup \{ \langle a,\text{bot} \rangle \mid a::'a . \text{atom } a \}

proof
  fix $x :: 'a \times 'b$
  assume $x \in \{ \langle a,b \rangle \mid a . b . \text{atom } (a,b) \}$
  from this obtain $a b$ where 7: $x = (a,b) \land \text{atom } (a,b)$
  by auto
  hence $(a = \text{bot} \land \text{atom } b) \lor (\text{atom } a \land b = \text{bot})$
  by simp
  thus $x \in \{ \langle \text{bot},b \rangle \mid b . \text{atom } b \} \cup \{ \langle a,\text{bot} \rangle \mid a . \text{atom } a \}$
  using 7 by auto
qed

have finite \{ \langle \text{bot},b \rangle \mid b::'b . \text{atom } b \} \land finite \{ \langle a,\text{bot} \rangle \mid a::'a . \text{atom } a \}
by (simp add: finiteatoms)
hence 8: finite \{ \langle \text{bot},b \rangle \mid b::'b . \text{atom } b \} \cup \{ \langle a,\text{bot} \rangle \mid a::'a . \text{atom } a \}\)
by blast
have 9: finite \{ \langle a,b \rangle \mid a . b . \text{atom } (a::'a::'b) \}
by (rule rev-finite-subset, rule 8, rule 6)
have \{ \langle a,b \rangle \mid a . b . \text{atom } (a,b) \} = \{ x :: 'a \times 'b . \text{atom } x \}
by auto
thus finite \{ x :: 'a \times 'b . \text{atom } x \}
using 9 by simp
qed

done

end

instantiation prod ::
(ra-card-notop-atomic-finiteatoms, ra-card-notop-atomic-finiteatoms)
ra-card-notop-atomic-finiteatoms
begin

lift-definition cardinality-prod :: 'a × 'b ⇒ enat = λ(u,v). #u + #v.

instance
  apply intro-classes
  subgoal apply transfer by (smt (verit) card-bot-iff case-prod-conv surj-pair
    zero-eq-add-iff-bot-eq-0)
  subgoal apply transfer by (simp add: card-cone)
  subgoal apply transfer by (clarsimp, metis card-add
    semiring-normalization-rules(20))
  subgoal apply transfer apply (clarsimp, metis iffI)
  subgoal by (metis add.commute add.right-neutral bot.extremum card-atom-iff
    card-bot-iff dual-order.rsf)
  subgoal for a b proof −
    assume 1: #a + #b = 1
    show ?thesis
    proof (cases #a = 0)
      case True
      hence #b = 1
      using 1 by auto
      thus ?thesis
      by (metis True bot.extremum-unique card-atom-iff card-bot-iff)
    next
    case False
    hence #a ≥ 1
    by (simp add: ileI1 one-eSuc)
    hence 2: #a = 1
    using 1 by (metis ile-add1 order-antisym)
    hence #b = 0
    using 1 by auto
    thus ?thesis
    using 2 by (metis bot.extremum-unique card-atom-iff card-bot-iff)
  qed
  qed
  done
subgoal apply transfer by (simp add: add-mono card-univ-comp-meet)
subgoal apply transfer by (simp add: add-mono card-univ-meet-comp)
done

end

type-synonym finite-4-square = Enum.finite-4 × Enum.finite-4

interpretation finite-4-square: ra-card-atomic-finiteatoms where cardinality =
  cardinality and inf = (⊓) and less-eq = (≤) and less = (<) and sup = (⊔) and
  bot = bot::finite-4-square and top = top and uminus = uminus and one = 1
and times = (⋅) and conv = conv and minus = (−) ..

interpretation finite-4-square: ra-card-all-atomic-finiteatoms where cardinality = cardinality and inf = (⊓) and less-eq = (≤) and less = (<) and sup = (⊔) and bot = bot::finite-4-square and top = top and uminus = uminus and one = 1 and times = (⋅) and conv = conv and minus = (−)
  apply unfold-locales
subgoal apply transfer by (simp add: cardinality-finite-4-def one-finite-4-def)
subgoal apply transfer by (smt (verit) card-add card-atom-iff card-bot-iff card-nAB cardinality-prod.abs-eq nAB-top-finite top-prod.abs-eq)
done

lemma counterexample-atom-rectangle-2:
  atom a → a * top * a ≤ (a::finite-4-square)
nitpick [expect:genuine]
oops

lemma counterexample-atom-univalent-2:
  atom a → univalent (a::finite-4-square)
nitpick [expect:genuine]
oops

lemma counterexample-point-dense-2:
  assumes x ≠ bot
  and x ≤ 1
  shows ∃ a::finite-4-square . a ≠ bot ∧ a * top * a ≤ 1 ∧ a ≤ x
nitpick [expect:genuine]
oops

type-synonym ra11 = ra1 × ra1

interpretation ra11: ra-card-atomic-finiteatoms where cardinality = cardinality and inf = (⊓) and less-eq = (≤) and less = (<) and sup = (⊔) and bot = bot::ra11 and top = top and uminus = uminus and one = 1 and times = (⋅) and conv = conv and minus = (−) ..

interpretation ra11: ra-card-all-atomic-finiteatoms where cardinality = cardinality and inf = (⊓) and less-eq = (≤) and less = (<) and sup = (⊔) and bot = bot::ra11 and top = top and uminus = uminus and one = 1 and times = (⋅) and conv = conv and minus = (−)
  apply unfold-locales
subgoal apply transfer apply transfer using r-all-different by auto
subgoal apply transfer apply transfer using numeral-ne-infinity by fastforce
done

interpretation ra11: stone-relation-algebra-atomrect where inf = (⊓) and
less-eq = (≤) and less = (<) and sup = (⊔) and bot = bot::ra11 and top = top
and uminus = uminus and one = 1 and times = (⋅) and conv = conv
  apply unfold-locales
apply transfer apply transfer
nitpick[expect=genuine]
oops

lemma ¬ (∀ a :: ra1 × ra1. atom a → a * top * a ≤ a)
proof 
  let ?a = (1 :: ra1, bot :: ra1)
  have 1: atom ?a
  proof
    show ?a ≠ bot
    by (metis (full-types) bot-prod.transfer bot-ra1.rep-eq one-ra1.rep-eq prod.inject)
    have ∩ (a :: ra1) (b :: ra1). (a, b) ≤ ?a ⟹ (a, b) ≠ bot ⟹ a = 1 ∧ b = bot
    proof
      fix a b :: ra1
      assume (a, b) ≤ ?a
      hence 2: a ≤ 1 ∧ b ≤ bot
      by (simp add: less-eq-prod-def)
      assume (a, b) ≠ bot
      hence 3: a ≠ bot ∧ b = bot
      using 2 by (simp add: bot.extremum-unique bot-prod.abs-eq)
      have atom (1 :: ra1)
        apply transfer apply (rule conjI)
        subgoal by (simp add: r-all-different)
        subgoal by auto
        done
        thus a = 1 ∧ b = bot
        using 2 3 by blast
        qed
        thus ∀ y. y ≠ bot ∧ y ≤ ?a ⟹ y = ?a
        by clarsimp
        qed
      have ¬ ?a * top * ?a ≤ ?a
        apply (unfold top-prod-def times-prod-def less-eq-prod-def)
        apply transfer
        by auto
        thus ?thesis
        using 1 by auto
      qed
    qed
  qed

end

References
