

Cardinality and Representation of Stone Relation Algebras

Walter Guttmann

June 17, 2024

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	12
1.3	Ideals, Ideal-Points and Matrices as Types	15
1.4	Isomorphism	20
2	Atoms Below an Element in Partial Orders	29
3	Atoms Below an Element in Stone Relation Algebras	37
3.1	Atomic	43
3.2	Atom-rectangular	46
3.3	Atomic and Atom-Rectangular	49
3.4	Atom-simple	50
3.5	Atomic and Atom-simple	50
3.6	Atom-rectangular and Atom-simple	52
3.7	Atomic, Atom-rectangular and Atom-simple	53
3.8	Finitely Many Atoms	54
3.9	Atomic and Finitely Many Atoms	54
3.10	Atom-rectangular and Finitely Many Atoms	55
3.11	Atomic, Atom-rectangular and Finitely Many Atoms	55
3.12	Atom-simple and Finitely Many Atoms	56
3.13	Atomic, Atom-simple and Finitely Many Atoms	56

3.14	Atom-rectangular, Atom-simple and Finitely Many Atoms . .	57
3.15	Atomic, Atom-rectangular, Atom-simple and Finitely Many Atoms	57
3.16	Relation Algebra and Atomic	59
3.17	Relation Algebra, Atomic and Finitely Many Atoms	60
4	Cardinality in Stone Relation Algebras	61
4.1	Cardinality in Relation Algebras	69
4.2	Counterexamples	74

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].

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  $F \neq \{\}$ 
  and  $F \subseteq S$ 
  and singleton:  $\bigwedge x . x \in S \implies P \{x\}$ 
  and insert:  $\bigwedge x F . \text{finite } F \implies F \neq \{\} \implies F \subseteq S \implies x \in S \implies x \notin F$ 
 $\implies P F \implies P (\text{insert } x F)$ 
shows  $P F$ 
using assms(1-3)
apply (induct rule: finite-ne-induct)
apply (simp add: singleton)
by (simp add: insert)

```

context *order-bot*

begin

abbreviation *atom* :: '*a* \Rightarrow bool

where $\text{atom } x \equiv x \neq \text{bot} \wedge (\forall y . y \neq \text{bot} \wedge y \leq x \longrightarrow y = x)$

end

context *semilattice-sup*
begin

lemma *nested-sup-fin*:

assumes *finite X*

and $X \neq \{\}$

and *finite Y*

and $Y \neq \{\}$

shows $\text{Sup-fin } \{ \text{Sup-fin } \{ f x y \mid x . x \in X \} \mid y . y \in Y \} = \text{Sup-fin } \{ f x y \mid x y . x \in X \wedge y \in Y \}$

proof (*rule order.antisym*)

have 1: *finite* $\{ f x y \mid x y . x \in X \wedge y \in Y \}$

proof –

have *finite* $(X \times Y)$

by (*simp add: assms(1,3)*)

hence *finite* $\{ f (\text{fst } z) (\text{snd } z) \mid z . z \in X \times Y \}$

by (*metis (mono-tags) Collect-mem-eq finite-image-set*)

thus *?thesis*

by *auto*

qed

show $\text{Sup-fin } \{ \text{Sup-fin } \{ f x y \mid x . x \in X \} \mid y . y \in Y \} \leq \text{Sup-fin } \{ f x y \mid x y . x \in X \wedge y \in Y \}$

apply (*rule Sup-fin.boundedI*)

subgoal by (*simp add: assms(3)*)

subgoal using *assms(4)* **by** *blast*

subgoal for *a*

proof –

assume $a \in \{ \text{Sup-fin } \{ f x y \mid x . x \in X \} \mid y . y \in Y \}$

from *this* **obtain** *y* **where** 2: $y \in Y \wedge a = \text{Sup-fin } \{ f x y \mid x . x \in X \}$

by *auto*

have $\text{Sup-fin } \{ f x y \mid x . x \in X \} \leq \text{Sup-fin } \{ f x y \mid x y . x \in X \wedge y \in Y \}$

apply (*rule Sup-fin.boundedI*)

subgoal by (*simp add: assms(1)*)

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 $\text{Sup-fin } \{ f x y \mid x y . x \in X \wedge y \in Y \} \leq \text{Sup-fin } \{ \text{Sup-fin } \{ f x y \mid x . x \in X \} \mid y . y \in Y \}$

apply (*rule Sup-fin.boundedI*)

subgoal using 1 **by** *simp*

subgoal using *assms(2,4)* **by** *blast*

subgoal for *a*

proof –

assume $a \in \{ f x y \mid x y . x \in X \wedge y \in Y \}$

from *this* **obtain** $x y$ **where** 3: $x \in X \wedge y \in Y \wedge a = f x y$

```

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

end

context bounded-semilattice-sup-bot
begin

lemma one-point-sup-fin:
  assumes finite X
    and  $y \in X$ 
  shows  $\text{Sup-fin } \{ (if\ x = y\ then\ f\ x\ else\ bot) \mid x . x \in X \} = f\ y$ 
proof (rule order.antisym)
  show  $\text{Sup-fin } \{ (if\ x = y\ then\ f\ x\ else\ bot) \mid x . x \in X \} \leq f\ y$ 
    apply (rule Sup-fin.boundedI)
    apply (simp add: assms(1))
    using assms(2) apply blast
    by auto
  show  $f\ y \leq \text{Sup-fin } \{ (if\ x = y\ then\ f\ x\ else\ bot) \mid x . x \in X \}$ 
    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  $\Rightarrow$  bool where ideal x  $\equiv$  vector x  $\wedge$  covector x

```

```

end

```

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

lemma *ideal-fixpoint*:

ideal $x \longleftrightarrow top * x * top = x$

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 $x \Longrightarrow ideal\ y \Longrightarrow ideal\ (x * y)$

by (*metis mult-assoc*)

end

context *bounded-idempotent-left-zero-semiring*
begin

lemma *ideal-sup-closed*:

ideal $x \Longrightarrow ideal\ y \Longrightarrow ideal\ (x \sqcup y)$

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 $Sup\text{-}fin\ \{ f\ x \mid x . x \in UNIV \} = (\bigsqcup_x f\ x)$

proof (*rule order.antisym*)

show $Sup\text{-}fin\ \{ f\ x \mid x . x \in UNIV \} \leq (\bigsqcup_x f\ x)$

apply (*rule Sup-fin.boundedI*)

apply (*metis (mono-tags) finite finite-image-set*)

apply *blast*

using *ub-sum* **by** *auto*

next

show $(\bigsqcup_x f\ x) \leq Sup\text{-}fin\ \{ f\ x \mid x . x \in UNIV \}$

apply (*rule lub-sum, rule allI*)

apply (*rule Sup-fin.coboundedI*)

apply (*metis (mono-tags) finite finite-image-set*)

by *auto*

qed

end

context *stone-relation-algebra*
begin

lemma *dedekind-univalent*:

assumes *univalent y*

shows $x * y \sqcap z = (x \sqcap z * y^T) * y$

proof (*rule order.antisym*)

show $x * y \sqcap z \leq (x \sqcap z * y^T) * y$

by (*simp add: dedekind-2*)

next

have $(x \sqcap z * y^T) * y \leq x * y \sqcap z * y^T * y$

using *comp-left-subdist-inf* **by** *auto*

also have $\dots \leq x * y \sqcap z$

by (*metis assms comp-associative comp-inf.mult-right-isotone comp-right-one mult-right-isotone*)

finally show $(x \sqcap z * y^T) * y \leq x * y \sqcap z$

·
qed

lemma *dedekind-injective*:

assumes *injective x*

shows $x * y \sqcap z = x * (y \sqcap x^T * z)$

proof (*rule order.antisym*)

show $x * y \sqcap z \leq x * (y \sqcap x^T * z)$

by (*simp add: dedekind-1*)

next

have $x * (y \sqcap x^T * z) \leq x * y \sqcap x * x^T * z$

using *comp-associative comp-right-subdist-inf* **by** *auto*

also have $\dots \leq x * y \sqcap z$

by (*metis assms coreflexive-comp-top-inf inf.boundedE inf.boundedI inf.cobounded2 inf-le1*)

finally show $x * (y \sqcap x^T * z) \leq x * y \sqcap z$

·
qed

lemma *domain-vector-conv*:

$1 \sqcap x * top = 1 \sqcap x * x^T$

by (*metis comp-right-one dedekind-eq ex231a inf.sup-monoid.add-commute inf-top.right-neutral total-conv-surjective vector-conv-covector vector-top-closed*)

lemma *domain-vector-covector*:

$1 \sqcap x * top = 1 \sqcap top * x^T$

by (*metis conv-dist-comp one-inf-conv symmetric-top-closed*)

lemma *domain-covector-conv*:

$1 \sqcap \text{top} * x^T = 1 \sqcap x * x^T$
using *domain-vector-conv domain-vector-covector* **by** *auto*

lemma *ideal-bot-closed*:
ideal bot
by *simp*

lemma *ideal-inf-closed*:
ideal x \implies ideal y \implies ideal (x \sqcap y)
by (*simp add: covector-comp-inf vector-inf-comp*)

lemma *ideal-conv-closed*:
ideal x \implies ideal (x^T)
using *covector-conv-vector vector-conv-covector* **by** *blast*

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

lemma *ideal-conv-id*:
ideal x \implies 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 \implies ideal y \implies x * y = x \sqcap y*
by (*metis inf-top-right vector-inf-comp*)

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

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

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

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

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

lemma *different-ideal-points-disjoint*:
assumes *ideal-point* p

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

lemma *points-disjoint-iff*:
assumes *vector* x
shows $x \sqcap y = \text{bot} \longleftrightarrow x^T * y = \text{bot}$
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 $p^T * q = \text{bot}$
using *assms different-ideal-points-disjoint points-disjoint-iff* **by** *blast*

lemma *mult-right-dist-sup-fin*:
assumes *finite* X
and $X \neq \{\}$
shows $\text{Sup-fin } \{ f x \mid x::'b . x \in X \} * y = \text{Sup-fin } \{ f x * y \mid x . x \in X \}$
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 $\bigwedge z . \text{Sup-fin } \{ f x \mid x . x \in \{z\} \} * y = \text{Sup-fin } \{ f x * y \mid x . x \in \{z\} \}$
by *auto*
fix $z F$
assume $1: \text{finite } F \ F \neq \{\} \ z \notin F \ \text{Sup-fin } \{ f x \mid x . x \in F \} * y = \text{Sup-fin } \{ f x * y \mid x . x \in F \}$
have $\{ f x \mid x . x \in \text{insert } z \ F \} = \text{insert } (f z) \{ f x \mid x . x \in F \}$
by *auto*
hence $\text{Sup-fin } \{ f x \mid x . x \in \text{insert } z \ F \} * y = (f z \sqcup \text{Sup-fin } \{ f x \mid x . x \in F \}) * y$
using *Sup-fin.insert 1* **by** *auto*

also have $\dots = f z * y \sqcup \text{Sup-fin } \{ f x \mid x . x \in F \} * y$
using *mult-right-dist-sup* **by** *blast*
also have $\dots = f z * y \sqcup \text{Sup-fin } \{ f x * y \mid x . x \in F \}$
using *1* **by** *simp*
also have $\dots = \text{Sup-fin } (\text{insert } (f z * y) \{ f x * y \mid x . x \in F \})$
using *1* **by** *auto*
also have $\dots = \text{Sup-fin } \{ f x * y \mid x . x \in \text{insert } z F \}$
by (*rule arg-cong*[**where** $f = \text{Sup-fin}$], *auto*)
finally show $\text{Sup-fin } \{ f x \mid x . x \in \text{insert } z F \} * y = \text{Sup-fin } \{ f x * y \mid x . x \in \text{insert } z F \}$
qed

lemma *mult-left-dist-sup-fin*:

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

lemma *inf-left-dist-sup-fin*:

assumes *finite X*
and $X \neq \{\}$
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 $X \neq \{\}$
using *assms*(2) **by** *simp*
show $\bigwedge z . y \sqcap \text{Sup-fin } \{ f x \mid x . x \in \{z\} \} = \text{Sup-fin } \{ y \sqcap f x \mid x . x \in \{z\} \}$
by *auto*
fix $z F$
assume 1: *finite* $F F \neq \{\}$ $z \notin F$ $y \sqcap \text{Sup-fin } \{ f x \mid x . x \in F \} = \text{Sup-fin } \{ y \sqcap f x \mid x . x \in F \}$
have $\{ f x \mid x . x \in \text{insert } z F \} = \text{insert } (f z) \{ f x \mid x . x \in F \}$
by *auto*
hence $y \sqcap \text{Sup-fin } \{ f x \mid x . x \in \text{insert } z F \} = y \sqcap (f z \sqcup \text{Sup-fin } \{ f x \mid x . x \in F \})$
using *Sup-fin.insert 1* **by** *auto*
also have $\dots = (y \sqcap f z) \sqcup (y \sqcap \text{Sup-fin } \{ f x \mid x . x \in F \})$
using *inf-sup-distrib1* **by** *auto*
also have $\dots = (y \sqcap f z) \sqcup \text{Sup-fin } \{ y \sqcap f x \mid x . x \in F \}$
using 1 **by** *simp*
also have $\dots = \text{Sup-fin } (\text{insert } (y \sqcap f z) \{ y \sqcap f x \mid x . x \in F \})$
using 1 **by** *auto*
also have $\dots = \text{Sup-fin } \{ y \sqcap f x \mid x . x \in \text{insert } z F \}$
by (*rule arg-cong*[**where** $f = \text{Sup-fin}$], *auto*)
finally show $y \sqcap \text{Sup-fin } \{ f x \mid x . x \in \text{insert } z F \} = \text{Sup-fin } \{ y \sqcap f x \mid x . x \in \text{insert } z F \}$
.

qed

lemma *top-one-sup-fin-iff*:

assumes *finite* P
and $P \neq \{\}$
and $\forall p \in P . \text{point } p$
shows $\text{top} = \text{Sup-fin } P \iff 1 = \text{Sup-fin } \{ p * p^T \mid p . p \in P \}$

proof

assume $\text{top} = \text{Sup-fin } P$
hence $1 = 1 \sqcap \text{Sup-fin } P$
using *inf-top-right* **by** *auto*
also have $\dots = \text{Sup-fin } \{ 1 \sqcap p \mid p . p \in P \}$
using *inf-Sup1-distrib* *assms*(1,2) **by** *simp*
also have $\dots = \text{Sup-fin } \{ p * p^T \mid p . p \in P \}$
by (*metis* (*no-types*, *opaque-lifting*) *point-meet-one* *assms*(3))
inf.sup-monoid.add-commute
finally show $1 = \text{Sup-fin } \{ p * p^T \mid p . p \in P \}$

next

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

by (*metis (no-types, opaque-lifting) point-meet-one assms(3)*
inf.sup-monoid.add-commute)
also have ... = *Sup-fin* { (1 \sqcap 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*

lemma *point-below-equal*:
point p \implies point q \implies p \leq q \implies p = q
by (*metis below-point-eq-domain comp-associative*)

lemma *ideal-point-without-ideal*:
ideal-point p \longleftrightarrow (point p \wedge (\forall q . point q \longrightarrow q \leq p \vee q \leq -p))

proof
assume 1: *ideal-point p*
have \forall q . point q \longrightarrow q \leq p \vee q \leq -p
proof (*rule allI, rule impI*)
fix q
assume 2: *point q*
hence 3: *ideal (q^T * p)*
using 1 **by** (*metis comp-associative vector-conv-covector*)
have q * (q^T * p) \leq p
using 2 *shunt-mapping surjective-conv-total* **by** *force*
hence q^T * p = bot \vee q \leq p
using 1 2 3 **by** *blast*
thus q \leq p \vee q \leq -p
using 2 **by** (*metis bot-least schroeder-3-p*)
qed
thus point p \wedge (\forall q . point q \longrightarrow q \leq p \vee q \leq -p)
using 1 **by** *blast*

next
assume 4: *point p \wedge (\forall q . point q \longrightarrow q \leq p \vee q \leq -p)*
have \forall y z . point y \wedge ideal z \wedge z \neq bot \wedge y * z \leq p \longrightarrow y \leq p

```

proof (intro allI, rule impI)
  fix y z
  assume 5: point y  $\wedge$  ideal z  $\wedge$  z  $\neq$  bot  $\wedge$  y * z  $\leq$  p
  show y  $\leq$  p
  proof (rule ccontr)
    assume  $\neg$  y  $\leq$  p
    hence y  $\leq$  -p
    using 4 5 by blast
    hence yT * p = bot
    using 5 points-disjoint-iff pseudo-complement by blast
    thus False
    using 5 bot-unique shunt-mapping surjective-conv-total by force
  qed
qed
thus ideal-point p
  using 4 by blast
qed

```

```

lemma ideal-point-without-ideal-2:
  ideal-point p  $\longleftrightarrow$  (point p  $\wedge$  ( $\forall$  q . point q  $\longrightarrow$  q = p  $\vee$  q  $\leq$  -p))
  by (smt (verit) ideal-point-without-ideal point-below-equal comp-associative
  mult-semi-associative)

```

```

lemma ideal-point-without-ideal-3:
  ideal-point p  $\longleftrightarrow$  (point p  $\wedge$  ( $\forall$  q . point q  $\wedge$  q  $\neq$  p  $\longrightarrow$  q  $\leq$  -p))
  using ideal-point-without-ideal-2 by force

```

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 * pT | p . ideal-point p }
proof -
  have 1 = Sup-fin { p * pT | 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 * pT | p . ideal-point p }
    by simp
  finally show ?thesis
qed

```

lemma *ideal-point-rep-1*:
 $x = \text{Sup-fin } \{ p * p^T * x * q * q^T \mid p \ q . \text{ideal-point } p \wedge \text{ideal-point } q \}$
proof –
let $?p = \{ p * p^T \mid p . p \in \text{ideal-points} \}$
have $x = \text{Sup-fin } ?p * (x * \text{Sup-fin } ?p)$
using *one-sup-ideal-points* **by** *auto*
also have $\dots = \text{Sup-fin } \{ p * p^T * (x * \text{Sup-fin } ?p) \mid p . p \in \text{ideal-points} \}$
apply (*rule mult-right-dist-sup-fin*)
using *finite-ideal-points ne-ideal-points* **by** *simp-all*
also have $\dots = \text{Sup-fin } \{ p * p^T * x * \text{Sup-fin } ?p \mid p . p \in \text{ideal-points} \}$
using *mult-assoc* **by** *simp*
also have $\dots = \text{Sup-fin } \{ \text{Sup-fin } \{ p * p^T * x * q * q^T \mid q . q \in \text{ideal-points} \} \mid p . p \in \text{ideal-points} \}$
proof –
have $\bigwedge p . p * p^T * x * \text{Sup-fin } ?p = \text{Sup-fin } \{ p * p^T * x * (q * q^T) \mid q . q \in \text{ideal-points} \}$
apply (*rule mult-left-dist-sup-fin*)
using *finite-ideal-points ne-ideal-points* **by** *simp-all*
thus *?thesis*
using *mult-assoc* **by** *simp*
qed
also have $\dots = \text{Sup-fin } \{ p * p^T * x * q * q^T \mid q \ p . q \in \text{ideal-points} \wedge p \in \text{ideal-points} \}$
apply (*rule nested-sup-fin*)
using *finite-ideal-points ne-ideal-points* **by** *simp-all*
also have $\dots = \text{Sup-fin } \{ p * p^T * x * q * q^T \mid p \ q . p \in \text{ideal-points} \wedge q \in \text{ideal-points} \}$
by *meson*
also have $\dots = \text{Sup-fin } \{ p * p^T * x * q * q^T \mid p \ q . \text{ideal-point } p \wedge \text{ideal-point } q \}$
by *auto*
finally show *?thesis*
qed

lemma *atom-below-ideal-point*:
assumes *atom a*
shows $\exists p . \text{ideal-point } p \wedge a \leq p$
proof –
have $a = a \sqcap \text{Sup-fin } \{ p \mid p . p \in \text{ideal-points} \}$
using *top-sup-ideal-points* **by** *auto*
also have $\dots = \text{Sup-fin } \{ a \sqcap p \mid p . p \in \text{ideal-points} \}$
apply (*rule inf-left-dist-sup-fin*)
using *finite-ideal-points* **apply** *blast*
using *ne-ideal-points* **by** *blast*
finally have *1*: $\text{Sup-fin } \{ a \sqcap p \mid p . p \in \text{ideal-points} \} \neq \text{bot}$
using *assms* **by** *auto*
have $\exists p \in \text{ideal-points} . a \sqcap p \neq \text{bot}$

```

proof (rule ccontr)
  assume  $\neg (\exists p \in \text{ideal-points} . a \sqcap p \neq \text{bot})$ 
  hence  $\forall p \in \text{ideal-points} . a \sqcap p = \text{bot}$ 
  by auto
  hence  $\{ a \sqcap p \mid p . p \in \text{ideal-points} \} = \{ \text{bot} \mid p . p \in \text{ideal-points} \}$ 
  by auto
  hence  $\text{Sup-fin } \{ a \sqcap p \mid p . p \in \text{ideal-points} \} = \text{Sup-fin } \{ \text{bot} \mid p . p \in \text{ideal-points} \}$ 
  by simp
  also have  $\dots \leq \text{bot}$ 
  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  $p \in \text{ideal-points} \wedge a \sqcap p \neq \text{bot}$ 
by auto
hence ideal-point  $p \wedge a \leq p$ 
using assms inf.absorb-iff1 inf-le1 by blast
thus ?thesis
by auto
qed

```

lemma *point-ideal-point-1*:

```

  assumes point a
  shows ideal-point a
proof (cases  $a = \text{bot}$ )
  case True
  thus ?thesis
  using assms by fastforce
next
  case False
  have  $a = a \sqcap \text{Sup-fin } \{ p \mid p . p \in \text{ideal-points} \}$ 
  using top-sup-ideal-points by auto
  also have  $\dots = \text{Sup-fin } \{ a \sqcap p \mid p . p \in \text{ideal-points} \}$ 
  apply (rule inf-left-dist-sup-fin)
  using finite-ideal-points apply blast
  using ne-ideal-points by blast
  finally have 1:  $\text{Sup-fin } \{ a \sqcap p \mid p . p \in \text{ideal-points} \} \neq \text{bot}$ 
  using False by auto
  have  $\exists p \in \text{ideal-points} . a \sqcap p \neq \text{bot}$ 
  proof (rule ccontr)
  assume  $\neg (\exists p \in \text{ideal-points} . a \sqcap p \neq \text{bot})$ 
  hence  $\forall p \in \text{ideal-points} . a \sqcap p = \text{bot}$ 
  by auto
  hence  $\{ a \sqcap p \mid p . p \in \text{ideal-points} \} = \{ \text{bot} \mid p . p \in \text{ideal-points} \}$ 
  by auto

```

```

    hence Sup-fin {  $a \sqcap p \mid p . p \in \text{ideal-points}$  } = Sup-fin {  $\text{bot} \mid p . p \in \text{ideal-points}$  }
    by simp
    also have ...  $\leq \text{bot}$ 
    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 2:  $p \in \text{ideal-points} \wedge a \sqcap p \neq \text{bot}$ 
by auto
hence  $a \leq p \vee a \leq -p$ 
using assms ideal-point-without-ideal by auto
hence  $a \leq p$ 
using 2 pseudo-complement by blast
thus ?thesis
using 2 assms point-below-equal by blast
qed

lemma point-ideal-point:
   $\text{point } x \longleftrightarrow \text{ideal-point } x$ 
  using point-ideal-point-1 by blast

```

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
  by simp

lift-definition less-eq-ideal :: 'a ideal  $\Rightarrow$  'a ideal  $\Rightarrow$  bool is less-eq .

lift-definition less-ideal :: 'a ideal  $\Rightarrow$  'a ideal  $\Rightarrow$  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  $\Rightarrow$  'a ideal is id
  by simp

lift-definition times-ideal :: 'a ideal  $\Rightarrow$  'a ideal  $\Rightarrow$  '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) square

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 uminus = uminus-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 ... = $?r^T * \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p q . p \in \text{UNIV} \wedge q \in \text{UNIV} \} * ?s$
by *simp*
also have ... = $?r^T * \text{Sup-fin } \{ \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \} * ?s$
proof –
have $\text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p q . p \in \text{UNIV} \wedge q \in \text{UNIV} \} = \text{Sup-fin } \{ \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \}$
by (*rule nested-sup-fin[symmetric], simp-all*)
thus *?thesis*
by *simp*
qed
also have ... = $\text{Sup-fin } \{ \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \} * ?s$
proof –
have 1: $?r^T * \text{Sup-fin } \{ \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \} = \text{Sup-fin } \{ ?r^T * \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \}$
by (*rule mult-left-dist-sup-fin, simp-all*)
have 2: $\bigwedge q . ?r^T * \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} = \text{Sup-fin } \{ ?r^T * (\text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T) \mid p . p \in \text{UNIV} \}$
by (*rule mult-left-dist-sup-fin, simp-all*)
have $\bigwedge p q . ?r^T * (\text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T) = ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T$
by (*simp add: mult.assoc*)
thus *?thesis*
using 1 2 **by** *simp*
qed
also have ... = $\text{Sup-fin } \{ \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T * ?s \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \}$
proof –
have 3: $\text{Sup-fin } \{ \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \} * ?s = \text{Sup-fin } \{ \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} * ?s \mid q . q \in \text{UNIV} \}$
by (*rule mult-right-dist-sup-fin, simp-all*)
have $\bigwedge q . \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T \mid p . p \in \text{UNIV} \} * ?s = \text{Sup-fin } \{ ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T * ?s \mid p . p \in \text{UNIV} \}$
by (*rule mult-right-dist-sup-fin, simp-all*)
thus *?thesis*
using 3 **by** *simp*
qed
also have ... = $\text{Sup-fin } \{ \text{Sup-fin } \{ \text{if } p = r \text{ then } ?r^T * \text{Rep-ideal-point } p * \text{Rep-ideal } (f p q) * (\text{Rep-ideal-point } q)^T * ?s \text{ else bot} \mid p . p \in \text{UNIV} \} \mid q . q \in \text{UNIV} \}$

```

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

```

```

    using Rep-ideal-point apply blast
    using Rep-ideal-point apply blast
    using False by (simp add: Rep-ideal-point-inject)
  thus ?thesis
    using False by simp
qed
qed
  hence  $\bigwedge q . ?r^T * ?r * \text{Rep-ideal } (f \ r \ q) * (\text{Rep-ideal-point } q)^T * ?s = (\text{if } q = s \text{ then } ?r^T * ?r * \text{Rep-ideal } (f \ r \ q) * (\text{Rep-ideal-point } q)^T * ?s \text{ else bot})$ 
    by (metis comp-associative mult-right-zero)
  thus ?thesis
    by simp
qed
  also have  $\dots = ?r^T * ?r * \text{Rep-ideal } (f \ r \ s) * ?s^T * ?s$ 
    by (subst one-point-sup-fn, simp-all)
  also have  $\dots = \text{top} * \text{Rep-ideal } (f \ r \ s) * \text{top}$ 
  proof -
    have  $?r^T * ?r = \text{top} \wedge ?s^T * ?s = \text{top}$ 
      using point-mult-top Rep-ideal-point by blast
    thus ?thesis
      by (simp add: mult.assoc)
  qed
  also have  $\dots = \text{Rep-ideal } (f \ r \ s)$ 
    by (metis (mono-tags, lifting) Rep-ideal mem-Collect-eq)
  finally show ?thesis
    by (simp add: Rep-ideal-inverse)
qed

```

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.

definition *sra-to-mat* :: *'a::stone-relation-algebra-pa* \Rightarrow *'a ideal-matrix*
where *sra-to-mat* $x \equiv \lambda(p,q) . \text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } q)$

definition *mat-to-sra* :: *'a::stone-relation-algebra-pa ideal-matrix* \Rightarrow *'a*
where *mat-to-sra* $f \equiv \text{Sup-fin } \{ \text{Rep-ideal-point } p * \text{Rep-ideal } (f \ (p,q)) * (\text{Rep-ideal-point } q)^T \mid p \ q . \text{True} \}$

lemma *sra-mat-sra*:

$$\text{mat-to-sra } (\text{sra-to-mat } x) = x$$

proof –

have *mat-to-sra* (*sra-to-mat* x) = *Sup-fin* { *Rep-ideal-point* $p * \text{Rep-ideal } (\text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } q)) * (\text{Rep-ideal-point } q)^T \mid p \ q . \text{True} \}$

by (*unfold sra-to-mat-def mat-to-sra-def, simp*)

$$\text{also have } \dots = \text{Sup-fin } \{ \text{Rep-ideal-point } p * (\text{Rep-ideal-point } p)^T * x * \dots \}$$

```

Rep-ideal-point q * (Rep-ideal-point q)T | p q . True }
proof -
  have  $\bigwedge p q . \text{ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } q)$ 
    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-fin { p * pT * x * q * qT | p q . ideal-point p  $\wedge$  ideal-point q }
proof -
  have { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T | p q . True } = { p * pT * x * q * qT | p q . ideal-point p  $\wedge$  ideal-point q }
  proof (rule set-eqI)
    fix z
    show z  $\in$  { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T | p q . True }  $\longleftrightarrow$  z  $\in$  { p * pT * x * q * qT | p q . ideal-point p  $\wedge$  ideal-point q }
  proof
    assume z  $\in$  { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T | p q . True }
    from this obtain p q where z = Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (Rep-ideal-point q)T
    by auto
    thus z  $\in$  { p * pT * x * q * qT | p q . ideal-point p  $\wedge$  ideal-point q }
    using Rep-ideal-point by blast
  next
    assume z  $\in$  { p * pT * x * q * qT | p q . ideal-point p  $\wedge$  ideal-point q }
    from this obtain p q where 1: ideal-point p  $\wedge$  ideal-point q  $\wedge$  z = p * pT * x * q * qT
    by auto
    hence Rep-ideal-point (Abs-ideal-point p) = p  $\wedge$  Rep-ideal-point (Abs-ideal-point q) = q
    using Abs-ideal-point-inverse by auto
    thus z  $\in$  { Rep-ideal-point p * (Rep-ideal-point p)T * x * Rep-ideal-point q * (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*:

sra-to-mat (mat-to-sra f) = f

by (*unfold sra-to-mat-def mat-to-sra-def, simp add:*
ideal-point-rep-2[symmetric])

lemma *sra-to-mat-sup-homomorphism*:

sra-to-mat (x \sqcup y) = sra-to-mat x \sqcup sra-to-mat y

proof (*rule ext,unfold split-paired-all*)

fix *p q*

have *sra-to-mat (x \sqcup y) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * (x \sqcup 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 \sqcup (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) \sqcup Abs-ideal ((Rep-ideal-point p)^T * y * Rep-ideal-point q)*

proof (*rule sup-ideal.abs-eq[symmetric]*)

have *1: $\bigwedge x$. ideal-point (Rep-ideal-point x)::'a)*

using *Rep-ideal-point by blast*

hence *2: 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)*

show *eq-onp 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-onp-same-args)*

qed

also have *... = sra-to-mat x (p,q) \sqcup sra-to-mat y (p,q)*

by (*unfold sra-to-mat-def, simp*)

finally show *sra-to-mat (x \sqcup y) (p,q) = (sra-to-mat x \sqcup sra-to-mat y) (p,q)*

by *simp*

qed

lemma *sra-to-mat-inf-homomorphism*:

sra-to-mat (x \sqcap y) = sra-to-mat x \sqcap sra-to-mat y

proof (*rule ext,unfold split-paired-all*)

fix *p q*

have *sra-to-mat (x \sqcap y) (p,q) = Abs-ideal ((Rep-ideal-point p)^T * (x \sqcap 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 \sqcap (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 ((Rep-ideal-point p)^T * x * Rep-ideal-point q) □ Abs-ideal ((Rep-ideal-point p)^T * y * Rep-ideal-point q)*
proof (*rule inf-ideal.abs-eq[symmetric]*)
have 1: $\bigwedge x . \text{ideal-point } (\text{Rep-ideal-point } x :: 'a)$
using *Rep-ideal-point by blast*
hence 2: *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*)
show *eq-onp 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-onp-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 q)^T * x * Rep-ideal-point p)*
proof –
have *ideal-point (Rep-ideal-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:*

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

lemma *sra-to-mat-bot-homomorphism:*

$sra\text{-to-mat } bot = bot$

proof (rule ext,unfold split-paired-all)

fix $p\ q :: 'a\ ideal\text{-point}$

have $sra\text{-to-mat } bot (p,q) = Abs\text{-ideal } ((Rep\text{-ideal-point } p)^T * bot * Rep\text{-ideal-point } q)$

by (unfold sra-to-mat-def, simp)

also have $\dots = bot$

by (simp add: *bot-ideal.abs-eq*)

finally show $sra\text{-to-mat } bot (p,q) = bot (p,q)$

by simp

qed

lemma *sra-to-mat-top-homomorphism:*

$sra\text{-to-mat } top = top$


```

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)
  by (unfold sra-to-mat-def, simp)
  also have ... = top
  proof -
    have  $\bigwedge x$  . ideal-point (Rep-ideal-point x::'a)
    using Rep-ideal-point by blast
    thus ?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:

sra-to-mat 1 = one-matrix

```

proof (rule ext,unfold split-paired-all)
  fix p q :: 'a ideal-point
  have sra-to-mat 1 (p,q) = Abs-ideal ((Rep-ideal-point p)T * Rep-ideal-point q)
  by (unfold sra-to-mat-def, simp)
  also have ... = one-matrix (p,q)
  proof (cases p = q)
    case True
    hence (Rep-ideal-point p)T * Rep-ideal-point q = top
    using Rep-ideal-point point-mult-top by auto
    hence Abs-ideal ((Rep-ideal-point p)T * 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 ?thesis
  .

```

next

case False

have (Rep-ideal-point p)^T * Rep-ideal-point q = 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 ?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 $\text{Abs-ideal } (\text{Sup-fin } \{ f x \mid x . x \in X \}) = \text{Sup-fin } \{ \text{Abs-ideal } (f x) \mid x . x \in X \}$

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 $\text{Abs-ideal } (\text{Sup-fin } \{ f x \mid x . x \in \{y\} \}) = \text{Sup-fin } \{ \text{Abs-ideal } (f x) \mid x . x \in \{y\} \}$
by *auto*
fix *F*
assume $2: \text{finite } F \ F \neq \{\} \ F \subseteq X \ y \notin F \ \text{Abs-ideal } (\text{Sup-fin } \{ f x \mid x . x \in F \}) = \text{Sup-fin } \{ \text{Abs-ideal } (f x) \mid x . x \in F \}$
have $\text{Abs-ideal } (\text{Sup-fin } \{ f x \mid x . x \in \text{insert } y \ F \}) = \text{Abs-ideal } (f y \sqcup \text{Sup-fin } \{ f x \mid x . x \in F \})$
proof –
have $\text{Sup-fin } \{ f x \mid x . x \in \text{insert } y \ F \} = f y \sqcup \text{Sup-fin } \{ f x \mid x . x \in F \}$
apply (*subst Sup-fin.insert[symmetric]*)
using 2 **apply** *simp*
using 2 **apply** *simp*
by (*auto intro: arg-cong*[**where** $f=\text{Sup-fin}$])
thus *?thesis*
by *simp*
qed
also have $\dots = \text{Abs-ideal } (f y) \sqcup \text{Abs-ideal } (\text{Sup-fin } \{ f x \mid x . x \in F \})$
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 $\text{top} * \text{Sup-fin } \{ f x \mid x . x \in F \} = \text{Sup-fin } \{ \text{top} * f x \mid x . x \in F \}$
using 2 *mult-left-dist-sup-fin* **by** *fastforce*
hence $\text{top} * \text{Sup-fin } \{ f x \mid x . x \in F \} * \text{top} = \text{Sup-fin } \{ \text{top} * f x \mid x . x \in F \} * \text{top}$
by *simp*
also have $\dots = \text{Sup-fin } \{ \text{top} * f x * \text{top} \mid x . x \in F \}$
using 2 *mult-right-dist-sup-fin* **by** *force*
also have $\dots = \text{Sup-fin } \{ f x \mid x . x \in F \}$
using 2 **by** (*metis assms(3) subset-iff*)
finally have $\text{top} * \text{Sup-fin } \{ f x \mid x . x \in F \} * \text{top} = \text{Sup-fin } \{ f x \mid x . x \in F \}$
}
hence *ideal (Sup-fin { f x | x . x ∈ F })*

```

    using ideal-fixpoint by blast
    thus eq-onp ideal (Sup-fin { f x | x . x ∈ F }) (Sup-fin { f x | x . x ∈ 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 * rT | 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 * rT | 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 * rT | r::'a . ideal-point r } ⟷ x ∈ { Rep-ideal-point r *
(Rep-ideal-point r)T | r . r ∈ UNIV }
proof
    assume x ∈ { r * rT | r::'a . ideal-point r }
    from this obtain r where 1: ideal-point r ∧ x = r * rT
    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

```

```

    thus  $x \in \{ r * r^T \mid r :: 'a . \text{ideal-point } r \}$ 
      using Rep-ideal-point by blast
  qed
  qed
  thus ?thesis
    by simp
  qed
  also have ... = Abs-ideal (Sup-fin { (Rep-ideal-point p)T * x * Rep-ideal-point r
    * (Rep-ideal-point r)T \mid r . r \in UNIV } * (y * Rep-ideal-point q))
    by (subst mult-left-dist-sup-fin, simp-all add: mult.assoc)
  also have ... = Abs-ideal (Sup-fin { (Rep-ideal-point p)T * x * Rep-ideal-point r
    * (Rep-ideal-point r)T * y * Rep-ideal-point q \mid r . r \in UNIV })
    by (subst mult-right-dist-sup-fin, simp-all add: mult.assoc)
  also have ... = Sup-fin { Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r
    * (Rep-ideal-point r)T * y * Rep-ideal-point q) \mid r . r \in UNIV }
  proof -
    have 1:  $\bigwedge r . \text{ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r * (\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q)$ 
  proof -
    fix  $r :: 'a \text{ ideal-point}$ 
    have  $\bigwedge x . \text{ideal-point } (\text{Rep-ideal-point } x :: 'a)$ 
      using Rep-ideal-point by blast
    thus ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r * (Rep-ideal-point r)T * y * 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 ... =  $(\bigsqcup_r \text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r * (\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q))$ 
    by (rule sup-fin-sum)
  also have ... =  $(\bigsqcup_r \text{Abs-ideal } ((\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r \sqcap (\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q))$ 
  proof -
    have  $\bigwedge r . (\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r * ((\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q) = (\text{Rep-ideal-point } p)^T * x * \text{Rep-ideal-point } r \sqcap (\text{Rep-ideal-point } r)^T * y * \text{Rep-ideal-point } q$ 
  proof (rule ideal-mult-inf)
    fix  $r :: 'a \text{ ideal-point}$ 
    have 2:  $\bigwedge x . \text{ideal-point } (\text{Rep-ideal-point } x :: 'a)$ 
      using Rep-ideal-point by blast
    thus ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r)
      by (simp add: covector-mult-closed vector-conv-covector vector-mult-closed)
    show ideal ((Rep-ideal-point r)T * y * Rep-ideal-point q)
      using 2 by (simp add: covector-mult-closed vector-conv-covector vector-mult-closed)
  qed
  qed

```

```

thus ?thesis
  by (simp add: mult.assoc)
qed
also have ... = ( $\bigsqcup_r$  Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r) *
Abs-ideal ((Rep-ideal-point r)T * y * Rep-ideal-point q))
proof -
  have  $\bigwedge r$  . Abs-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r  $\sqcap$ 
(Rep-ideal-point r)T * y * Rep-ideal-point q) = Abs-ideal ((Rep-ideal-point p)T *
x * Rep-ideal-point r) * Abs-ideal ((Rep-ideal-point r)T * y * Rep-ideal-point q)
  proof (rule times-ideal.abs-eq[symmetric])
    fix r :: 'a ideal-point
    have  $\exists$ :  $\bigwedge x$  . ideal-point (Rep-ideal-point x::'a)
      using Rep-ideal-point by blast
    hence  $\exists$ : covector ((Rep-ideal-point p)T)  $\wedge$  covector ((Rep-ideal-point r)T)
      using vector-conv-covector by blast
    thus eq-onp-ideal ((Rep-ideal-point p)T * x * Rep-ideal-point r)
((Rep-ideal-point p)T * x * Rep-ideal-point r)
      using  $\exists$  by (simp add: comp-associative covector-mult-closed
eq-onp-same-args)
    show eq-onp-ideal ((Rep-ideal-point r)T * y * Rep-ideal-point q)
((Rep-ideal-point r)T * y * Rep-ideal-point q)
      using  $\exists$   $\exists$  by (simp add: comp-associative covector-mult-closed
eq-onp-same-args)
  qed
thus ?thesis
  by simp
qed
also have ... = ( $\bigsqcup_r$  sra-to-mat x (p,r) * sra-to-mat y (r,q))
  by (unfold sra-to-mat-def, simp)
finally show sra-to-mat (x * y) (p,q) = (sra-to-mat x  $\odot$  sra-to-mat y) (p,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 *enat*, which are natural numbers with infinity, and *icard*, which modifies *card* by giving a separate option of being infinite. We include general results about *enat*, *icard*, sets functions and atoms.

lemma *enat-mult-strict-mono*:

assumes $a < b$ $c < d$ $(0::enat) < b$ $0 \leq c$
shows $a * c < b * d$

proof –

have $a \neq \infty \wedge c \neq \infty$

using *assms*(1,2) *linorder-not-le* **by** *fastforce*

thus *?thesis*

using *assms* **by** (*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*)

qed

lemma *enat-mult-strict-mono'*:

assumes $a < b$ **and** $c < d$ **and** $(0::enat) \leq a$ **and** $0 \leq c$

shows $a * c < b * d$

using *assms* **by** (*auto* *simp* *add*: *enat-mult-strict-mono*)

lemma *finite-icard-card*:

finite $A \implies \text{icard } A = \text{icard } B \implies \text{card } A = \text{card } B$

by (*metis* *icard-def* *icard-eq-enat-imp-card*)

lemma *icard-eq-sum*:

finite $A \implies \text{icard } A = \text{sum } (\lambda x. 1) A$

by (*simp* *add*: *icard-def* *of-nat-eq-enat*)

lemma *icard-sum-constant-function*:

assumes $\forall x \in A. f x = c$

and *finite* A

shows $\text{sum } f A = (\text{icard } A) * c$

by (*metis* *assms* *icard-finite-conv* *of-nat-eq-enat* *sum.cong* *sum-constant*)

lemma *icard-le-finite*:

assumes $\text{icard } A \leq \text{icard } B$

and *finite* B

shows *finite* A

by (*metis* *assms* *enat-ord-simps*(5) *icard-infinite-conv*)

lemma *bij-betw-same-icard*:

bij-betw $f A B \implies \text{icard } A = \text{icard } B$

by (*simp* *add*: *bij-betw-finite* *bij-betw-same-card* *icard-def*)

lemma *surj-icard-le*: $B \subseteq f ' A \implies \text{icard } B \leq \text{icard } A$

by (*meson* *icard-image-le* *icard-mono* *preorder-class.order-trans*)

lemma *icard-image-part-le*:
assumes $\forall x \in A . f x \subseteq B$
and $\forall x \in A . f x \neq \{\}$
and $\forall x \in A . \forall y \in A . x \neq y \longrightarrow f x \cap f y = \{\}$
shows $\text{icard } A \leq \text{icard } B$
proof –
have $\forall x \in A . \exists y . y \in f x \cap B$
using *assms(1,2)* **by** *fastforce*
hence $\exists g . \forall x \in A . g x \in f x \cap B$
using *bchoice* **by** *simp*
from *this* **obtain** *g* **where** $1: \forall x \in A . g x \in f x \cap B$
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 $\forall x \in A . f x \subseteq B$
and $\forall x \in A . f x \neq \{\}$
and $\forall x \in A . \forall y \in A . x \neq y \longrightarrow f x \cap f y = \{\}$
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 $\forall x \in A . f x = c$
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* $a \in A$ $a \notin F$ $\text{sum } f F = \text{of-nat } (\text{card } F) * c$
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

end

context order
begin

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

end

context order-bot
begin

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

definition num-atoms-below :: 'a  $\Rightarrow$  enat (nAB)
  where num-atoms-below  $x \equiv \text{icard } (\text{atoms-below } x)$ 

lemma AB-iso:
   $x \leq y \implies AB\ x \subseteq AB\ y$ 
  by (simp add: Collect-mono dual-order.trans)

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

lemma nAB-bot:
   $nAB\ \text{bot} = 0$ 
proof -
  have  $nAB\ \text{bot} = \text{icard } (AB\ \text{bot})$ 

```



```

    by (simp add: num-atoms-below-def)
  also have ... = 0
    by (metis (mono-tags, lifting) AB-bot icard-empty)
  finally show ?thesis
.
qed

lemma AB-atom:
  atom a  $\longleftrightarrow$  AB a = {a}
  by blast

lemma nAB-atom:
  atom a  $\implies$  nAB a = 1
proof -
  assume atom a
  hence 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  $\sqcup$  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 = bot
  using inf-le1 le-iff-inf by auto

lemma AB-dist-inf:
  AB (x  $\sqcap$  y) = AB x  $\cap$  AB y
  by auto

lemma AB-iso-inf:
  AB (x  $\sqcap$  y)  $\subseteq$  AB x

```

by (*simp add: Collect-mono*)

lemma *AB-iso-sup*:
 $AB\ x \subseteq AB\ (x \sqcup y)$
by (*simp add: Collect-mono le-supI1*)

lemma *AB-disjoint*:
assumes $x \sqcap y = bot$
shows $AB\ x \cap AB\ y = \{\}$
proof (*rule Int-emptyI*)
fix a
assume $a \in AB\ x\ a \in AB\ y$
hence $atom\ a \wedge a \leq x \wedge a \leq y$
by *simp*
thus *False*
using *assms bot-unique by fastforce*
qed

lemma *nAB-iso-inf*:
 $nAB\ (x \sqcap y) \leq nAB\ x$
by (*simp add: nAB-iso*)

end

context *distrib-lattice-bot*
begin

lemma *atom-in-sup*:
assumes $atom\ a$
and $a \leq x \sqcup y$
shows $a \leq x \vee a \leq y$
proof –
have $1: a = (a \sqcap x) \sqcup (a \sqcap y)$
using *assms(2) inf-sup-distrib1 le-iff-inf by force*
have $a \sqcap x = bot \vee a \sqcap x = a$
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 $a \leq x \sqcup y \iff a \leq x \vee a \leq y$
using *assms atom-in-sup le-supI1 le-supI2 by blast*

lemma *atom-in-sup-xor*:
 $atom\ a \implies a \leq x \sqcup y \implies x \sqcap y = bot \implies (a \leq x \wedge \neg a \leq y) \vee (\neg a \leq x \wedge a \leq y)$
using *atom-in-sup bot-unique le-inf-iff by blast*

lemma *atom-in-sup-xor-iff*:
assumes *atom a*
and $x \sqcap y = \text{bot}$
shows $a \leq x \sqcup y \iff (a \leq x \wedge \neg a \leq y) \vee (\neg a \leq x \wedge a \leq y)$
using *assms atom-in-sup-xor le-supI1 le-supI2* **by** *auto*

lemma *AB-dist-sup*:
 $AB (x \sqcup y) = AB x \cup AB y$
proof
show $AB (x \sqcup y) \subseteq AB x \cup AB y$
using *atom-in-sup* **by** *fastforce*
next
show $AB x \cup AB y \subseteq AB (x \sqcup y)$
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 $nAB x + nAB y = \text{icard} (AB x \cup AB y) + \text{icard} (AB x \cap AB y)$
using *num-atoms-below-def icard-Un-Int* **by** *auto*
also have $\dots = nAB (x \sqcup y) + nAB (x \sqcap y)$
using *num-atoms-below-def AB-dist-inf AB-dist-sup* **by** *auto*
finally show *?thesis*
qed

lemma *nAB-split-disjoint*:
assumes $x \sqcap y = \text{bot}$
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*:
 $\text{atom } a \implies a \leq x \vee a \leq -x$
using *inf.orderI pseudo-complement* **by** *force*

lemma *atom-in-p-xor*:
 $\text{atom } a \implies (a \leq x \wedge \neg a \leq -x) \vee (\neg a \leq x \wedge a \leq -x)$

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 *atom-in-sup'*:

$atom\ a \implies a \leq x \sqcup y \implies a \leq x \vee a \leq y$

by (*metis inf.absorb-iff2 inf.sup-ge2 pseudo-complement sup-least*)

lemma *AB-dist-sup'*:

$AB\ (x \sqcup y) = AB\ x \cup AB\ y$

proof

show $AB\ (x \sqcup y) \subseteq AB\ x \cup AB\ y$

using *atom-in-sup'* by *fastforce*

next

show $AB\ x \cup AB\ y \subseteq AB\ (x \sqcup y)$

using *le-supI1 le-supI2* by *fastforce*

qed

lemma *AB-split-1*:

$AB\ x = AB\ ((x \sqcap y) \sqcup (x \sqcap -y))$

proof

show $AB\ x \subseteq AB\ ((x \sqcap y) \sqcup (x \sqcap -y))$

proof

fix *a*

assume $a \in AB\ x$

hence $atom\ a \wedge a \leq x$

by *simp*

hence $atom\ a \wedge a \leq (x \sqcap y) \sqcup (x \sqcap -y)$

by (*metis atom-in-p-xor inf.boundedI le-supI1 le-supI2*)

thus $a \in AB\ ((x \sqcap y) \sqcup (x \sqcap -y))$

by *simp*

qed

next

show $AB\ ((x \sqcap y) \sqcup (x \sqcap -y)) \subseteq AB\ x$

using *atom-in-sup' inf.boundedE* by *blast*

qed

lemma *AB-split-2*:

$AB\ x = AB\ (x \sqcap y) \cup AB\ (x \sqcap -y)$

using *AB-dist-sup' AB-split-1* by *auto*

lemma *AB-split-2-disjoint*:

$AB\ (x \sqcap y) \cap AB\ (x \sqcap -y) = \{\}$

using *atom-in-p-xor* by *fastforce*

lemma *AB-pp*:

$AB\ (-x) = AB\ x$

by (*metis (opaque-lifting) atom-in-p-xor*)

```

lemma nAB-pp:
  nAB (¬¬x) = nAB x
  using AB-pp num-atoms-below-def by auto

lemma nAB-split-1:
  nAB x = nAB ((x ⊓ y) ⊔ (x ⊓ ¬y))
  using AB-split-1 num-atoms-below-def by simp

lemma nAB-split-2:
  nAB x = nAB (x ⊓ y) + nAB (x ⊓ ¬y)
proof –
  have icard (AB (x ⊓ y)) + icard (AB (x ⊓ ¬y)) = icard (AB (x ⊓ y) ∪ AB (x ⊓ ¬y)) + icard (AB (x ⊓ y) ∩ AB (x ⊓ ¬y))
  using icard-Un-Int by auto
  also have ... = icard (AB x)
  using AB-split-2 AB-split-2-disjoint by auto
  finally show ?thesis
  using 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.

```

context stone-relation-algebra
begin

```

```

abbreviation rectangle :: 'a ⇒ bool where rectangle x ≡ x * top * x ≤ x
abbreviation simple :: 'a ⇒ bool where simple x ≡ top * x * top = top

```

```

lemma rectangle-eq:
  rectangle x ⟷ x * top * x = x
  by (simp add: order.eq-iff ex231d)

```

```

lemma arc-univalent-injective-rectangle-simple:
  arc a ⟷ univalent a ∧ injective a ∧ rectangle a ∧ 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:

```

```

atom x  $\implies$  atom (xT)
by (metis conv-involutive conv-isotone symmetric-bot-closed)

lemma conv-atom-iff:
atom x  $\longleftrightarrow$  atom (xT)
by (metis conv-atom conv-involutive)

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

lemma AB-card-4-1:
a  $\leq$  x  $\wedge$  a  $\leq$  y  $\longleftrightarrow$  a  $\leq$  x  $\sqcup$  y  $\wedge$  a  $\leq$  x  $\sqcap$  y
using le-supI1 by auto

lemma AB-card-4-2:
assumes atom a
shows (a  $\leq$  x  $\wedge$   $\neg$  a  $\leq$  y)  $\vee$  ( $\neg$  a  $\leq$  x  $\wedge$  a  $\leq$  y)  $\longleftrightarrow$  a  $\leq$  x  $\sqcup$  y  $\wedge$   $\neg$  a  $\leq$  x  $\sqcap$  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  $\wedge$   $\neg$  a  $\leq$  y  $\longleftrightarrow$   $\neg$  a  $\leq$  x  $\sqcup$  y  $\wedge$   $\neg$  a  $\leq$  x  $\sqcap$  y
using assms AB-card-4-2 by auto

lemma AB-card-5-1:
assumes atom a
and a  $\leq$  xT * y  $\sqcap$  z
shows x * a  $\sqcap$  y  $\leq$  x * z  $\sqcap$  y
and x * a  $\sqcap$  y  $\neq$  bot
proof -
show x * a  $\sqcap$  y  $\leq$  x * z  $\sqcap$  y
using assms(2) comp-inf.mult-left-isotone mult-right-isotone by auto
show x * a  $\sqcap$  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  $\leq$  xT * y  $\sqcap$  z

```

and $a \neq b$
shows $(x * a \sqcap y) \sqcap (x * b \sqcap y) = \text{bot}$
and $x * a \sqcap y \neq x * b \sqcap y$
proof –
show $(x * a \sqcap y) \sqcap (x * b \sqcap y) = \text{bot}$
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 $x * a \sqcap y \neq x * b \sqcap y$
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 $a * \text{top} \sqcap b * \text{top} = \text{bot}$
proof –
have $a^T * b \leq 1$
by (*meson* *assms*(1,3,5) *comp-isotone conv-isotone dual-order.trans*)
hence $a * \text{top} \sqcap b = \text{bot}$
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 $a \leq x \sqcap y * z^T$
shows $a * z \sqcap y \leq x * z \sqcap y$
and $a * z \sqcap y \neq \text{bot}$
proof –
show $a * z \sqcap y \leq x * z \sqcap y$
using *assms*(2) *inf.sup-left-isotone mult-left-isotone* **by** *auto*
show $a * z \sqcap y \neq \text{bot}$
by (*metis* *assms* *inf.absorb2 inf.boundedE schroeder-2*)
qed

lemma *AB-card-6-2*:

assumes *univalent* x
and *atom* a
and $a \leq x \sqcap y * z^T$
and *atom* b
and $b \leq x \sqcap y * z^T$
and $a \neq b$

shows $(a * z \sqcap y) \sqcap (b * z \sqcap y) = \text{bot}$
and $a * z \sqcap y \neq b * z \sqcap y$
proof –
have $(a * z \sqcap y) \sqcap (b * z \sqcap y) \leq a * \text{top} \sqcap b * \text{top}$
by (*meson comp-inf.comp-isotone comp-inf.ex231d inf.boundedE mult-right-isotone*)
also have $\dots = \text{bot}$
using *AB-card-6-0 assms* **by force**
finally show $(a * z \sqcap y) \sqcap (b * z \sqcap y) = \text{bot}$
using *le-bot* **by blast**
thus $a * z \sqcap y \neq b * z \sqcap y$
using *AB-card-6-1(2) assms(4,5)* **by fastforce**
qed

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

lemma *domain-atom*:
assumes *atom a*
shows *atom (a * top ⊓ 1)*
proof
show $a * \text{top} \sqcap 1 \neq \text{bot}$
by (*metis assms domain-vector-conv ex231a inf-vector-comp mult-left-zero vector-export-comp-unit*)


```

next
show  $\forall y. y \neq \text{bot} \wedge y \leq a * \text{top} \sqcap 1 \longrightarrow y = a * \text{top} \sqcap 1$ 
proof (rule allI, rule impI)
  fix y
  assume 1:  $y \neq \text{bot} \wedge y \leq a * \text{top} \sqcap 1$ 
  hence 2:  $y = 1 \sqcap y * a * \text{top}$ 
  using dedekind-injective comp-associative coreflexive-idempotent
  coreflexive-symmetric inf.absorb2 inf.sup-monoid.add-commute by auto
  hence  $y * a \neq \text{bot}$ 
  using 1 comp-inf.semiring.mult-zero-right vector-bot-closed by force
  hence  $a = y * a$ 
  using 1 by (metis assms comp-right-one coreflexive-comp-top-inf
  inf.boundedE mult-sub-right-one)
  thus  $y = a * \text{top} \sqcap 1$ 
  using 2 inf.sup-monoid.add-commute by auto
qed
qed

```

```

lemma codomain-atom:
  assumes atom a
  shows atom (top * a  $\sqcap$  1)
proof -
  have top * a  $\sqcap$  1 =  $a^T * \text{top} \sqcap 1$ 
  by (simp add: domain-vector-covector inf.sup-monoid.add-commute)
  thus ?thesis
  using domain-atom conv-atom assms by auto
qed

```

```

lemma atom-rectangle-atom-one-rep:
  ( $\forall a. \text{atom } a \longrightarrow a * \text{top} * a \leq a$ )  $\longleftrightarrow$  ( $\forall a. \text{atom } a \wedge a \leq 1 \longrightarrow a * \text{top} * a \leq 1$ )
proof
  assume  $\forall a. \text{atom } a \longrightarrow a * \text{top} * a \leq a$ 
  thus  $\forall a. \text{atom } a \wedge a \leq 1 \longrightarrow a * \text{top} * a \leq 1$ 
  by auto
next
assume 1:  $\forall a. \text{atom } a \wedge a \leq 1 \longrightarrow a * \text{top} * a \leq 1$ 
show  $\forall a. \text{atom } a \longrightarrow a * \text{top} * a \leq a$ 
proof (rule allI, rule impI)
  fix a
  assume atom a
  hence atom (a * top  $\sqcap$  1)
  by (simp add: domain-atom)
  hence (a * top  $\sqcap$  1) * top * (a * top  $\sqcap$  1)  $\leq 1$ 
  using 1 by simp
  hence a * top *  $a^T \leq 1$ 
  by (smt comp-associative conv-dist-comp coreflexive-symmetric ex231e
  inf-top.right-neutral symmetric-top-closed vector-export-comp-unit)
  thus a * top * 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-card-2-1*:

assumes $a * top * a \leq a$
shows $(a * top \sqcap 1) * top * (top * a \sqcap 1) = a$
by (*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*)

lemma *atomsimple-atom1simple*:

$(\forall a . atom\ a \longrightarrow top * a * top = top) \longleftrightarrow (\forall a . atom\ a \wedge a \leq 1 \longrightarrow top * a * top = top)$

proof

assume $\forall a . atom\ a \longrightarrow top * a * top = top$
thus $\forall a . atom\ a \wedge a \leq 1 \longrightarrow top * a * top = top$
by *simp*

next

assume $1: \forall a . atom\ a \wedge a \leq 1 \longrightarrow top * a * top = top$

show $\forall a . atom\ a \longrightarrow top * a * top = top$

proof (*rule allI, rule impI*)

fix a

assume $atom\ a$

hence $2: atom\ (a * top \sqcap 1)$

by (*simp add: domain-atom*)

have $top * (a * top \sqcap 1) * top = top * a * top$

using *comp-associative vector-export-comp-unit* by *auto*

thus $top * a * top = top$

using $1\ 2$ by *auto*

qed

qed

lemma *AB-card-2-2*:

assumes $atom\ a$

and $a \leq 1$

and $atom\ b$

and $b \leq 1$

and $\forall a . atom\ a \longrightarrow top * a * top = top$

shows $a * top * b * top \sqcap 1 = a$ and $top * a * top * b \sqcap 1 = b$

proof –

show $a * top * b * top \sqcap 1 = a$

using *assms(2,3,5) comp-associative coreflexive-comp-top-inf-one* by *auto*

show $top * a * top * b \sqcap 1 = b$

using *assms(1,4,5) epm-3 inf.sup-monoid.add-commute* by *auto*

qed

```

abbreviation dom-cod :: 'a ⇒ 'a × 'a
  where dom-cod a ≡ (a * top ⊔ 1, top * a ⊔ 1)

lemma dom-cod-atoms-1:
  dom-cod ' AB top ⊆ AB 1 × AB 1
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

class stone-relation-algebra-simple = stone-relation-algebra +
  assumes simple: x ≠ bot ⟶ simple x
begin

lemma point-ideal-point:
  point x ⟷ ideal-point x
  using simple by fastforce

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: $\forall a . \text{atom } a \longrightarrow \text{top} * a * \text{top} = \text{top}$
show $\forall a . a \neq \text{bot} \longrightarrow \text{top} * a * \text{top} = \text{top}$
proof (*rule allI, rule impI*)
 fix a
 assume $a \neq \text{bot}$
 from *this atomic* **obtain** b **where** 2: $\text{atom } b \wedge b \leq a$
 by *auto*
 hence $\text{top} * b * \text{top} = \text{top}$
 using 1 **by** *auto*
 thus $\text{top} * a * \text{top} = \text{top}$
 using 2 **by** (*metis order.antisym mult-left-isotone mult-right-isotone top.extremum*)
qed
qed

lemma *AB-card-2-3*:
assumes $a \neq \text{bot}$
 and $a \leq 1$
 and $b \neq \text{bot}$
 and $b \leq 1$
 and $\forall a . a \neq \text{bot} \longrightarrow \text{top} * a * \text{top} = \text{top}$
shows $a * \text{top} * b * \text{top} \sqcap 1 = a$ **and** $\text{top} * a * \text{top} * b \sqcap 1 = b$
proof –
 show $a * \text{top} * b * \text{top} \sqcap 1 = a$
 using *assms(2,3,5) comp-associative coreflexive-comp-top-inf-one* **by** *auto*
 show $\text{top} * a * \text{top} * b \sqcap 1 = b$
 using *assms(1,4,5) epm-3 inf.sup-monoid.add-commute* **by** *auto*
qed

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

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

lemma *nAB-bot-iff*:
 $x = \text{bot} \iff \text{nAB } x = 0$
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 $\text{nAB } (x^T * y \sqcap z) \leq \text{nAB } (x * z \sqcap y)$
proof (*unfold num-atoms-below-def, rule icard-image-part-le*)
 show $\forall a \in AB (x^T * y \sqcap z) . AB (x * a \sqcap y) \subseteq AB (x * z \sqcap y)$

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

lemma *nAB-univ-meet-comp*:
assumes *univalent x*
shows $nAB(x \sqcap y * z^T) \leq nAB(x * z \sqcap y)$
proof (*unfold num-atoms-below-def, rule icard-image-part-le*)
show $\forall a \in AB(x \sqcap y * z^T) . AB(a * z \sqcap y) \subseteq AB(x * z \sqcap y)$
proof
fix a
assume $a \in AB(x \sqcap y * z^T)$
hence $a * z \sqcap y \leq x * z \sqcap y$
using *AB-card-6-1(1)* **by** *auto*
thus $AB(a * z \sqcap y) \subseteq AB(x * z \sqcap y)$
using *AB-iso* **by** *blast*
qed
next
show $\forall a \in AB(x \sqcap y * z^T) . AB(a * z \sqcap y) \neq \{\}$
proof
fix a
assume $a \in AB(x \sqcap y * z^T)$

```

    hence  $a * z \sqcap y \neq \text{bot}$ 
    using AB-card-6-1(2) by auto
    thus  $AB (a * z \sqcap y) \neq \{\}$ 
    using atomic by fastforce
  qed
next
  show  $\forall a \in AB (x \sqcap y * z^T) . \forall b \in AB (x \sqcap y * z^T) . a \neq b \longrightarrow AB (a * z \sqcap y) \cap AB (b * z \sqcap y) = \{\}$ 
  proof (intro ball, rule impI)
    fix  $a b$ 
    assume  $a \in AB (x \sqcap y * z^T) b \in AB (x \sqcap y * z^T) a \neq b$ 
    hence  $(a * z \sqcap y) \sqcap (b * z \sqcap y) = \text{bot}$ 
    using assms AB-card-6-2(1) by auto
    thus  $AB (a * z \sqcap y) \cap AB (b * z \sqcap y) = \{\}$ 
    using AB-bot AB-dist-inf by blast
  qed
qed
end

```

3.2 Atom-rectangular

```

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

```

```

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

```

```

lemma AB-card-2-4:
  assumes atom a
  shows  $(a * \text{top} \sqcap 1) * \text{top} * (\text{top} * a \sqcap 1) = a$ 
  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  $x \leq a * \text{top} * b$ 
  shows  $x = a * \text{top} * b$ 

```

```

proof -
  have 1:  $x * \text{top} \sqcap 1 \neq \text{bot}$ 
    by (metis assms(5) inf-top-right le-bot top-right-mult-increasing vector-bot-closed vector-export-comp-unit)
  have  $x * \text{top} \sqcap 1 \leq a * \text{top} * b * \text{top} \sqcap 1$ 
    using assms(6) comp-inf.comp-isotone comp-isotone by blast

```

also have $\dots \leq a * top \sqcap 1$
by (*metis comp-associative comp-inf.mult-right-isotone*
inf.sup-monoid.add-commute mult-right-isotone top.extremum)
also have $\dots = a$
by (*simp add: assms(2) coreflexive-comp-top-inf-one*)
finally have $2: x * top \sqcap 1 = a$
using 1 **by** (*simp add: assms(1) domain-atom*)
have $3: top * x \sqcap 1 \neq bot$
using 1 **by** (*metis schroeder-1 schroeder-2 surjective-one-closed*
symmetric-top-closed total-one-closed)
have $top * x \sqcap 1 \leq top * a * top * b \sqcap 1$
by (*metis assms(6) comp-associative comp-inf.comp-isotone mult-right-isotone*
reflexive-one-closed)
also have $\dots \leq top * b \sqcap 1$
using *inf.sup-mono mult-left-isotone top-greatest* **by** *blast*
also have $\dots = b$
using *assms(4) epm-3 inf.sup-monoid.add-commute* **by** *auto*
finally have $top * x \sqcap 1 = b$
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 $\dots = a * top * b * top * (x \sqcap a * top * b)$
using *assms(6) calculation inf-absorb1* **by** *auto*
also have $\dots \leq a * top * (x \sqcap a * top * b)$
by (*metis comp-associative comp-inf-covector inf.idem inf.order-iff*
mult-right-isotone)
also have $\dots \leq a * top * (x \sqcap a * top)$
using *comp-associative comp-inf.mult-right-isotone mult-right-isotone* **by** *auto*
also have $\dots = a * top * a^T * x$
by (*metis comp-associative comp-inf-vector inf-top.left-neutral*)
also have $\dots = a * top * a * x$
by (*simp add: assms(2) coreflexive-symmetric*)
also have $\dots = a * x$
by (*simp add: assms(1) atomrect-eq*)
also have $\dots \leq x$
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 $1: a \in AB \text{ top } b \in AB \text{ top } dom-cod a = dom-cod b$

have $a = a * top * a$

using 1 *atomrect-eq* **by** *auto*

also have ... = $(a * top \sqcap 1) * top * (top * a \sqcap 1)$
using *calculation AB-card-2-1* **by** *auto*
also have ... = $(b * top \sqcap 1) * top * (top * b \sqcap 1)$
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*:

$finite (AB top) \longleftrightarrow finite (AB 1)$

proof

have $AB 1 \subseteq AB top$

by *auto*

thus $finite (AB top) \implies finite (AB 1)$

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 $dom-cod \text{ ' } AB top \subseteq AB 1 \times AB 1$

using *dom-cod-atoms-1* **by** *blast*

show $finite (AB 1 \times AB 1)$

using *1* **by** *blast*

qed

qed

lemma *nAB-top-1*:

$nAB top \leq nAB 1 * nAB 1$

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 $dom-cod \text{ ' } AB top \subseteq AB 1 \times AB 1$

using *dom-cod-atoms-1* **by** *blast*

qed

lemma *atom-vector-injective*:

assumes *atom x*

shows $injective (x * top)$

proof –

have $atom (x * top \sqcap 1)$

by (*simp add: assms domain-atom*)

hence $(x * top \sqcap 1) * top * (x * top \sqcap 1) \leq 1$
using *atom-rectangle-atom-one-rep atomrect* **by** *auto*
hence $x * top * x^T \leq 1$
by (*smt comp-associative conv-dist-comp coreflexive-symmetric ex231e*
inf-top.right-neutral symmetric-top-closed vector-export-comp-unit)
thus *injective* $(x * top)$
by (*metis comp-associative conv-dist-comp symmetric-top-closed*
vector-top-closed)
qed

lemma *atom-injective*:
atom $x \implies$ *injective* x
by (*metis atom-vector-injective comp-associative conv-dist-comp*
dual-order.trans mult-right-isotone symmetric-top-closed top-left-mult-increasing)

lemma *atom-covector-univalent*:
atom $x \implies$ *univalent* $(top * x)$
by (*metis comp-associative conv-involutive atom-vector-injective conv-atom-iff*
conv-dist-comp symmetric-top-closed)

lemma *atom-univalent*:
atom $x \implies$ *univalent* x
using *atom-injective conv-atom-iff univalent-conv-injective* **by** *blast*

lemma *counterexample-atom-simple*:
atom $x \implies$ *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 $x = x * top * x^T$
using *assms atomrect-eq* **by** *auto*
also have $\dots \leq 1$
by (*metis assms(1) atom-vector-injective conv-dist-comp*
equivalence-top-closed ideal-top-closed mult-assoc)
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  $x \neq \text{bot}$ 
    and  $x \leq 1$ 
  shows  $\exists a . a \neq \text{bot} \wedge a * \text{top} * a \leq 1 \wedge a \leq x$ 
proof -
  from atomic obtain a where 1:  $\text{atom } a \wedge a \leq x$ 
  using assms(1) by auto
  hence  $a * \text{top} * a \leq a$ 
  by (simp add: atomrect)
  also have  $\dots \leq 1$ 
  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:  $\text{atom } a \longrightarrow \text{simple } a$ 
begin

```

```

lemma AB-card-2-5:
  assumes atom a
    and  $a \leq 1$ 
    and atom b
    and  $b \leq 1$ 
  shows  $a * \text{top} * b * \text{top} \sqcap 1 = a$  and  $\text{top} * a * \text{top} * b \sqcap 1 = b$ 
  using assms AB-card-2-2 atomsimple by auto

```

```

lemma simple-atom-1:
   $\text{atom } a \implies \text{atom } b \implies a * \text{top} * b \neq \text{bot}$ 
  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

```

```

subclass stone-relation-algebra-simple
  apply unfold-locales
  using atomsimple atomsimple-simple by blast

```

lemma *AB-card-2-6*:

assumes $a \neq \text{bot}$

and $a \leq 1$

and $b \neq \text{bot}$

and $b \leq 1$

shows $a * \text{top} * b * \text{top} \sqcap 1 = a$ **and** $\text{top} * a * \text{top} * b \sqcap 1 = b$

using *assms AB-card-2-3 simple atomsimple-simple* **by** *auto*

lemma *dom-cod-atoms-2*:

$AB\ 1 \times AB\ 1 \subseteq \text{dom-cod} \text{ ' } AB\ \text{top}$

proof

fix x

assume $x \in AB\ 1 \times AB\ 1$

from *this* **obtain** $a\ b$ **where** $1: \text{atom } a \wedge a \leq 1 \wedge \text{atom } b \wedge b \leq 1 \wedge x = (a, b)$

by *auto*

hence $a * \text{top} * b \neq \text{bot}$

by (*simp add: simple-atom-1*)

from *this* **obtain** c **where** $2: \text{atom } c \wedge c \leq a * \text{top} * b$

using *atomic* **by** *blast*

hence $c * \text{top} \sqcap 1 \leq a * \text{top} \sqcap 1$

by (*smt comp-inf.comp-isotone inf.boundedE inf.orderE inf-vector-comp*

reflexive-one-closed top-right-mult-increasing)

also **have** $\dots = a$

using 1 **by** (*simp add: coreflexive-comp-top-inf-one*)

finally **have** $3: c * \text{top} \sqcap 1 = a$

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 $\text{top} * c \sqcap 1 \leq b$

using 1 **by** (*smt epm-3 inf.cobounded1 inf.left-commute inf.orderE*

injective-one-closed reflexive-one-closed)

hence $\text{top} * c \sqcap 1 = b$

using $1\ 2$ *codomain-atom* **by** *simp*

hence $\text{dom-cod } c = x$

using $1\ 3$ **by** *simp*

thus $x \in \text{dom-cod} \text{ ' } AB\ \text{top}$

using 2 **by** *auto*

qed

lemma *dom-cod-atoms*:

$AB\ 1 \times AB\ 1 = \text{dom-cod} \text{ ' } AB\ \text{top}$

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 ≤ 1
    and atom b
    and b ≤ 1
  shows atom (a * top * b)
  using assms simple-atom-1 simple-atom-2 by auto
```

lemma *nAB-top-2*:

```
nAB 1 * nAB 1 ≤ nAB top
```

proof (*unfold num-atoms-below-def icard-cartesian-product[THEN sym], rule icard-inj-on-le*)

```
  let ?f = λ(a,b) . a * top * b
```

```
  show inj-on ?f (AB 1 × AB 1)
```

proof

```
  fix x y
```

```
  assume x ∈ AB 1 × AB 1 y ∈ AB 1 × AB 1
```

```
  from this obtain a b c d where 1: atom a ∧ a ≤ 1 ∧ atom b ∧ b ≤ 1 ∧ x =
(a,b) ∧ atom c ∧ c ≤ 1 ∧ atom d ∧ d ≤ 1 ∧ y = (c,d)
```

```
  by auto
```

```
  assume ?f x = ?f y
```

```
  hence 2: a * top * b = c * top * d
```

```
  using 1 by auto
```

```
  hence 3: a = c
```

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

```
  have b = d
```

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

```
  thus x = y
```

```
  using 1 3 by simp
```

qed

```
show ?f ‘ (AB 1 × AB 1) ⊆ AB top
```

proof

```
  fix x
```

```
  assume x ∈ ?f ‘ (AB 1 × AB 1)
```

```
  from this obtain a b where 4: atom a ∧ a ≤ 1 ∧ atom b ∧ b ≤ 1 ∧ x = a *
top * b
```

```
  by auto
```

```
  hence a * top * b ∈ AB top
```

```
  using simple-atom by simp
```

```
  thus x ∈ AB top
```

```
  using 4 by simp
```

qed

qed

```

lemma nAB-top:
  nAB 1 * nAB 1 = nAB top
  using nAB-top-1 nAB-top-2 by auto

lemma atom-covector-mapping:
  atom a  $\implies$  mapping (top * a)
  using atom-covector-univalent atomsimple by blast

lemma atom-covector-regular:
  atom a  $\implies$  regular (top * a)
  by (simp add: atom-covector-mapping mapping-regular)

lemma atom-vector-bijective:
  atom a  $\implies$  bijective (a * top)
  using atom-vector-injective comp-associative atomsimple by auto

lemma atom-vector-regular:
  atom a  $\implies$  regular (a * top)
  by (simp add: atom-vector-bijective bijective-regular)

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

lemma atom-regular:
  atom a  $\implies$  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  $2: atom\ b \wedge b \leq a$ 
  by auto
hence  $3: AB\ (a \sqcap b) = \{b\}$ 
  by fastforce
have  $AB\ (a \sqcap b) \cup AB\ (a \sqcap -b) = AB\ a \wedge AB\ (a \sqcap b) \cap AB\ (a \sqcap -b) = \{\}$ 
  using AB-split-2 AB-split-2-disjoint by simp
hence  $\{b\} \cup AB\ (a \sqcap -b) = \{b\} \wedge \{b\} \cap AB\ (a \sqcap -b) = \{\}$ 
  using  $1\ 3$  by simp
hence  $AB\ (a \sqcap -b) = \{\}$ 
  by auto
hence  $a \sqcap -b = bot$ 
  using AB-nonempty-iff by blast
hence  $a \leq b$ 
  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  $\bigwedge p . ideal-point\ p \implies p = bot$ 

```

```

    using le-bot top.extremum by blast
  hence { p . ideal-point p } ⊆ {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 ∀ x ∈ ?p . AB x ⊆ AB top
    using top.extremum by auto
  have ∀ x ∈ ?p . x ≠ bot
    using False by auto
  thus ∀ x ∈ ?p . AB x ≠ {}
    using AB-nonempty by auto
  show ∀ x ∈ ?p . ∀ y ∈ ?p . x ≠ y → AB x ∩ AB y = {}
  proof (intro ballI, rule impI, 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*:

$atom\ x \longleftrightarrow nAB\ x = 1$

nitpick[*expect=genuine,card=3*]

oops

lemma *counterexample-nAB-top-iff-eq*:

$nAB\ x = nAB\ top \longleftrightarrow x = top$

nitpick[*expect=genuine,card=3*]

oops

lemma *counterexample-nAB-top-iff-leq*:

$nAB\ top \leq nAB\ x \longleftrightarrow x = top$

nitpick[*expect=genuine,card=3*]

oops

end

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*:

$nAB\ 1 * nAB\ 1 \leq nAB\ top$

proof (*unfold num-atoms-below-def icard-cartesian-product[THEN sym], rule surj-icard-le*)

show $AB\ 1 \times AB\ 1 \subseteq dom-cod\ 'AB\ top$

using *dom-cod-atoms-2* **by** *blast*

qed

lemma *counterexample-nAB-atom-iff-2*:

$atom\ x \longleftrightarrow nAB\ x = 1$

nitpick[*expect=genuine,card=6*]

oops

lemma *counterexample-nAB-top-iff-eq-2*:

$nAB\ x = nAB\ top \longleftrightarrow x = top$


```

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

```

```

lemma counterexample-nAB-atom-top-iff-leq-2:
  (atom x ↔ nAB x = 1) ∨ (nAB y = nAB top ↔ y = top) ∨ (nAB top ≤
nAB y ↔ y = 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
begin

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:  $AB\ x \neq \{\}$ 
  using AB-nonempty by blast
have 2: finite ( $AB\ x$ )
  using finite-AB by blast
have 3: regular (Sup-fin ( $AB\ x$ ))
proof (rule finite-ne-subset-induct')
  show finite ( $AB\ x$ )
    using 2 by simp
  show  $AB\ x \neq \{\}$ 
    using 1 by simp
  show  $AB\ x \subseteq AB\ top$ 
    by auto
  show  $\bigwedge a . a \in AB\ top \implies Sup-fin\ \{a\} = --Sup-fin\ \{a\}$ 
    using atom-regular by auto
  show  $\bigwedge a\ F . finite\ F \implies F \neq \{\} \implies F \subseteq AB\ top \implies a \in AB\ top \implies a \notin F$ 
 $\implies Sup-fin\ F = --Sup-fin\ F \implies Sup-fin\ (insert\ a\ F) = --Sup-fin\ (insert\ a\ F)$ 
    using atom-regular by auto
qed
have  $x \sqcap -Sup-fin\ (AB\ x) = bot$ 
proof (rule ccontr)
  assume  $x \sqcap -Sup-fin\ (AB\ x) \neq bot$ 
  from this obtain b where 4:  $atom\ b \wedge b \leq x \sqcap -Sup-fin\ (AB\ x)$ 
    using atomic by blast
  hence  $b \leq Sup-fin\ (AB\ x)$ 
    using Sup-fin.coboundedI 2 by force
  thus False
    using 4 atom-in-p-xor by auto
qed
hence 5:  $x \leq Sup-fin\ (AB\ x)$ 
  using 3 by (simp add: pseudo-complement)
have  $Sup-fin\ (AB\ x) \leq x$ 
  using 1 2 Sup-fin.boundedI by fastforce
thus ?thesis
  using 3 5 order.antisym by force
qed

sublocale ra: relation-algebra where minus =  $\lambda x\ y . x \sqcap - y$ 
proof
  show  $\bigwedge x . x \sqcap - x = bot$ 
    by simp
  show  $\bigwedge x . x \sqcup - x = top$ 
    using all-regular pp-sup-p by fast
  show  $\bigwedge x\ y . x \sqcap - y = x \sqcap - y$ 
    by simp
qed

end

```

```

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

subclass stone-relation-algebra-atomic-finiteatoms
proof
  show finite { a . atom a }
  by simp
  show  $\bigwedge x. x \neq \text{bot} \longrightarrow (\exists a. \text{atom } a \wedge a \leq x)$ 
  proof
    fix x
    assume 1:  $x \neq \text{bot}$ 
    let ?s = { y . y ≤ x ∧ y ≠ bot }
    have 2: finite ?s
    by auto
    have 3: ?s ≠ {}
    using 1 by blast
    from ne-finite-has-minimal obtain m where  $m \in ?s \wedge (\forall x \in ?s. x \leq m \longrightarrow x = m)$ 
    using 2 3 by meson
    hence atom m ∧ m ≤ x
    using order-trans by blast
    thus  $\exists a. \text{atom } a \wedge a \leq x$ 
    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*:

$\text{atom } a \longleftrightarrow nAB \ a = 1$

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: $\text{atom } b \wedge b \leq a$

by auto

hence 3: $AB \ (a \sqcap b) = \{b\}$

by fastforce

have $AB \ (a \sqcap b) \cup AB \ (a \sqcap -b) = AB \ a \wedge AB \ (a \sqcap b) \cap AB \ (a \sqcap -b) = \{ \}$

```

    using AB-split-2 AB-split-2-disjoint by simp
  hence  $\{b\} \cup AB (a \sqcap -b) = \{b\} \wedge \{b\} \cap AB (a \sqcap -b) = \{\}$ 
    using 1 3 by simp
  hence  $AB (a \sqcap -b) = \{\}$ 
    by auto
  hence  $a \sqcap -b = bot$ 
    using AB-nonempty-iff by blast
  hence  $a \leq b$ 
    by (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

```

Sup-fin only works for non-empty finite sets.

```

lemma atomistic:
  assumes  $x \neq bot$ 
    shows  $x = Sup-fin (AB x)$ 
proof (rule order.antisym)
  show  $x \leq Sup-fin (AB x)$ 
  proof (rule ccontr)
    assume  $\neg x \leq Sup-fin (AB x)$ 
    hence  $x \sqcap -Sup-fin (AB x) \neq bot$ 
      using shunting-1 by blast
    from this obtain a where 1:  $atom a \wedge a \leq x \sqcap -Sup-fin (AB x)$ 
      using atomic by blast
    hence  $a \in AB x$ 
      by simp
    hence  $a \leq Sup-fin (AB x)$ 
      using Sup-fin.coboundedI finite-AB by auto
    thus False
      using 1 atom-in-p-xor by auto
  qed
  show  $Sup-fin (AB x) \leq x$ 
  proof (rule Sup-fin.boundedI)
    show finite (AB x)
      using finite-AB by auto
    show  $AB x \neq \{\}$ 
      using assms atomic by blast
    show  $\bigwedge a. a \in AB x \implies a \leq x$ 
      by auto
  qed
qed

```

```

lemma counterexample-nAB-top:
   $1 \neq \text{top} \implies nAB \text{ top} = nAB \ 1 * nAB \ 1$ 
  nitpick[expect=genuine,card=4]
  oops

end

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

lemma counterexample-atom-rectangle:
   $\text{atom } x \longrightarrow \text{rectangle } x$ 
  nitpick[expect=genuine,card=4]
  oops

lemma counterexample-atom-univalent:
   $\text{atom } x \longrightarrow \text{univalent } x$ 
  nitpick[expect=genuine,card=4]
  oops

lemma counterexample-point-dense:
  assumes  $x \neq \text{bot}$ 
  and  $x \leq 1$ 
  shows  $\exists a . a \neq \text{bot} \wedge a * \text{top} * a \leq 1 \wedge a \leq x$ 
  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  $\Rightarrow$  enat (#- [100] 100)

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

abbreviation card-bot :: 'a  $\Rightarrow$  bool where card-bot -  $\equiv$  #bot
= 0

```

abbreviation *card-bot-iff* :: 'a ⇒ bool **where** *card-bot-iff* - ≡
 $\forall x::'a . \#x = 0 \longleftrightarrow x = \text{bot}$
abbreviation *card-top* :: 'a ⇒ bool **where** *card-top* - ≡
 $\#top = \#1 * \#1$
abbreviation *card-conv* :: 'a ⇒ bool **where** *card-conv* - ≡
 $\forall x::'a . \#(x^T) = \#x$
abbreviation *card-add* :: 'a ⇒ bool **where** *card-add* - ≡ $\forall x$
 $y::'a . \#x + \#y = \#(x \sqcup y) + \#(x \sqcap y)$
abbreviation *card-iso* :: 'a ⇒ bool **where** *card-iso* - ≡ $\forall x$
 $y::'a . x \leq y \longrightarrow \#x \leq \#y$
abbreviation *card-univ-comp-meet* :: 'a ⇒ bool **where** *card-univ-comp-meet* - ≡
 $\equiv \forall x y z::'a . \text{univalent } x \longrightarrow \#(x^T * y \sqcap z) \leq \#(x * z \sqcap y)$
abbreviation *card-univ-meet-comp* :: 'a ⇒ bool **where** *card-univ-meet-comp* - ≡
 $\equiv \forall x y z::'a . \text{univalent } x \longrightarrow \#(x \sqcap y * z^T) \leq \#(x * z \sqcap y)$
abbreviation *card-comp-univ* :: 'a ⇒ bool **where** *card-comp-univ* - ≡
 $\forall x y::'a . \text{univalent } x \longrightarrow \#(y * x) \leq \#y$
abbreviation *card-univ-meet-vector* :: 'a ⇒ bool **where** *card-univ-meet-vector* - ≡
 $\equiv \forall x y::'a . \text{univalent } x \longrightarrow \#(x \sqcap y * \text{top}) \leq \#y$
abbreviation *card-univ-meet-conv* :: 'a ⇒ bool **where** *card-univ-meet-conv* - ≡
 $\equiv \forall x y::'a . \text{univalent } x \longrightarrow \#(x \sqcap y * y^T) \leq \#y$
abbreviation *card-domain-sym* :: 'a ⇒ bool **where** *card-domain-sym* - ≡
 $\equiv \forall x::'a . \#(1 \sqcap x * x^T) \leq \#x$
abbreviation *card-domain-sym-conv* :: 'a ⇒ bool **where** *card-domain-sym-conv* - ≡
 $- \equiv \forall x::'a . \#(1 \sqcap x^T * x) \leq \#x$
abbreviation *card-domain* :: 'a ⇒ bool **where** *card-domain* - ≡
 $\forall x::'a . \#(1 \sqcap x * \text{top}) \leq \#x$
abbreviation *card-domain-conv* :: 'a ⇒ bool **where** *card-domain-conv* - ≡
 $\equiv \forall x::'a . \#(1 \sqcap x^T * \text{top}) \leq \#x$
abbreviation *card-codomain* :: 'a ⇒ bool **where** *card-codomain* - ≡
 $\forall x::'a . \#(1 \sqcap \text{top} * x) \leq \#x$
abbreviation *card-codomain-conv* :: 'a ⇒ bool **where** *card-codomain-conv* - ≡
 $\equiv \forall x::'a . \#(1 \sqcap \text{top} * x^T) \leq \#x$
abbreviation *card-univ* :: 'a ⇒ bool **where** *card-univ* - ≡
 $\forall x::'a . \text{univalent } x \longrightarrow \#x \leq \#(x * \text{top})$
abbreviation *card-atom* :: 'a ⇒ bool **where** *card-atom* - ≡
 $\forall x::'a . \text{atom } x \longrightarrow \#x = 1$
abbreviation *card-atom-iff* :: 'a ⇒ bool **where** *card-atom-iff* - ≡
 $\forall x::'a . \text{atom } x \longleftrightarrow \#x = 1$
abbreviation *card-top-iff-eq* :: 'a ⇒ bool **where** *card-top-iff-eq* - ≡
 $\forall x::'a . \#x = \#top \longleftrightarrow x = \text{top}$
abbreviation *card-top-iff-leq* :: 'a ⇒ bool **where** *card-top-iff-leq* - ≡
 $\forall x::'a . \#top \leq \#x \longleftrightarrow x = \text{top}$
abbreviation *card-top-finite* :: 'a ⇒ bool **where** *card-top-finite* - ≡
 $\#top \neq \infty$

lemma *card-domain-iff*:
card-domain - \longleftrightarrow *card-domain-sym* -
by (*simp add: domain-vector-conv*)

lemma *card-codomain-conv-iff*:
card-codomain-conv - \longleftrightarrow *card-domain* -
by (*simp add: domain-vector-covector*)

lemma *card-codomain-iff*:
assumes *card-conv*: *card-conv* -
shows *card-codomain* - \longleftrightarrow *card-codomain-conv* -
by (*metis card-conv conv-involutive*)

lemma *card-domain-conv-iff*:
card-codomain - \longleftrightarrow *card-domain-conv* -
using *domain-vector-covector* **by** *auto*

lemma *card-domain-sym-conv-iff*:
card-domain-conv - \longleftrightarrow *card-domain-sym-conv* -
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 $\#(x^T * y \sqcap z) = \#(y^T * x \sqcap z^T)$
by (*metis card-conv conv-dist-comp conv-dist-inf conv-involutive*)
also have $\dots = \#((y^T \sqcap z^T * x^T) * x)$
using *1* **by** (*simp add: dedekind-univalent*)
also have $\dots \leq \#(y^T \sqcap z^T * x^T)$
using *1* *card-comp-univ* **by** *blast*
also have $\dots = \#(x * z \sqcap y)$
by (*metis card-conv conv-dist-comp conv-dist-inf inf.sup-monoid.add-commute*)
finally show $\#(x^T * y \sqcap z) \leq \#(x * z \sqcap y)$
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 $x \sqcap y = \text{bot}$
shows $\#(x \sqcup y) = \#x + \#y$
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 $A \neq \{\}$
and *finite* A
and $\forall x \in A . \forall y \in A . x \neq y \longrightarrow x \sqcap y = \text{bot}$
shows $\# \text{Sup-fin } A = \text{sum cardinality } A$

proof (*rule finite-ne-subset-induct'*)

show *finite* A

using *assms(4)* **by** *simp*

show $A \neq \{\}$

using *assms(3)* **by** *simp*

show $A \subseteq A$

by *simp*

show $\bigwedge x . x \in A \implies \# \text{Sup-fin } \{x\} = \text{sum cardinality } \{x\}$

by *auto*

fix $x F$

assume *1*: *finite* F $F \neq \{\}$ $F \subseteq A$ $x \in A$ $x \notin F$ $\# \text{Sup-fin } F = \text{sum cardinality } F$

have $\# \text{Sup-fin } (\text{insert } x F) = \#(x \sqcup \text{Sup-fin } F)$

using *1* **by** *simp*

also have $\dots = \#x + \# \text{Sup-fin } F$

proof -

have $x \sqcap \text{Sup-fin } F = \text{Sup-fin } \{ x \sqcap y \mid y . y \in F \}$

using *1 inf-Sup1-distrib* **by** *simp*

also have $\dots = \text{Sup-fin } \{ \text{bot} \mid y . y \in F \}$

using *1 assms(5)* **by** (*metis (mono-tags, opaque-lifting) subset-iff*)

also have $\dots \leq \text{bot}$

by (*rule Sup-fin.boundedI, simp-all add: 1*)

finally have $x \sqcap \text{Sup-fin } F = \text{bot}$

by (*simp add: order.antisym*)

thus *?thesis*

using *card-add-disjoint assms* **by** *auto*

qed

also have $\dots = \text{sum cardinality } (\text{insert } x F)$

using *1* **by** *simp*

finally show $\# \text{Sup-fin } (\text{insert } x F) = \text{sum cardinality } (\text{insert } x F)$

qed

lemma *card-dist-sup-atoms*:

assumes *card-bot*: *card-bot* -
and *card-add*: *card-add* -
and $A \neq \{\}$
and *finite* A
and $A \subseteq AB$ *top*

shows $\#Sup\text{-}fin\ A = sum\ cardinality\ A$
proof –
have $\forall x \in A . \forall y \in A . x \neq y \longrightarrow x \sqcap y = bot$
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 $\#x \leq \#top$
by (*simp add: card-iso*)

lemma *card-pp-increasing*:
assumes *card-iso*: *card-iso* -
shows $\#x \leq \#(-x)$
by (*simp add: card-iso pp-increasing*)

lemma *card-top-iff-eq-leq*:
assumes *card-iso*: *card-iso* -
shows *card-top-iff-eq* - \longleftrightarrow *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 $\#(y * x) = \#(x^T * y^T)$
by (*metis card-conv conv-dist-comp*)
also have $\dots = \#(top \sqcap x^T * y^T)$
by *simp*
also have $\dots \leq \#(x * top \sqcap y^T)$
using 1 **by** (*metis card-univ-comp-meet inf.sup-monoid.add-commute*)
also have $\dots \leq \#(y^T)$
using *card-iso* **by** *simp*
also have $\dots = \#y$
by (*simp add: card-conv*)
finally show $\#(y * x) \leq \#y$
qed

lemma *card-comp-univ-iff-card-univ-comp-meet*:
assumes *card-iso*: *card-iso* -
and *card-conv*: *card-conv* -
shows *card-comp-univ* - \longleftrightarrow *card-univ-comp-meet* -
using *card-iso* *card-univ-comp-meet-implies-card-comp-univ* *card-conv*
card-comp-univ-implies-card-univ-comp-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 $\#(x \sqcap y * z^T) = \#(x \sqcap (y \sqcap x * z) * (z^T \sqcap y^T * x))$
by (*metis conv-involutive dedekind-eq inf.sup-monoid.add-commute*)
also have $\dots \leq \#(x \sqcap (y \sqcap x * z) * top)$
using *card-iso* *inf.sup-right-isotone* *mult-isotone* **by** *auto*
also have $\dots \leq \#(x * z \sqcap y)$
using *1* **by** (*simp add: card-univ-meet-vector inf.sup-monoid.add-commute*)
finally show $\#(x \sqcap y * z^T) \leq \#(x * z \sqcap y)$
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 $\#(x \sqcap y * top) \leq \#(x * top \sqcap y)$
using *1* **by** (*metis card-univ-meet-comp symmetric-top-closed*)
also have $\dots \leq \#y$
using *card-iso* **by** *auto*
finally show $\#(x \sqcap y * top) \leq \#y$
qed

lemma *card-univ-meet-vector-iff-card-univ-meet-comp*:
assumes *card-iso*: *card-iso* -
shows *card-univ-meet-vector* - \longleftrightarrow *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*: *card-univ-meet-vector* -

shows *card-univ-meet-conv* -
proof (*intro allI, rule impI*)
fix $x\ y\ z$
assume 1 : *univalent x*
have $\#(x \sqcap y * y^T) \leq \#(x \sqcap y * top)$
using *card-iso comp-inf.mult-right-isotone mult-right-isotone* **by** *auto*
also have $\dots \leq \#y$
using 1 **by** (*simp add: card-univ-meet-vector*)
finally show $\#(x \sqcap y * y^T) \leq \#y$

\cdot
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 $\#(x \sqcap y * top) = \#((y * top \sqcap 1) * (x \sqcap y * top))$
by (*simp add: inf.absorb2 vector-export-comp-unit*)
also have $\dots \leq \#(y * top \sqcap 1)$
using 1 **by** (*simp add: card-comp-univ univalent-inf-closed*)
also have $\dots \leq \#y$
using *card-domain-sym card-domain-iff inf.sup-monoid.add-commute* **by** *auto*
finally show $\#(x \sqcap y * top) \leq \#y$

\cdot
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* -
using *card-iso card-comp-univ card-domain-sym-implies-card-univ-meet-vector*
card-univ-meet-vector-implies-card-univ-meet-conv
card-univ-meet-conv-implies-card-domain-sym **by** *blast*

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-conv* - \longleftrightarrow *card-univ-meet-comp* -
using *card-iso card-comp-univ card-domain-sym-implies-card-univ-meet-vector*
card-univ-meet-vector-iff-card-univ-meet-comp
card-univ-meet-vector-implies-card-univ-meet-conv univalent-one-closed **by** *blast*

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* -

using *card-iso card-comp-univ card-domain-sym-implies-card-univ-meet-vector*
card-univ-meet-conv-iff-card-univ-meet-comp
card-univ-meet-vector-iff-card-univ-meet-comp
card-univ-meet-conv-implies-card-domain-sym **by** *blast*

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 $\#x = \#(x \sqcap top * y^T)$
using *assms(4) total-conv-surjective* **by** *auto*
also have $\dots \leq \#(x * y \sqcap top)$
using *assms(3) card-univ-meet-comp* **by** *blast*
finally have $\#x \leq \#(x * y)$
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

lemma *counterexample-card-univ-comp-meet-card-comp-univ*:

assumes *card-add: card-add -*
and *card-conv: card-conv -*
and *card-bot-iff: card-bot-iff -*
and *card-atom-iff: card-atom-iff -*
and *card-univ-meet-comp: card-univ-meet-comp -*
shows *card-univ-comp-meet - \longleftrightarrow card-comp-univ -*
nitpick[*expect=genuine*]
oops

lemma *counterexample-card-univ-meet-comp-card-univ-meet-vector*:

assumes *card-add: card-add -*

```

    and card-conv: card-conv -
    and card-bot-iff: card-bot-iff -
    and card-atom-iff: card-atom-iff -
    and card-univ-comp-meet: card-univ-comp-meet -
    shows card-univ-meet-comp -  $\longleftrightarrow$  card-univ-meet-vector -
    nitpick[expect=genuine]
oops

```

```

lemma counterexample-card-univ-meet-comp-card-univ-meet-conv:
  assumes card-add: card-add -
    and card-conv: card-conv -
    and card-bot-iff: card-bot-iff -
    and card-atom-iff: card-atom-iff -
    and card-univ-comp-meet: card-univ-comp-meet -
  shows card-univ-meet-comp -  $\longleftrightarrow$  card-univ-meet-conv -
  nitpick[expect=genuine]
oops

```

```

lemma counterexample-card-univ-meet-vector-card-domain-sym:
  assumes card-add: card-add -
    and card-conv: card-conv -
    and card-bot-iff: card-bot-iff -
    and card-atom-iff: card-atom-iff -
    and card-univ-comp-meet: card-univ-comp-meet -
  shows card-univ-meet-vector -  $\longleftrightarrow$  card-domain-sym -
  nitpick[expect=genuine]
oops

```

```

lemma counterexample-card-univ-meet-conv-card-domain-sym:
  assumes card-add: card-add -
    and card-conv: card-conv -
    and card-bot-iff: card-bot-iff -
    and card-atom-iff: card-atom-iff -
    and card-univ-comp-meet: card-univ-comp-meet -
  shows card-univ-meet-conv -  $\longleftrightarrow$  card-domain-sym -
  nitpick[expect=genuine]
oops

```

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  $x \leq y$ 
  hence  $\#y = \#(x \sqcup (-x \sqcap y))$ 
    by (simp add: sup-absorb2)
  also have  $\dots = \#(x \sqcup (-x \sqcap y)) + \#(x \sqcap (-x \sqcap y))$ 
    by (simp add: card-bot)
  also have  $\dots = \#x + \#(-x \sqcap y)$ 
    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  $\#top = \#(x \sqcup -x)$ 
    by simp
  also have  $\dots = \#x + \#(-x)$ 
    using card-add card-bot-iff card-add-disjoint inf-p by blast
  also have  $\dots = \#top + \#(-x)$ 
    using 1 by simp
  finally have  $\#(-x) = 0$ 
    by (simp add: card-top-finite)
  hence  $-x = bot$ 
    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

end

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

lemma counterexample-card-nAB:
  assumes card-bot-iff: card-bot-iff -
    and card-atom-iff: card-atom-iff -

```

```

    and card-conv: card-conv -
    and card-add: card-add -
    and card-iso: card-iso -
    and card-top-iff-eq: card-top-iff-eq -
    and card-top-finite: card-top-finite -
    shows #x = nAB x
    nitpick[expect=genuine]
    oops

end

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 ≠ {}
    using False AB-nonempty-iff by blast
  have #x = #Sup-fin (AB x)
    using atomistic False by auto
  also have ... = sum cardinality (AB x)
    using 1 2 card-bot card-add card-dist-sup-disjoint different-atoms-disjoint by
force
  also have ... = sum (λx . 1) (AB x)
    using card-atom by simp
  also have ... = icard (AB x)
    by (metis (mono-tags, lifting) icard-eq-sum finite-AB)
  also have ... = nAB x
    by (simp add: num-atoms-below-def)
  finally show ?thesis
  .
qed

end

class card-ab = sra-card +
  assumes card-nAB': #x = nAB 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  $\bigwedge x . x \neq \text{bot} \longrightarrow (\exists a . \text{atom } a \wedge a \leq x)$ 
  proof
    fix x
    assume  $x \neq \text{bot}$ 
    hence  $\#x \neq 0$ 
      using card-bot-iff by auto
    hence  $nAB \ x \neq 0$ 
      by (simp add: card-nAB')
    hence  $AB \ x \neq \{\}$ 
      by (metis (mono-tags, lifting) icard-empty num-atoms-below-def)
    thus  $\exists a . \text{atom } a \wedge a \leq x$ 
      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  $\text{icard } (\text{dom-cod } 'AB \ \text{top}) = \text{icard } (AB \ 1 \times AB \ 1)$ 
    using dom-cod-atoms by auto
  also have  $\dots = \text{icard } (AB \ 1) * \text{icard } (AB \ 1)$ 
    using icard-cartesian-product by blast
  also have  $\dots = \#1 * \#1$ 
    by (simp add: card-nAB' num-atoms-below-def)
  also have  $\dots = \#top$ 
    by (simp add: card-top)
  also have  $\dots = \text{icard } (AB \ \text{top})$ 
    by (simp add: card-nAB' num-atoms-below-def)
  finally have  $\text{icard } (\text{dom-cod } 'AB \ \text{top}) = \text{icard } (AB \ \text{top})$ 
    .
  thus  $\text{card } (\text{dom-cod } 'AB \ \text{top}) = \text{card } (AB \ \text{top})$ 
    using 1 by (smt (z3) finite-icard-card)
qed

subclass stone-relation-algebra-atomic-atomrect-atomsimple-finiteatoms
proof
  have  $\bigwedge a . \text{atom } a \wedge a \leq 1 \longrightarrow a * \text{top} * a \leq 1$ 
  proof
    fix a

```



```

assume 1: atom  $a \wedge a \leq 1$ 
show  $a * top * a \leq 1$ 
proof (rule ccontr)
  assume  $\neg a * top * a \leq 1$ 
  hence  $a * top * a \sqcap -1 \neq bot$ 
  by (simp add: pseudo-complement)
  from this obtain  $b$  where 2: atom  $b \wedge b \leq a * top * a \sqcap -1$ 
  using atomic by blast
  hence  $b * top \leq a * top$ 
  by (metis comp-associative dual-order.trans inf.boundedE mult-left-isotone
mult-right-isotone top.extremum)
  hence  $b * top \sqcap 1 \leq a * top \sqcap 1$ 
  using 1 comp-inf.comp-isotone by auto
  hence 3:  $b * top \sqcap 1 = a * top \sqcap 1$ 
  using 1 2 domain-atom by simp
  have  $top * b \leq top * a$ 
  using 2 by (metis comp-associative comp-inf.vector-top-closed
comp-inf-covector inf.boundedE mult-right-isotone vector-export-comp-unit
vector-top-closed)
  hence  $top * b \sqcap 1 \leq top * a \sqcap 1$ 
  using inf-mono by blast
  hence  $top * b \sqcap 1 = top * a \sqcap 1$ 
  using 1 2 codomain-atom by simp
  hence 4: dom-cod  $b = dom-cod\ a$ 
  using 3 by simp
  have  $b \in AB\ top \wedge a \in AB\ top$ 
  using 1 2 by simp
  hence  $b = a$ 
  using inj-onD dom-cod-inj-atoms 4 by smt
  thus False
  using 1 2 comp-inf.coreflexive-pseudo-complement le-bot by fastforce
qed
qed
thus  $\bigwedge a . atom\ a \longrightarrow a * top * a \leq a$ 
by (metis atom-rectangle-atom-one-rep)
qed

```

```

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

```

```

lemma atom-regular-rectangle:
  assumes atom  $a$ 
  shows  $--a = a * top * a$ 
proof (rule order.antisym)
  show  $--a \leq a * top * a$ 
  using asms atom-rectangle-regular ex231d pp-dist-comp by auto
  show  $a * top * a \leq --a$ 

```

```

proof (rule ccontr)
  assume  $\neg a * top * a \leq --a$ 
  hence  $a * top * a \sqcap -a \neq bot$ 
  by (simp add: pseudo-complement)
  from this obtain  $b$  where  $1: atom\ b \wedge b \leq a * top * a \sqcap -a$ 
  using atomic by blast
  hence  $2: b \neq a$ 
  using inf.absorb2 by fastforce
  have  $3: a \in AB\ (a * top * a) \wedge b \in AB\ (a * top * a)$ 
  using 1 assms ex231d by auto
  from atom-rectangle-card obtain  $c$  where  $AB\ (a * top * a) = \{c\}$ 
  using card-nAB' num-atoms-below-def assms icard-1-imp-singleton one-eSuc
by fastforce
  thus False
  using  $2\ 3$  by auto
qed
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 +
ra-card-notop

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

abbreviation r0000 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r0000 x y  $\equiv$  False
abbreviation r1000 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r1000 x y  $\equiv$   $\neg x \wedge \neg y$ 
abbreviation r0001 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r0001 x y  $\equiv$   $x \wedge y$ 
abbreviation r1001 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r1001 x y  $\equiv$   $x = y$ 
abbreviation r0110 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r0110 x y  $\equiv$   $x \neq y$ 
abbreviation r1111 :: bool  $\Rightarrow$  bool  $\Rightarrow$  bool where r1111 x y  $\equiv$  True

lemma r-all-different:
  r0000  $\neq$  r1000 r0000  $\neq$  r0001 r0000  $\neq$  r1001 r0000  $\neq$  r0110
r0000  $\neq$  r1111
  r1000  $\neq$  r0000 r1000  $\neq$  r0001 r1000  $\neq$  r1001 r1000  $\neq$  r0110
r1000  $\neq$  r1111
  r0001  $\neq$  r0000 r0001  $\neq$  r1000 r0001  $\neq$  r1001 r0001  $\neq$  r0110
r0001  $\neq$  r1111
  r1001  $\neq$  r0000 r1001  $\neq$  r1000 r1001  $\neq$  r0001 r1001  $\neq$  r0110
r1001  $\neq$  r1111
  r0110  $\neq$  r0000 r0110  $\neq$  r1000 r0110  $\neq$  r0001 r0110  $\neq$  r1001
r0110  $\neq$  r1111
  r1111  $\neq$  r0000 r1111  $\neq$  r1000 r1111  $\neq$  r0001 r1111  $\neq$  r1001 r1111  $\neq$  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.a2*
definition *conv-finite-4* :: *Enum.finite-4* \Rightarrow *Enum.finite-4* **where** *conv-finite-4* *x*
= *x*
definition *times-finite-4* :: *Enum.finite-4* \Rightarrow *Enum.finite-4* \Rightarrow *Enum.finite-4*
where *times-finite-4* *x y* = (case (*x,y*) of (*finite-4.a1,-*) \Rightarrow *finite-4.a1* |
(*-,finite-4.a1*) \Rightarrow *finite-4.a1* | (*finite-4.a2,y*) \Rightarrow *y* | (*x,finite-4.a2*) \Rightarrow *x* | - \Rightarrow
finite-4.a4)
definition *cardinality-finite-4* :: *Enum.finite-4* \Rightarrow *enat* **where** *cardinality-finite-4*
x = (case *x* of *finite-4.a1* \Rightarrow 0 | *finite-4.a4* \Rightarrow 2 | - \Rightarrow 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*
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** *zero-one-enat-neq(2)* **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* \Rightarrow *ra1* **is** *id* **by** *simp*

lift-definition *uminus-ra1* :: *ra1* \Rightarrow *ra1* **is** $\lambda r x y . \neg r x y$ **by** *auto*

lift-definition *sup-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *ra1* **is** $\lambda q r x y . q x y \vee r x y$ **by** *auto*

lift-definition *inf-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *ra1* **is** $\lambda q r x y . q x y \wedge r x y$ **by** *auto*

lift-definition *times-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *ra1* **is** $\lambda q r x y . \exists z . q x z \wedge r z y$ **by**
fastforce

lift-definition *minus-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *ra1* **is** $\lambda q r x y . q x y \wedge \neg r x y$ **by**
auto

lift-definition *less-eq-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *bool* **is** $\lambda q r . \forall x y . q x y \longrightarrow r x y .$

lift-definition *less-ra1* :: *ra1* \Rightarrow *ra1* \Rightarrow *bool* **is** $\lambda q r . (\forall x y . q x y \longrightarrow r x y) \wedge$
 $q \neq r .$

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

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** *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** *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 blast
subgoal apply transfer by simp
done

end

lemma four-cases:
  assumes  $P\ x1\ P\ x2\ P\ x3\ P\ x4$ 
  shows  $\forall y \in \{x \mid x \in \{x1, x2, x3, x4\}\} . P\ y$ 
  using assms by auto

lemma r-aux:
   $(\lambda x\ y. r1001\ x\ y \vee r0110\ x\ y) = r1111\ (\lambda x\ y. r1001\ x\ y \wedge r0110\ x\ y) = r0000$ 
   $(\lambda x\ y. r0110\ x\ y \vee r1001\ x\ y) = r1111\ (\lambda x\ y. r0110\ x\ y \wedge r1001\ x\ y) = r0000$ 
   $(\lambda x\ y. r1000\ x\ y \vee r0001\ x\ y) = r1001\ (\lambda x\ y. r1000\ x\ y \wedge r0001\ x\ y) = r0000$ 
   $(\lambda x\ y. r1000\ x\ y \vee r1001\ x\ y) = r1001\ (\lambda x\ y. r1000\ x\ y \wedge r1001\ x\ y) = r1000$ 
   $(\lambda x\ y. r0001\ x\ y \vee r1000\ x\ y) = r1001\ (\lambda x\ y. r0001\ x\ y \wedge r1000\ x\ y) = r0000$ 
   $(\lambda x\ y. r0001\ x\ y \vee r1001\ x\ y) = r1001\ (\lambda x\ y. r0001\ x\ y \wedge r1001\ x\ y) = r0001$ 
   $(\lambda x\ y. r1001\ x\ y \vee r1000\ x\ y) = r1001\ (\lambda x\ y. r1001\ x\ y \wedge r1000\ x\ y) = r1000$ 
   $(\lambda x\ y. r1001\ x\ y \vee r0001\ x\ y) = r1001\ (\lambda x\ y. r1001\ x\ y \wedge r0001\ x\ y) = r0001$ 
  by meson+

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

instance
  apply intro-classes
  subgoal 1 apply transfer by (metis zero-neq-numeral zero-one-enat-neq(1))
  subgoal 2 apply transfer by simp
  subgoal 3 apply transfer using r-aux r-all-different by auto
  subgoal 4 apply transfer using r-all-different zero-one-enat-neq(1) by auto
  subgoal 5 using 1 3 4 card-nAB nAB-univ-comp-meet 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 ra2 :: ra-card-atomic-finiteatoms
begin

lift-definition bot-ra2 :: ra2 is r0000 by simp
lift-definition one-ra2 :: ra2 is r1001 by simp
lift-definition top-ra2 :: ra2 is r1001 by simp
lift-definition conv-ra2 :: ra2  $\Rightarrow$  ra2 is id by simp
lift-definition uminus-ra2 :: ra2  $\Rightarrow$  ra2 is  $\lambda r\ x\ y. x = y \wedge \neg r\ x\ y$  by auto

```

lift-definition *sup-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *ra2* **is** $\lambda q r x y . q x y \vee r x y$ **by** *auto*
lift-definition *inf-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *ra2* **is** $\lambda q r x y . q x y \wedge r x y$ **by** *auto*
lift-definition *times-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *ra2* **is** $\lambda q r x y . \exists z . q x z \wedge r z y$ **by**
auto
lift-definition *minus-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *ra2* **is** $\lambda q r x y . q x y \wedge \neg r x y$ **by**
auto
lift-definition *less-eq-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *bool* **is** $\lambda q r . \forall x y . q x y \longrightarrow r x y .$
lift-definition *less-ra2* :: *ra2* \Rightarrow *ra2* \Rightarrow *bool* **is** $\lambda q r . (\forall x y . q x y \longrightarrow r x y) \wedge$
 $q \neq r .$
lift-definition *cardinality-ra2* :: *ra2* \Rightarrow *enat* **is** $\lambda q . \text{if } q = r0000 \text{ then } 0 \text{ else if } q$
 $= r1001 \text{ then } 2 \text{ else } 1 .$

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 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 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 simp
done

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*
nAB-univ-meet-comp)
 done

end

instantiation *prod* :: (*stone-relation-algebra, stone-relation-algebra*)
stone-relation-algebra

begin

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

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:  $\forall a::'a . \forall b::'b . atom (a,b) \longrightarrow (a = bot \wedge atom b) \vee (atom a \wedge b = bot)$ 
proof (intro allI, rule impI)
  fix a :: 'a and b :: 'b
  assume 2: atom (a,b)
  show (a = bot  $\wedge$  atom b)  $\vee$  (atom a  $\wedge$  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: atom c  $\wedge$  c  $\leq$  b
      using atomic by auto
      hence (bot,c)  $\leq$  (a,b)  $\wedge$  (bot,c)  $\neq$  bot
      by (simp add: less-eq-prod-def bot-prod.abs-eq)
      hence (bot,c) = (a,b)
      using 2 by auto
      thus ?thesis
      using 4 by auto
    qed
  next
    case False
    from this obtain c where 5: atom c  $\wedge$  c  $\leq$  a
    using atomic by auto
    hence (c,bot)  $\leq$  (a,b)  $\wedge$  (c,bot)  $\neq$  bot
    by (simp add: less-eq-prod-def bot-prod.abs-eq)
    hence (c,bot) = (a,b)
    using 2 by auto
    thus ?thesis
    using 5 by auto
  qed
qed
have 6:  $\{ (a,b) \mid a b . atom (a,b) \} \subseteq \{ (bot,b) \mid b::'b . atom b \} \cup \{ (a,bot) \mid a::'a . atom a \}$ 
proof
  fix x :: 'a  $\times$  'b
  assume x  $\in$   $\{ (a,b) \mid a b . atom (a,b) \}$ 
  from this obtain a b where 7: x = (a,b)  $\wedge$  atom (a,b)
  by auto
  hence (a = bot  $\wedge$  atom b)  $\vee$  (atom a  $\wedge$  b = bot)
  using 1 by simp
  thus x  $\in$   $\{ (bot,b) \mid b . atom b \} \cup \{ (a,bot) \mid a . atom a \}$ 
  using 7 by auto

```

```

qed
have finite { (bot,b) | b::'b . atom b } ∧ finite { (a,bot) | a::'a . atom a }
  by (simp add: finiteatoms)
hence 8: finite ( { (bot,b) | b::'b . atom b } ∪ { (a,bot) | a::'a . atom a } )
  by blast
have 9: finite { (a,b) | a b . atom (a::'a,b::'b) }
  by (rule rev-finite-subset, rule 8, rule 6)
have { (a,b) | a b . atom (a,b) } = { x :: 'a × 'b . atom x }
  by auto
thus finite { x :: 'a × 'b . 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 is 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-both-eq-0)
  subgoal apply transfer by (simp add: card-conv)
  subgoal apply transfer by (clarsimp, metis card-add
semiring-normalization-rules(20))
  subgoal apply transfer apply (clarsimp, rule iffI)
  subgoal by (metis add.commute add.right-neutral bot.extremum card-atom-iff
card-bot-iff dual-order.refl)
  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 $inf = (\sqcap)$ **and** $less-eq = (\leq)$ **and** $less = (<)$ **and** $sup = (\sqcup)$ **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** $inf = (\sqcap)$ **and** $less-eq = (\leq)$ **and** $less = (<)$ **and** $sup = (\sqcup)$ **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** $inf = (\sqcap)$ **and** $less-eq = (\leq)$ **and** $less = (<)$ **and** $sup = (\sqcup)$ **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 $\neg (\forall a::ra1 \times ra1 . atom\ a \longrightarrow a * top * a \leq a)$

proof -

let $?a = (1::ra1, bot::ra1)$

have $1: atom\ ?a$

proof

show $?a \neq bot$

by (*metis (full-types) bot-prod.transfer bot-ra1.rep-eq one-ra1.rep-eq*

prod.inject)

have $\bigwedge (a :: ra1) (b :: ra1) . (a,b) \leq ?a \implies (a,b) \neq bot \implies a = 1 \wedge b = bot$

proof -

fix $a\ b :: ra1$

assume $(a,b) \leq ?a$

hence $2: a \leq 1 \wedge b \leq bot$

by (*simp add: less-eq-prod-def*)

assume $(a,b) \neq bot$

hence $3: a \neq bot \wedge b = bot$

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 $a = 1 \wedge b = bot$

using $2\ 3$ **by** *blast*

qed

thus $\forall y . y \neq bot \wedge y \leq ?a \longrightarrow y = ?a$

by *clarsimp*

qed

```
have  $\neg ?a * top * ?a \leq ?a$ 
  apply (unfold top-prod-def times-prod-def less-eq-prod-def)
  apply transfer
  by auto
thus ?thesis
  using 1 by auto
qed

end
```

References

- [1] H. Furusawa and W. Guttman. Cardinality and representation of Stone relation algebras. *arXiv*, 2309.11676, 2023. <https://arxiv.org/abs/2309.11676>.
- [2] W. Guttman. Stone relation algebras. In P. Höfner, D. Pous, and G. Struth, editors, *Relational and Algebraic Methods in Computer Science*, volume 10226 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2017.
- [3] W. Guttman. Stone relation algebras. *Archive of Formal Proofs*, 2017.