

Birkhoff's Representation Theorem For Finite Distributive Lattices

Matthew Doty

February 6, 2026

Abstract

This theory proves a theorem of Birkhoff that asserts that any finite distributive lattice is isomorphic to the set of *down-sets* of that lattice's join-irreducible elements. The isomorphism preserves order, meets and joins as well as complementation in the case the lattice is a Boolean algebra. A consequence of this representation theorem is that every finite Boolean algebra is isomorphic to a powerset algebra.

Contents

1	Atoms, Join Primes and Join Irreducibles	2
2	Birkhoff's Representation Theorem For Finite Distributive Lattices	6
3	Finite Distributive Lattice Isomorphism	12
4	Cardinality	16

```

theory Birkhoff-Finite-Distributive-Lattices
  imports
    HOL-Library.Finite-Lattice
    HOL.Transcendental
begin

```

```

unbundle lattice-syntax

```

The proof of Birkhoff's representation theorem for finite distributive lattices [1] presented here follows Davey and Priestley [2].

1 Atoms, Join Primes and Join Irreducibles

Atomic elements are defined as follows.

```

definition (in bounded-lattice-bot) atomic :: 'a ⇒ bool where
  atomic x ≡ x ≠ ⊥ ∧ (∀ y. y ≤ x → y = ⊥ ∨ y = x)

```

Two related concepts are *join-prime* elements and *join-irreducible* elements.

```

definition (in bounded-lattice-bot) join-prime :: 'a ⇒ bool where
  join-prime x ≡ x ≠ ⊥ ∧ (∀ y z. x ≤ y ⊔ z → x ≤ y ∨ x ≤ z)

```

```

definition (in bounded-lattice-bot) join-irreducible :: 'a ⇒ bool where
  join-irreducible x ≡ x ≠ ⊥ ∧ (∀ y z. y < x → z < x → y ⊔ z < x)

```

```

lemma (in bounded-lattice-bot) join-irreducible-def':
  join-irreducible x = (x ≠ ⊥ ∧ (∀ y z. x = y ⊔ z → x = y ∨ x = z))
unfolding join-irreducible-def
by (metis
  nless-le
  sup.bounded-iff
  sup.cobounded1
  sup-ge2)

```

Every join-prime is also join-irreducible.

```

lemma (in bounded-lattice-bot) join-prime-implies-join-irreducible:
  assumes join-prime x
  shows join-irreducible x
  using assms
unfolding
  join-irreducible-def'
  join-prime-def
by (simp add: dual-order.eq-iff)

```

In the special case when the underlying lattice is distributive, the join-prime elements and join-irreducible elements coincide.

```

class bounded-distrib-lattice-bot = bounded-lattice-bot +
  assumes sup-inf-distrib1: x ⊔ (y ⊓ z) = (x ⊔ y) ⊓ (x ⊔ z)

```

```

begin

subclass distrib-lattice
  by (unfold-locales, metis (full-types) sup-inf-distrib1)

end

context complete-distrib-lattice
begin

subclass bounded-distrib-lattice-bot
  by (unfold-locales,
      metis (full-types)
      sup-inf-distrib1)

end

lemma (in bounded-distrib-lattice-bot) join-irreducible-is-join-prime:
  join-irreducible  $x$  = join-prime  $x$ 
proof
  assume join-prime  $x$ 
  thus join-irreducible  $x$ 
    by (simp add: join-prime-implies-join-irreducible)
next
  assume join-irreducible  $x$ 
  {
    fix  $y z$ 
    assume  $x \leq y \sqcup z$ 
    hence  $x = x \sqcap (y \sqcup z)$ 
      by (metis local.inf.orderE)
    hence  $x = (x \sqcap y) \sqcup (x \sqcap z)$ 
      using inf-sup-distrib1 by auto
    hence  $(x = x \sqcap y) \vee (x = x \sqcap z)$ 
      using ⟨join-irreducible  $x$ ⟩
      unfolding join-irreducible-def'
      by metis
    hence  $(x \leq y) \vee (x \leq z)$ 
      by (metis (full-types) local.inf.cobounded2)
  }
  thus join-prime  $x$ 
    by (metis
        ⟨join-irreducible  $x$ ⟩
        join-irreducible-def'
        join-prime-def)
qed

```

Every atomic element is join-irreducible.

```

lemma (in bounded-lattice-bot) atomic-implies-join-prime:
  assumes atomic  $x$ 

```

```

shows join-irreducible x
using assms
unfolding
  atomic-def
  join-irreducible-def'
by (metis (no-types, opaque-lifting)
      sup.cobounded2
      sup-bot.right-neutral)

```

In the case of Boolean algebras, atomic elements and join-prime elements are one-in-the-same.

lemma (in *boolean-algebra*) *join-prime-is-atomic*:
atomic x = join-prime x

```

proof
assume atomic x
{
  fix y z
  assume  $x \leq y \sqcup z$ 
  hence  $x = (x \sqcap y) \sqcup (x \sqcap z)$ 
  using inf.absorb1 inf-sup-distrib1 by fastforce
  moreover
  have  $x \leq y \vee (x \sqcap y) = \perp$ 
     $x \leq z \vee (x \sqcap z) = \perp$ 
  using  $\langle \text{atomic } x \rangle$  inf.cobounded1 inf.cobounded2
  unfolding atomic-def
  by fastforce+
  ultimately have  $x \leq y \vee x \leq z$ 
  using  $\langle \text{atomic } x \rangle$  atomic-def by auto
}
thus join-prime x
using  $\langle \text{atomic } x \rangle$  join-prime-def atomic-def
by auto
next
assume join-prime x
{
  fix y
  assume  $y \leq x \ y \neq x$ 
  hence  $x = x \sqcup y$ 
  using sup.orderE by blast
  also have  $\dots = (x \sqcup y) \sqcap (y \sqcup -y)$ 
  by simp
  finally have  $x = (x \sqcap -y) \sqcup y$ 
  by (simp add: sup-inf-distrib2)
  hence  $x \leq -y$ 
  using
     $\langle \text{join-prime } x \rangle$ 
     $\langle y \neq x \rangle$ 
     $\langle y \leq x \rangle$ 
    antisym-conv
}

```

```

      inf-le2
      sup-neg-inf
unfolding join-prime-def
by blast
hence  $y \leq y \sqcap -y$ 
by (metis
      ⟨ $x = x \sqcup y$ ⟩
      inf.orderE
      inf-compl-bot-right
      inf-sup-absorb
      order-refl
      sup commute)
hence  $y = \perp$ 
      using sup-absorb2 by fastforce
}
thus atomic  $x$ 
using ⟨join-prime  $x$ ⟩
unfolding
  atomic-def
  join-prime-def
by auto
qed

```

All atomic elements are disjoint.

lemma (in *bounded-lattice-bot*) *atomic-disjoint*:

```

assumes atomic  $\alpha$ 
and atomic  $\beta$ 
shows  $(\alpha = \beta) \longleftrightarrow (\alpha \sqcap \beta \neq \perp)$ 

```

proof

```

assume  $\alpha = \beta$ 
hence  $\alpha \sqcap \beta = \alpha$ 
by simp
thus  $\alpha \sqcap \beta \neq \perp$ 
using ⟨atomic  $\alpha$ ⟩
unfolding atomic-def
by auto

```

next

```

assume  $\alpha \sqcap \beta \neq \perp$ 
hence  $\beta \leq \alpha \wedge \alpha \leq \beta$ 
by (metis
      assms
      atomic-def
      inf-absorb2
      inf-le1
      inf-le2)

```

```

thus  $\alpha = \beta$  by auto

```

qed

definition (in *bounded-lattice-bot*) *atomic-elements* ($\langle \mathcal{A} \rangle$) **where**

$\mathcal{A} \equiv \{a \text{ . atomic } a\}$

definition (in *bounded-lattice-bot*) *join-irreducible-elements* ($\langle \mathcal{J} \rangle$) **where**
 $\mathcal{J} \equiv \{a \text{ . join-irreducible } a\}$

2 Birkhoff's Representation Theorem For Finite Distributive Lattices

Birkhoff's representation theorem for finite distributive lattices follows from the fact that every non- \perp element can be represented by the join-irreducible elements beneath it.

In this section we merely demonstrate the representation aspect of Birkhoff's theorem. In §3 we show this representation is a lattice homomorphism.

The first step to representing elements is to show that there *exist* join-irreducible elements beneath them. This is done by showing if there is no join-irreducible element, we can make a descending chain with more elements than the finite Boolean algebra under consideration.

fun (in *order*) *descending-chain-list* :: 'a list \Rightarrow bool **where**
descending-chain-list [] = True
| *descending-chain-list* [x] = True
| *descending-chain-list* (x # x' # xs)
= (x < x' \wedge *descending-chain-list* (x' # xs))

lemma (in *order*) *descending-chain-list-tail*:
assumes *descending-chain-list* (s # S)
shows *descending-chain-list* S
using *assms*
by (*induct* S, *auto*)

lemma (in *order*) *descending-chain-list-drop-penultimate*:
assumes *descending-chain-list* (s # s' # S)
shows *descending-chain-list* (s # S)
using *assms*
by (*induct* S, *simp*, *auto*)

lemma (in *order*) *descending-chain-list-less-than-others*:
assumes *descending-chain-list* (s # S)
shows $\forall s' \in \text{set } S. s < s'$
using *assms*
by (*induct* S,
auto,
simp add: descending-chain-list-drop-penultimate)

lemma (in *order*) *descending-chain-list-distinct*:
assumes *descending-chain-list* S

shows *distinct S*
using *assms*
by (*induct S*,
simp,
meson
descending-chain-list-less-than-others
descending-chain-list-tail
distinct.simps(2)
less-irrefl)

lemma (*in finite-distrib-lattice*) *join-irreducible-lower-bound-exists*:

assumes $\neg (x \leq y)$
shows $\exists z \in \mathcal{J}. z \leq x \wedge \neg (z \leq y)$
proof (*rule ccontr*)
assume $\star: \neg (\exists z \in \mathcal{J}. z \leq x \wedge \neg (z \leq y))$
{
 fix $z :: 'a$
 assume
 $z \leq x$
 $\neg (z \leq y)$
 with \star **obtain** $p\ q$ **where**
 $p < z$
 $q < z$
 $p \sqcup q = z$
 by (*metis (full-types)*
 bot-least
 dual-order.not-eq-order-implies-strict
 join-irreducible-def'
 join-irreducible-elements-def
 sup-ge1
 sup-ge2
 mem-Collect-eq)
 hence $\neg (p \leq y) \vee \neg (q \leq y)$
 by (*metis (full-types) $\neg z \leq y$ sup-least*)
 hence $\exists p < z. \neg (p \leq y)$
 by (*metis <math>p < z</math> <math>q < z</math>*)
}
note *fresh = this*
{
 fix $n :: nat$
 have $\exists S. \text{descending-chain-list } S$
 $\wedge \text{length } S = n$
 $\wedge (\forall s \in \text{set } S. s \leq x \wedge \neg (s \leq y))$
 proof (*induct n*)
 case 0
 then show ?*case* **by** *simp*
 next
 case (*Suc n*)
 then show ?*case* **proof** (*cases n = 0*)

```

case True
hence descending-chain-list [x]
       $\wedge$  length [x] = Suc n
       $\wedge$  ( $\forall s \in \text{set } [x]. s \leq x \wedge \neg (s \leq y)$ )
by (metis
      Suc
      assms
      length-0-conv
      length-Suc-conv
      descending-chain-list.simps(2)
      le-less set-ConsD)
then show ?thesis
by blast
next
case False
from this obtain s S where
      descending-chain-list (s # S)
      length (s # S) = n
       $\forall s \in \text{set } (s \# S). s \leq x \wedge \neg (s \leq y)$ 
using
      Suc.hyps
      length-0-conv
      descending-chain-list.elims(2)
by metis
note A = this
hence  $s \leq x \wedge \neg (s \leq y)$  by auto
obtain s' :: 'a where
       $s' < s$ 
       $\neg (s' \leq y)$ 
using
      fresh [OF ‹s ≤ x› ‹¬ (s ≤ y)›]
by auto
note B = this
let ?S' = s' # s # S
from A and B have
      descending-chain-list ?S'
      length ?S' = Suc n
       $\forall s \in \text{set } ?S'. s \leq x \wedge \neg (s \leq y)$ 
by auto
then show ?thesis by blast
qed
qed
}
from this obtain S :: 'a list where
      descending-chain-list S
       $\text{length } S = 1 + (\text{card } (\text{UNIV}::'a \text{ set}))$ 
by auto
hence  $\text{card } (\text{set } S) = 1 + (\text{card } (\text{UNIV}::'a \text{ set}))$ 
using descending-chain-list-distinct

```

```

      distinct-card
    by fastforce
  hence  $\neg \text{card } (\text{set } S) \leq \text{card } (\text{UNIV}::'a \text{ set})$ 
    by presburger
  thus False
    using card-mono finite-UNIV by blast
qed

```

definition (in *bounded-lattice-bot*)
join-irreducibles-embedding :: $'a \Rightarrow 'a \text{ set } (\langle \{ - \} \rangle [50])$ **where**
 $\{ x \} \equiv \{ a \in \mathcal{J}. a \leq x \}$

We can now show every element is exactly the suprema of the join-irreducible elements beneath them in any distributive lattice.

theorem (in *finite-distrib-lattice*) *sup-join-prime-embedding-ident*:

$$x = \bigsqcup \{ x \}$$

proof –

```

  have  $\forall a \in \{ x \}. a \leq x$ 
    by (metis (no-types, lifting)
        join-irreducibles-embedding-def
        mem-Collect-eq)

```

```

  hence  $\bigsqcup \{ x \} \leq x$ 
    by (simp add: Sup-least)

```

moreover

```

{
  fix  $y :: 'a$ 
  assume  $\bigsqcup \{ x \} \leq y$ 
  have  $x \leq y$ 
  proof (rule ccontr)
    assume  $\neg x \leq y$ 
    from this obtain  $a$  where
       $a \in \mathcal{J}$ 
       $a \leq x$ 
       $\neg a \leq y$ 
    using join-irreducible-lower-bound-exists [OF  $\langle \neg x \leq y \rangle$ ]
    by metis
  hence  $a \in \{ x \}$ 
    by (metis (no-types, lifting)
        join-irreducibles-embedding-def
        mem-Collect-eq)
  hence  $a \leq y$ 
    using  $\langle \bigsqcup \{ x \} \leq y \rangle$ 
      Sup-upper
      order.trans
    by blast
  thus False
    by (metis (full-types)  $\langle \neg a \leq y \rangle$ )
}

```

ultimately show *?thesis*
 using *antisym-conv* by *blast*
 qed

Just as $x = \bigsqcup \{ x \}$, the reverse is also true; $\lambda x. \{ x \}$ and $\lambda S. \bigsqcup S$ are inverses where $S \in \mathcal{OJ}$, the set of downsets in $Pow \mathcal{J}$.

definition (in *bounded-lattice-bot*) *down-irreducibles* ($\langle \mathcal{OJ} \rangle$) **where**
 $\mathcal{OJ} \equiv \{ S \in Pow \mathcal{J} . (\exists x . S = \{ x \}) \}$

lemma (in *finite-distrib-lattice*) *join-irreducible-embedding-sup-ident*:

assumes $S \in \mathcal{OJ}$

shows $S = \{ \bigsqcup S \}$

proof –

obtain x **where**

$S = \{ x \}$

using

$\langle S \in \mathcal{OJ} \rangle$

unfolding

down-irreducibles-def

by *auto*

with $\langle S \in \mathcal{OJ} \rangle$ **have** $\forall s \in S. s \in \mathcal{J} \wedge s \leq \bigsqcup S$

unfolding

down-irreducibles-def

Pow-def

using *Sup-upper*

by *fastforce*

hence $S \subseteq \{ \bigsqcup S \}$

unfolding *join-irreducibles-embedding-def*

by *blast*

moreover

{

fix y

assume

$y \in \mathcal{J}$

$y \leq \bigsqcup S$

have *finite S* **by** *auto*

from $\langle \text{finite } S \rangle$ **and** $\langle y \leq \bigsqcup S \rangle$ **have** $\exists s \in S. y \leq s$

proof (*induct S rule: finite-induct*)

case *empty*

hence $y \leq \perp$

by (*metis Sup-empty*)

then show *?case*

using

$\langle y \in \mathcal{J} \rangle$

unfolding

join-irreducible-elements-def

join-irreducible-def

by (*metis (mono-tags, lifting)*)

le-bot

```

      mem-Collect-eq)
next
  case (insert s S)
  hence  $y \leq s \vee y \leq \sqcup S$ 
  using
     $\langle y \in \mathcal{J} \rangle$ 
  unfolding
    join-irreducible-elements-def
    join-irreducible-is-join-prime
    join-prime-def
  by auto
  then show ?case
  by (metis (full-types)
      insert.hyps(3)
      insertCI)
qed
hence  $y \leq x$ 
  by (metis (no-types, lifting)
       $\langle S = \{ x \} \rangle$ 
      join-irreducibles-embedding-def
      order-trans
      mem-Collect-eq)
hence  $y \in S$ 
  by (metis (no-types, lifting)
       $\langle S = \{ x \} \rangle$ 
       $\langle y \in \mathcal{J} \rangle$ 
      join-irreducibles-embedding-def
      mem-Collect-eq)
}
hence  $\{ \sqcup S \} \subseteq S$ 
  unfolding
    join-irreducibles-embedding-def
  by blast
ultimately show ?thesis by auto
qed

```

Given that $\lambda x. \{ x \}$ has a left and right inverse, we can show it is a *bijection*.

The bijection below is recognizable as a form of *Birkhoff's Representation Theorem* for finite distributive lattices.

theorem (in *finite-distrib-lattice*) *birkhoffs-theorem*:

bij-betw $(\lambda x. \{ x \}) \text{ UNIV } \mathcal{O}\mathcal{J}$

unfolding *bij-betw-def*

proof

```

{
  fix x y
  assume  $\{ x \} = \{ y \}$ 
  hence  $\sqcup \{ x \} = \sqcup \{ y \}$ 

```

```

    by simp
  hence  $x = y$ 
    using sup-join-prime-embedding-ident
    by auto
}
thus inj ( $\lambda x. \{ x \}$ )
  unfolding inj-def
  by auto
next
show range ( $\lambda x. \{ x \}$ ) =  $\mathcal{O}\mathcal{J}$ 
  unfolding
    down-irreducibles-def
    join-irreducibles-embedding-def
  by auto
qed

```

3 Finite Distributive Lattice Isomorphism

The form of Birkhoff's theorem presented in §2 simply gave a bijection between a finite distributive lattice and the downsets of its join-irreducible elements. This relationship can be extended to a full-blown *lattice homomorphism*. In particular we have the following properties:

- \perp and \top are preserved; specifically $\{ \perp \} = \{ \}$ and $\{ \top \} = \mathcal{J}$.
- Order is preserved: $x \leq y = (\{ x \} \subseteq \{ y \})$.
- $\lambda x . \{ x \}$ is a lower complete semi-lattice homomorphism, mapping $\{ \sqcup X \} = (\cup x \in X . \{ x \})$.
- In addition to preserving arbitrary joins, $\lambda x . \{ x \}$ is a lattice homomorphism, since it also preserves finitary meets with $\{ x \sqcap y \} = \{ x \} \cap \{ y \}$. Arbitrary meets are also preserved, but relative to a top element \mathcal{J} , or in other words $\{ \prod X \} = \mathcal{J} \cap (\cap x \in X . \{ x \})$.
- In the case of a Boolean algebra, complementation corresponds to relative set complementation via $\{ - x \} = \mathcal{J} - \{ x \}$.

lemma (in *finite-distrib-lattice*) *join-irreducibles-bot*:

```

{  $\perp$  } = { }
unfolding
  join-irreducibles-embedding-def
  join-irreducible-elements-def
  join-irreducible-is-join-prime
  join-prime-def
by (simp add: bot-unique)

```

lemma (in *finite-distrib-lattice*) *join-irreducibles-top*:

$$\{\top\} = \mathcal{J}$$

unfolding

join-irreducibles-embedding-def

join-irreducible-elements-def

join-irreducible-is-join-prime

join-prime-def

by *auto*

lemma (in *finite-distrib-lattice*) *join-irreducibles-order-isomorphism*:

$$x \leq y = (\{x\} \subseteq \{y\})$$

by (*rule iffI*,

metis (*mono-tags*, *lifting*)

join-irreducibles-embedding-def

order-trans

mem-Collect-eq

subsetI,

metis (*full-types*)

Sup-subset-mono

sup-join-prime-embedding-ident)

lemma (in *finite-distrib-lattice*) *join-irreducibles-join-homomorphism*:

$$\{x \sqcup y\} = \{x\} \cup \{y\}$$

proof

show $\{x \sqcup y\} \subseteq \{x\} \cup \{y\}$

unfolding

join-irreducibles-embedding-def

join-irreducible-elements-def

join-irreducible-is-join-prime

join-prime-def

by *blast*

next

show $\{x\} \cup \{y\} \subseteq \{x \sqcup y\}$

unfolding

join-irreducibles-embedding-def

join-irreducible-elements-def

join-irreducible-is-join-prime

join-prime-def

using

le-supI1

sup.absorb-iff1

sup.assoc

by *force*

qed

lemma (in *finite-distrib-lattice*) *join-irreducibles-sup-homomorphism*:

$$\{\bigsqcup X\} = (\bigcup x \in X . \{x\})$$

proof –

have *finite X*

```

    by simp
  thus ?thesis
proof (induct X rule: finite-induct)
  case empty
  then show ?case by (simp add: join-irreducibles-bot)
next
  case (insert x X)
  then show ?case by (simp add: join-irreducibles-join-homomorphism)
qed
qed

```

```

lemma (in finite-distrib-lattice) join-irreducibles-meet-homomorphism:
  { x  $\sqcap$  y } = { x }  $\cap$  { y }
  unfolding
    join-irreducibles-embedding-def
  by auto

```

Arbitrary meets are also preserved, but relative to a top element \mathcal{J} .

```

lemma (in finite-distrib-lattice) join-irreducibles-inf-homomorphism:
  {  $\sqcap$  X } =  $\mathcal{J} \cap (\bigcap x \in X. \{ x \})$ 
proof -
  have finite X
  by simp
  thus ?thesis
proof (induct X rule: finite-induct)
  case empty
  then show ?case by (simp add: join-irreducibles-top)
next
  case (insert x X)
  then show ?case by (simp add: join-irreducibles-meet-homomorphism, blast)
qed
qed

```

Finally, we show that complementation is preserved.

To begin, we define the class of finite Boolean algebras. This class is simply an extension of *boolean-algebra*, extended with *finite UNIV* as per the axiom class *finite*. We also extend the language of the class with *infima* and *suprema* (i.e. $\sqcap A$ and $\sqcup A$ respectively).

```

class finite-boolean-algebra = boolean-algebra + finite + Inf + Sup +
  assumes Inf-def:  $\sqcap A = \text{Finite-Set.fold } (\sqcap) \top A$ 
  assumes Sup-def:  $\sqcup A = \text{Finite-Set.fold } (\sqcup) \perp A$ 
begin

```

Finite Boolean algebras are trivially a subclass of finite distributive lattices, which are necessarily *complete*.

```

subclass finite-distrib-lattice-complete

```

```

using
  Inf-fin.coboundedI
  Sup-fin.coboundedI
  finite-UNIV
  le-bot
  top-unique
  Inf-def
  Sup-def
by (unfold-locales, blast, fastforce+)

subclass bounded-distrib-lattice-bot
by (unfold-locales, metis sup-inf-distrib1)
end

lemma (in finite-boolean-algebra) join-irreducibles-complement-homomorphism:
   $\{ - x \} = \mathcal{J} - \{ x \}$ 
proof
show  $\{ - x \} \subseteq \mathcal{J} - \{ x \}$ 
proof
  fix j
  assume  $j \in \{ - x \}$ 
  hence  $j \notin \{ x \}$ 
  unfolding
    join-irreducibles-embedding-def
    join-irreducible-elements-def
    join-irreducible-is-join-prime
    join-prime-def
  by (metis
    (mono-tags, lifting)
    CollectD
    bot-unique
    inf.boundedI
    inf-compl-bot)
  thus  $j \in \mathcal{J} - \{ x \}$ 
  using  $\langle j \in \{ - x \} \rangle$ 
  unfolding
    join-irreducibles-embedding-def
  by blast
qed
next
show  $\mathcal{J} - \{ x \} \subseteq \{ - x \}$ 
proof
  fix j
  assume  $j \in \mathcal{J} - \{ x \}$ 
  hence  $j \in \mathcal{J}$  and  $\neg j \leq x$ 
  unfolding join-irreducibles-embedding-def
  by blast+
  moreover have  $j \leq x \sqcup -x$ 
  by auto

```

ultimately have $j \leq -x$
unfolding *join-irreducible-elements-def*
join-irreducible-is-join-prime
join-prime-def
by *blast*
thus $j \in \{ -x \}$
unfolding *join-irreducibles-embedding-def*
using $\langle j \in \mathcal{J} \rangle$
by *auto*
qed
qed

4 Cardinality

Another consequence of Birkhoff's theorem from §2 is that every finite Boolean algebra has a cardinality which is a power of two. This gives a bound on the number of atoms/join-prime/irreducible elements, which must be logarithmic in the size of the finite Boolean algebra they belong to.

We first show that $\mathcal{O}\mathcal{J}$, the downsets of the join-irreducible elements \mathcal{J} , are the same as the powerset of \mathcal{J} in any finite Boolean algebra.

lemma (in *finite-boolean-algebra*) $\mathcal{O}\mathcal{J}$ -is-Pow- \mathcal{J} :

$$\mathcal{O}\mathcal{J} = \text{Pow } \mathcal{J}$$

proof

show $\mathcal{O}\mathcal{J} \subseteq \text{Pow } \mathcal{J}$

unfolding *down-irreducibles-def*

by *auto*

next

show $\text{Pow } \mathcal{J} \subseteq \mathcal{O}\mathcal{J}$

proof (*rule ccontr*)

assume $\neg \text{Pow } \mathcal{J} \subseteq \mathcal{O}\mathcal{J}$

from this obtain S **where**

$$S \subseteq \mathcal{J}$$

$$\forall x. S \neq \{a \in \mathcal{J}. a \leq x\}$$

unfolding

down-irreducibles-def

join-irreducibles-embedding-def

by *auto*

hence $S \neq \{a \in \mathcal{J}. a \leq \bigsqcup S\}$

by *auto*

moreover

have $\forall s \in S. s \in \mathcal{J} \wedge s \leq \bigsqcup S$

by (*metis (no-types, lifting)*)

$$\langle S \subseteq \mathcal{J} \rangle$$

Sup-upper subsetD)

hence $S \subseteq \{a \in \mathcal{J}. a \leq \bigsqcup S\}$

by (*metis (mono-tags, lifting) Ball-Collect*)

ultimately have $\exists y \in \mathcal{J} . y \leq \bigsqcup S \wedge y \notin S$
by (*metis* (*mono-tags*, *lifting*)
mem-Collect-eq
subsetI
subset-antisym)

moreover
 {
fix y
assume
 $y \in \mathcal{J}$
 $y \leq \bigsqcup S$
from
finite [*of S*]
 $\langle y \leq \bigsqcup S \rangle$
 $\langle S \subseteq \mathcal{J} \rangle$
have $y \in S$
proof (*induct S rule: finite-induct*)
case *empty*
hence $y \leq \perp$
by (*metis* (*full-types*) *local.Sup-empty*)
then show *?case*
using $\langle y \in \mathcal{J} \rangle$
unfolding
join-irreducible-elements-def
join-irreducible-def
by (*metis* (*mono-tags*, *lifting*)
le-bot
mem-Collect-eq)

next
case (*insert s S*)
hence $y \leq s \vee y \leq \bigsqcup S$
using $\langle y \in \mathcal{J} \rangle$
unfolding
join-irreducible-elements-def
join-irreducible-is-join-prime
join-prime-def
by *simp*

moreover
 {
assume $y \leq s$
have *atomic s*
by (*metis in-mono*
insert.premis(2)
insertCI
join-irreducible-elements-def
join-irreducible-is-join-prime
join-prime-is-atomic
mem-Collect-eq)
hence $y = s$

```

    by (metis (no-types, lifting)
        ⟨y ∈ J⟩
        ⟨y ≤ s⟩
        atomic-def
        join-irreducible-def
        join-irreducible-elements-def
        mem-Collect-eq)
  }
  ultimately show ?case
  by (metis
      insert.prem2
      insert-iff
      insert-subset
      insert3)
qed
}
ultimately show False by auto
qed
qed

```

lemma (in *finite-boolean-algebra*) *UNIV-card*:
 $\text{card } (\text{UNIV}::'a \text{ set}) = \text{card } (\text{Pow } \mathcal{J})$
using *bij-betw-same-card* [where $f = \lambda x. \{ \{ x \} \}$]
birkhoffs-theorem
unfolding *OT-is-Pow-J*
by *blast*

lemma *finite-Pow-card*:
assumes *finite X*
shows $\text{card } (\text{Pow } X) = 2^{\text{powr } (\text{card } X)}$
using *assms*
proof (*induct X rule: finite-induct*)
case *empty*
then show ?case **by** *fastforce*
next
case (*insert x X*)
have $0 \leq (2 :: \text{real})$ **by** *auto*
hence *two-powr-one*: $(2 :: \text{real}) = 2^{\text{powr } 1}$ **by** *fastforce*
have *bij-betw* $(\lambda x. \text{fst } x \cup \text{snd } x) (\{\{\}, \{x\}\} \times \text{Pow } X) (\text{Pow } (\text{insert } x X))$
unfolding *bij-betw-def*
proof
 {
fix $y z$
assume
 $y \in \{\{\}, \{x\}\} \times \text{Pow } X$
 $z \in \{\{\}, \{x\}\} \times \text{Pow } X$

```

    fst y ∪ snd y = fst z ∪ snd z
    (is ?Uy = ?Uz)
  hence
    x ∉ snd y
    x ∉ snd z
    fst y = {x} ∨ fst y = {}
    fst z = {x} ∨ fst z = {}
  using insert.hyps(2) by auto
  hence
    x ∈ ?Uy ⟷ fst y = {x}
    x ∈ ?Uz ⟷ fst z = {x}
    x ∉ ?Uy ⟷ fst y = {}
    x ∉ ?Uz ⟷ fst z = {}
    snd y = ?Uy - {x}
    snd z = ?Uz - {x}
  by auto
  hence
    x ∈ ?Uy ⟷ y = ({x}, ?Uy - {x})
    x ∈ ?Uz ⟷ z = ({x}, ?Uz - {x})
    x ∉ ?Uy ⟷ y = ({} , ?Uy - {x})
    x ∉ ?Uz ⟷ z = ({} , ?Uz - {x})
  by (metis fst-conv prod.collapse)+
  hence y = z
  using ⟨?Uy = ?Uz⟩
  by metis
}
thus inj-on (λx. fst x ∪ snd x) ({}, {x}) × Pow X
  unfolding inj-on-def
  by auto
next
show (λx. fst x ∪ snd x) ‘ ({} , {x}) × Pow X = Pow (insert x X)
proof (intro equalityI subsetI)
  fix y
  assume y ∈ (λx. fst x ∪ snd x) ‘ ({} , {x}) × Pow X
  from this obtain z where
    z ∈ ({} , {x}) × Pow X
    y = fst z ∪ snd z
  by auto
  hence
    snd z ⊆ X
    fst z ⊆ insert x X
  using SigmaE by auto
  thus y ∈ Pow (insert x X)
  using ⟨y = fst z ∪ snd z⟩ by blast
next
  fix y
  assume y ∈ Pow (insert x X)
  let ?z = (if x ∈ y then {x} else {}, y - {x})
  have ?z ∈ ({} , {x}) × Pow X

```

```

    using ⟨y ∈ Pow (insert x X)⟩ by auto
  moreover have (λx. fst x ∪ snd x) ?z = y
    by auto
  ultimately show y ∈ (λx. fst x ∪ snd x) ‘ ({{}}, {x}) × Pow X
    by blast
qed
qed
hence card (Pow (insert x X)) = card ({{}}, {x}) × Pow X
  using bij-betw-same-card by fastforce
also have ... = 2 * card (Pow X)
  by (simp add: insert.hyps(1))
also have ... = 2 * (2 powr (card X))
  by (simp add: insert.hyps(3))
also have ... = (2 powr 1) * 2 powr (card X)
  using two-powr-one
  by fastforce
also have ... = 2 powr (1 + card X)
  by (simp add: powr-add)
also have ... = 2 powr (card (insert x X))
  by (simp add: insert.hyps(1) insert.hyps(2))
finally show ?case .
qed

```

```

lemma (in finite-boolean-algebra) UNIV-card-powr-2:
  card (UNIV :: 'a set) = 2 powr (card  $\mathcal{J}$ )
  using
    finite [of  $\mathcal{J}$ ]
    finite-Pow-card [of  $\mathcal{J}$ ]
    UNIV-card
  by linarith

```

```

lemma (in finite-boolean-algebra) join-irreducibles-card-log-2:
  card  $\mathcal{J}$  = log 2 (card (UNIV :: 'a set))
proof (cases card (UNIV :: 'a set) = 1)
  case True
  hence ∃ x :: 'a. UNIV = {x}
    using card-1-singletonE by blast
  hence ∀ x y :: 'a. x ∈ UNIV ⟶ y ∈ UNIV ⟶ x = y
    by (metis (mono-tags) singletonD)
  hence ∀ x y :: 'a. x = y
    by blast
  hence ∀ x. x = ⊥
    by blast
  hence  $\mathcal{J}$  = {}
  unfolding
    join-irreducible-elements-def
    join-irreducible-is-join-prime
    join-prime-def
  by blast

```

```

hence  $\text{card } \mathcal{J} = (0 :: \text{real})$ 
  by simp
moreover
have  $\log 2 (\text{card } (\text{UNIV} :: 'a \text{ set})) = 0$ 
  by (simp add: True)
ultimately show ?thesis by auto
next
case False
hence  $0 < 2^{\text{powr } (\text{card } \mathcal{J})} - 2^{\text{powr } (\text{card } \mathcal{J})} \neq 1$ 
  using finite-UNIV-card-ge-0 finite UNIV-card-powr-2
  by (simp, linarith)
hence  $\log 2 (2^{\text{powr } (\text{card } \mathcal{J})}) = \text{card } \mathcal{J}$ 
  by simp
then show ?thesis
  using UNIV-card-powr-2
  by simp
qed
end

```

References

- [1] G. Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, Sept. 1937.
- [2] B. A. Davey and H. A. Priestley. Chapter 5. Representation: The finite case. In *Introduction to Lattices and Order*, pages 112–124. Cambridge University Press, Cambridge, UK ; New York, NY, 2nd ed edition, 2002.