Computing and formalizing residuated lattices and relation algebras

Peter Jipsen

Chapman University

Logic, Algebra and Truth Degrees

LATD 2025, July 21 – 25

University of Siena, Italy

Some historical background on residuated lattices

Ward and Dilworth defined **residuated lattices** (RLs) in 1939

abstracting from a lattice of ideals of a ring with ideal multiplication.

Gentzen (1935) developed **proof theory** of propositional logics without assuming all **structural rules**.

Proof theory of many logics was developed as sequent calculi

Intuitionistic, relevance and linear logic fit well into this framework

Dŏsen 1990 proposed using the term **substructural logic**

Blount 1999, Tsinakis 2003: On the structure of residuated lattices

Residuated lattices and relation algebras in proof assistants

Rocq: Damien Pous, **Relation Algebra and KAT in Coq**, 2012,
`https://perso.ens-lyon.fr/damien.pous/ra/`

Isabelle: A Armstrong, S Foster, G Struth, T Weber, 2014,
Archive of Formal Proofs, **Relation Algebra**
`https://www.isa-afp.org/entries/Relation_Algebra.html`

Isabelle: Victor B. F. Gomes, Georg Struth, 2015,
Archive of Formal Proofs, **Residuated lattices**
`https://www.isa-afp.org/entries/Residuated_Lattices.html`

Brief background on proof assistants

Automated theorem provers have been developed since the 1960s, see McCune and Wos [1997] for a brief history.

Mostly restricted to first-order logic: **Otter, Prover9/Mace4, SPASS, E-prover, Vampire,** ...

Satisfiability Modulo Theories (SMT) solvers: **Z3, CVC5,** ...

Interactive theorem provers: **Mizar, PVS, HOL, HOL-light, Isabelle, Rocq, Agda, Lean,** ...

Based on higher-order logics, (dependent) type theories, large libraries of formal proofs

Lean classes for residuated lattices

```
import Mathlib.Order.Lattice
import Mathlib.Algebra.Group.Defs

class ResiduatedPoSemigroup (A : Type*) extends
    PartialOrder A, Semigroup A, Div A, SDiff A  where

  lres : ∀ x y z : A, y ≤ x \ z ↔ x * y ≤ z

  rres : ∀ x y z : A, x ≤ z / y ↔ x * y ≤ z

class ResiduatedPoMonoid (A : Type*) extends
    ResiduatedPoSemigroup A, Monoid A

class ResiduatedLattice (A : Type*) extends
    ResiduatedPoMonoid A, Lattice A
```

Proving some lemmas about residuated lattices

```
@[simp]
lemma le_rres_iff : x ≤ z / y ↔ x * y ≤ z :=
  ResiduatedPoSemigroup.rres _ _ _

lemma le_mul_rres : x ≤ (x * y) / y := by rw [le_rres_iff]

lemma le_rres_lres1 : x ≤ (y / x) \ y := by
  rw [le_lres_iff]
  exact rres_mul_le

lemma rres_le_rres_left (x y z : A) :
      x ≤ y → z / y ≤ z / x := by
  intro h
  rw [le_rres_iff]
  rw [← le_lres_iff]
  exact le_trans h le_rres_lres1
```

Another lemma about residuated lattices

```
lemma mul_join : x * (y ⊔ z) = x * y ⊔ x * z := by
  apply le_antisymm
  . have : y ≤ y ⊔ z := by exact le_sup_left
    rw [← le_lres_iff]
    rw [sup_le_iff]
    apply And.intro
    . rw [le_lres_iff]
      exact le_sup_left
    . rw [le_lres_iff]
      exact le_sup_right
  . rw [sup_le_iff]
    apply And.intro
    . exact mul_le_mul_left _ _ _ le_sup_left
    . exact mul_le_mul_left _ _ _ le_sup_right
```

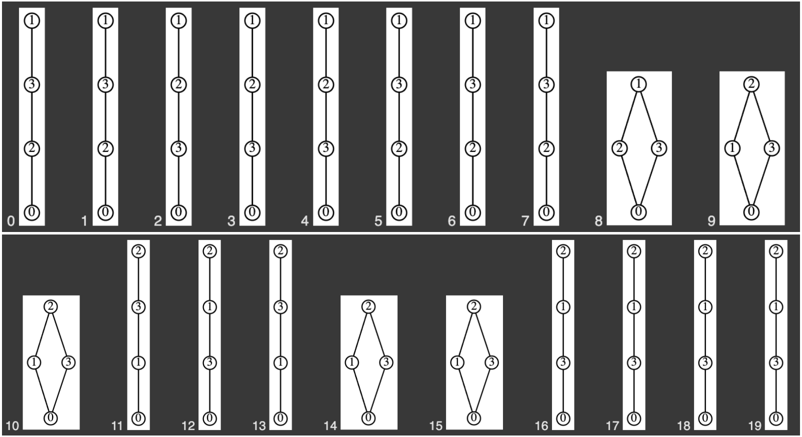
```
!pip install git+https://github.com/jipsen/provers.git
from provers import *

RL = [
  "(x v y)v z = x v(y v z)","x v y = y v x","x v(x^y)=x",
  "(x ^ y)^ z = x ^(y ^ z)","x ^ y = y ^ x","x^(x v y)=x",
  "(x*y)*z = x*(y*z)", "x*1 = x", "1*x = x",
  "(x*y)v z = z <-> y^(x\ z) = y",
  "(x*y)v z = z <-> x^(z/y) = x",
]
Z = ["x v 0 = x"]
a = p9(RL+Z,[],100,0,[6])
```

Output:

```
Number of nonisomorphic models of cardinality 2  is  1
Number of nonisomorphic models of cardinality 3  is  3
Number of nonisomorphic models of cardinality 4  is  20
Number of nonisomorphic models of cardinality 5  is  149
Number of nonisomorphic models of cardinality 6  is  1488
Wall time: 9.41 s
```

Residuated lattices of cardinality 4

```
show(a[4])
```



Residuated lattices up to cardinality 6

RESIDUATED LATTICES OF SIZE UP TO 6

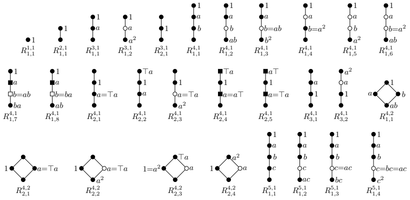
NICK GALATOS AND PETER JIPSEN

There are $1+1+3+20+149+1488 = 1662$ residuated lattices with ≤ 6 elements. In the list below, each algebra is named R_{ij}^{mn} where m is the cardinality and n enumerates nonisomorphic lattices of size m, in order of decreasing height. The depth of the identity element 1 is given by i, and j enumerates nonisomorphic algebras. For lattices of the same height distributive lattices appear before modular lattices, followed by nonmodular lattices, and selfdual lattices appear before nonselfdual lattices. Algebras with more central elements (round circles) are listed earlier, hence commutative residuated lattices precede noncommutative ones. Finally, algebras are listed in decreasing order of number of idempotents (black nodes).

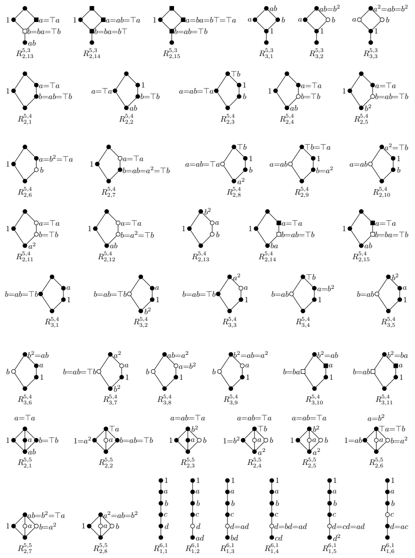
The monoid operation is indicated by labels. If a nonobvious product xy is not listed, then it can be deduced from the given information: either it follows from idempotence ($x^2 = x$) indicated by a black node or from commutativity or there are products $uv = wz$ such that $u \leq x \leq w$ and $v \leq y \leq z$ (possibly $uv = \bot\bot$ or $wz = \top\top$).

If you have comments or notice any issues in this list, please email jipsen.AT.chapman.edu.

● = central idempotent
○ = central nonidempotent
■ = noncentral idempotent
□ = noncentral nonidempotent



Date: April 16, 2017.



DISTRIBUTIVE INVOLUTIVE RESIDUATED LATTICES UP TO CARDINALITY 8

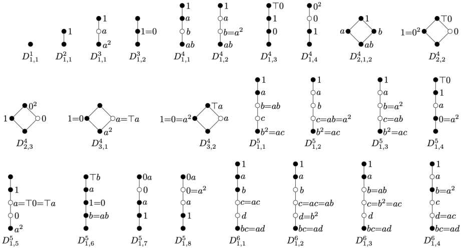
ANDREW CRAIG, PETER JIPSEN, AND CLAUDETTE ROBINSON

There are $1+1+2+9+8+43+49+282 = 395$ distributive involutive residuated lattices with ≤ 8 elements. In the list below, each algebra is named $D^n_{m,i(,k)}$ where n is the cardinality and m enumerates nonisomorphic involutive lattices of size n, in order of decreasing height. The index i enumerates nonisomorphic algebras with the same involutive lattice reduct. The value k indicates the number of nonisomorphic DqRA algebras that have this DInRL as reduct (by default, $k = 1$ is not shown). For $k = 2$, if the algebra is commutative, $\neg x = \sim x = -x$ is one of the DqRAs, and the other one uses an order-reversing isomorphism for $\neg x \neq \sim x$. The linear negations $\sim, -$ are determined by the element labeled 0 (bottom for integral algebras). Algebras with more central elements (round circles) are listed earlier, hence commutative algebras precede noncommutative ones. Finally, algebras are listed in decreasing order of number of idempotents (black nodes).

The monoid operation is indicated by labels. If a nonobvious product xy is not listed, then it can be deduced from the given information: either it follows from idempotence ($x^2 = x$) indicated by a black node or from commutativity or there are products $uv = wz$ such that $u \leq x \leq w$ and $v \leq y \leq z$ (possibly $uv = \bot\bot$ or $wz = \top\top$).

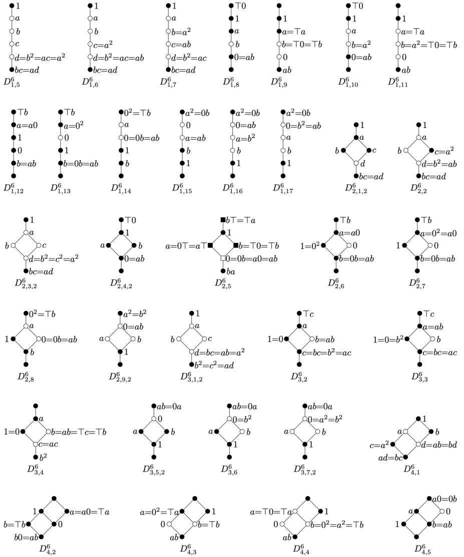
If you have comments or notice any issues in this list, please email `jipsen.AT.chapman.edu`.

● = central idempotent
○ = central nonidempotent
■ = noncentral idempotent
□ = noncentral nonidempotent



Date: May 5, 2025.

1



Computing products of structures in Python

```
def product(self, B):
    base = sorted([[x,y] for x in range(self.cardinality) for y in range (B.cardinality)])
    op = {}
    for f in B.operations:
        fA = self.operations[f]
        fB = B.operations[f]
        if type(fB)==list:
            if type(fB[0])==list:
                op[f] = [[base.index([fA[p[0]][q[0]],fB[p[1]][q[1]]]) for q in base] for p in base]
            else: op[f] = [base.index([fA[p[0]],fB[p[1]]]) for p in base]
        else: op[f] = base.index([fA,fB])
    rel = {}
    for r in B.relations:
        rA = self.relations[r]
        rB = B.relations[r]
        if type(rB[0])==list:
            rel[r] = [[1 if rA[p[0]][q[0]]==1 and rB[p[1]][q[1]]==1 else 0
                          for q in base] for p in base]
        else: rel[r] =[1 if rA[p[0]]==1 and rB[p[1]]==1 else 0 for p in base]
    C = Model(len(base),None,op,rel)
    tupA = {x:(x,) for x in range(self.cardinality)}
    tupB =  {x:(x,) for x in range(B.cardinality)}
    C.tup = {x*B.cardinality+y:tupA[x]+tupB[y] for x in range(self.cardinality) for y in range(B.cardinality)}
    C.elt = {tup[x]:x for x in range(self.cardinality * B.cardinality)}
    return C
```

`Prod([list of algebras])` uses this code to calculate the product of a finite list of finite structures

Poset products and conuclei on products of RLs

Petr Hájek invited me to Soft Computing 2003 in Brno, where I met **Franco Montagna** for the first time.

I gave a talk **An overview of generalized basic logic algebras** and Franco suggested I should visit him during the next summer.

So 24 years ago, on my first trip to Italy, I got to spend 3 weeks in Siena and worked with Franco on the structure of GBL-algebras.

Eventually we developed **poset products** and proved that all finite GBL-algebras are poset products of MV-algebras [J., Montagna 2009]

[Fussner 2022] proved poset products are conuclear images of products

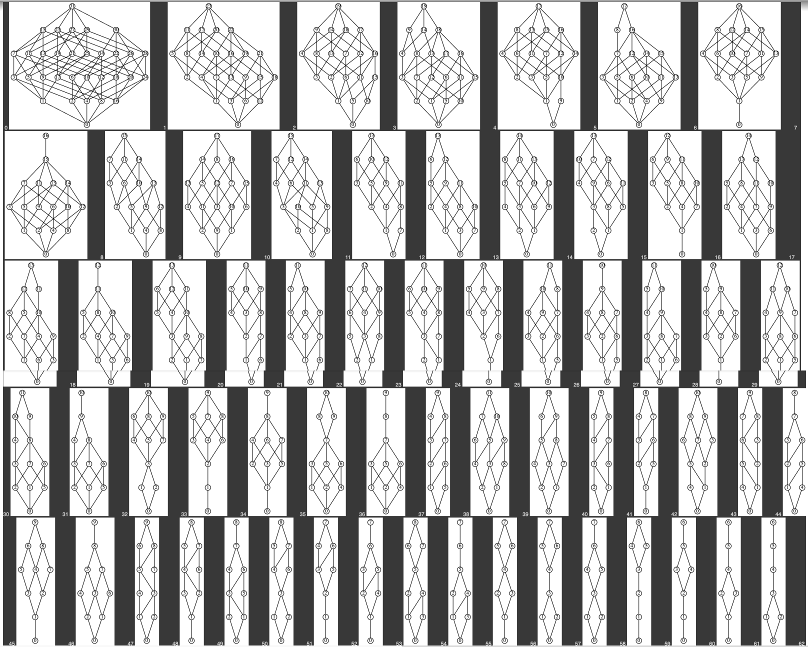
This makes it straightforward to implement them for finite structures.

Conuclear image of a product determined by a poset

```python
def WithConucleus(A, P):
  m = A.cardinality
  n = P.cardinality
  le = P.relations["<="]
  Box = [A.elt[tuple((A.tup[f][x] if all(x==y or not le[y][x]==1 or A.tup[f][y]==1
        for y in range(n)) else 0) for x in range(n))] for f in range(m)]
  A.operations["B"] = Box
  return A
def IdempotentImage(A, c):
  S = list(set(c))
  g = {S[i]:i for i in range(len(S))}
  B = Model(len(S),0,{},{})
  B.tup = [A.tup[f] for f in S]
  B.elt = {tuple(B.tup[i]):i for i in range(len(S))}
  for k in A.relations:
    r = A.relations[k]
    if type(r)==list:
      if type(r[0])==list: B.relations[k] = [[r[x][y] for y in S] for x in S]
      else: B.relations[k] = [r[x] for x in S]
  for k in A.operations:
    o = A.operations[k]
    if type(o)==list:
      if type(o[0])==list: B.operations[k] = [[g[c[o[x][y]]] for y in S] for x in S]
      else: B.operations[k] = [g[c[o[x]]] for x in S]
    else: B.operations[k] = g[c[o]]
  return B
def ConuclearImage(A, P):
  A = WithConucleus(A, P)
  return IdempotentImage(A, A.operations["B"])
```

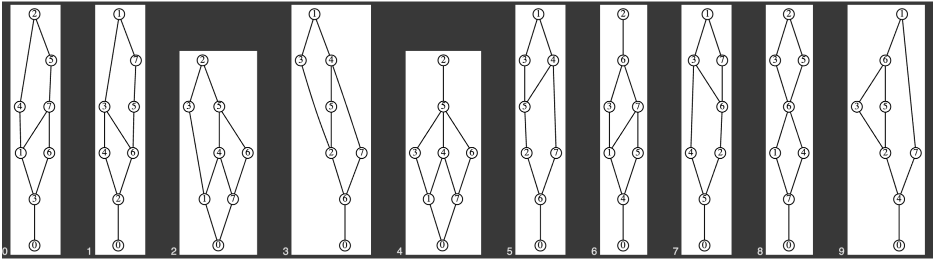
Franco Montagna in Buenos Aires, SLALM 2014

Conuclear images of BA_5 by 5-element posets



These integral residuated lattices are well suited as components for **Plonka sum constructions** of nonintegral residuated lattices and residuated semigroups.

The Python code for Plonka sums is a bit longer (omitted here).



Recently, in joint work with **Simon Santschi**, we used Prover9/Mace4 and Python to find a specific finite V-formation of 3 residuated lattices that Mace4 was not able to amalgamate.

The amalgamation property

A class K of algebras has the **amalgamation property (AP)**

if for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathsf{K}$ and embeddings $f\colon \mathbf{A} \to \mathbf{B}$, $g\colon \mathbf{A} \to \mathbf{C}$

there exists $\mathbf{D} \in \mathsf{K}$ and embeddings $f'\colon \mathbf{B} \to \mathbf{D}$, $g'\colon \mathbf{C} \to \mathbf{D}$ such that

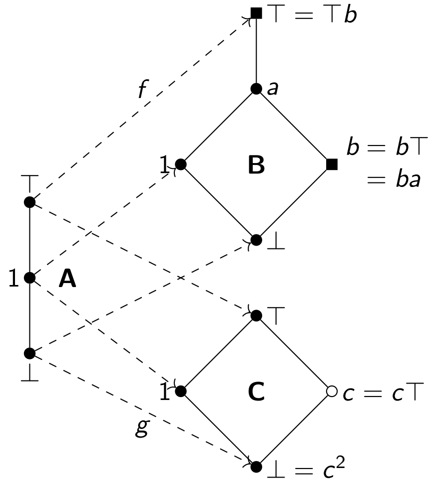
$f' \circ f = g' \circ g.$

$$\begin{tikzcd} & \mathbf{B} \arrow[dr, hook, dashed, "f'"] & \\ \mathbf{A} \arrow[ur, hook, "f"] \arrow[dr, hook, "g"'] & & \mathbf{D} \\ & \mathbf{C} \arrow[ur, hook, dashed, "g'"'] & \end{tikzcd}$$

Does **AP** hold for **all residuated lattices**? (**open since** < 2002)

Kowalski, Takamura [2004]: **AP holds** for commutative RLs.

Theorem: **AP** fails for RL



black = idempotent, round = central

Proof: Straightforward to check **A**, **B**, **C** are RLs and f, g are embeddings.

Assume by contradiction $\exists$ amalgam **D**.

$1 \vee c = \top$ and $1 \vee b = 1 \vee a = a < \top$ hence $g'(c) \neq f'(a)$ and $g'(c) \neq f'(b)$.

So f', g' are inclusions and **B**, **C** $\leq$ **D**

Now, since $c = c\top$ and $\top b = \top$,

in **D** we have $cb = c\top b = c\top = c$.

Moreover $\top = 1 \vee c$ and $c^2 = \bot$,

show $c = \top c = \top bc = (1 \vee c)bc$

$= bc \vee cbc = bc \vee c^2 = bc \vee \bot = bc$

(using $\bot \leq c$ implies $\bot = b\bot \leq bc$).

But also $b = b\top = b(1 \vee c) = b \vee bc$

gives $c = bc \leq b \leq a$. Hence

$\top = 1 \vee c \leq a \vee c = a$; contradiction!

Some remarks

The proof on the previous slide also shows that the **AP** already fails for the variety of **distributive residuated lattices**,

as well as for the $\{\backslash, /\}$-free subreducts of residuated lattices, i.e., for **lattice-ordered monoids**.

Also the proof does not depend on meet or on the constant 1 being in the signature, so the following varieties do not have **AP**:

- **residuated lattice-ordered semigroups**,
- **lattice-ordered semigroups**,
- **residuated join-semilattice-ordered semigroups** and
- **join-semilattice-ordered semigroups** $=$ **additively idempotent semirings**.

Similar examples show that **AP** fails in **idempotent RLs** and in **involutive FL-algebras**.

Definition of relation algebra

Alfred Tarski defined (abstract) relation algebras (RAs) in 1941

A **relation algebra** $\mathbf{A} = \langle A, \sqcup, ^c, ;, 1', ^{-1} \rangle$ is a

1. Boolean algebra $\langle A, \sqcup, ^c \rangle$ with operations $;$, $1'$, $^{-1}$ that satisfy
2. **assoc**: $\forall xyz, (x \,;\, y) \,;\, z = x \,;\, (y \,;\, z)$
3. **rdist**: $\forall xyz, (x \sqcup y) \,;\, z = x \,;\, z \sqcup y \,;\, z$
4. **comp_one**: $\forall x, x \,;\, 1 = x$
5. **conv_conv**: $\forall x, x^{-1-1} = x$
6. **conv_dist**: $\forall xy, (x \sqcup y)^{-1} = x^{-1} \sqcup y^{-1}$
7. **conv_comp**: $\forall xy, (x \,;\, y)^{-1} = y^{-1} \,;\, x^{-1}$
8. **Schroeder**: $\forall xy, x^{-1} \,;\, (x \,;\, y)^c \leq y^c$

A Lean class for relation algebras

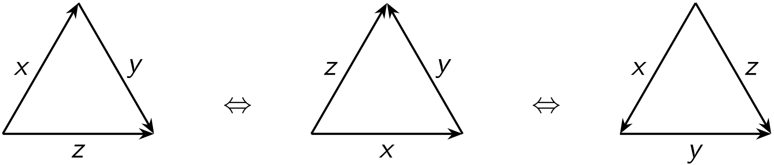
```
class RelationAlgebra (A : Type u) extends
  BooleanAlgebra A, Comp A, One A, Inv A where
  assoc : ∀ x y z : A, (x ; y) ; z = x ; (y ; z)
  rdist : ∀ x y z : A, (x ⊔ y) ; z = x ; z ⊔ y ; z
  comp_one  : ∀ x : A, x ; 1 = x
  conv_conv : ∀ x : A, x⁻¹⁻¹ = x
  conv_dist : ∀ x y : A, (x ⊔ y)⁻¹ = x⁻¹ ⊔ y⁻¹
  conv_comp : ∀ x y : A, (x ; y)⁻¹ = y⁻¹ ; x⁻¹
  schroeder : ∀ x y : A, x⁻¹ ; (x ; y)ᶜ ≤ yᶜ
```

This definition is based on Lean's mathlib4

Properties of relation algebra

Relation algebras satisfy the Peircean law:

$$x;y \sqcap z = \bot \quad \Leftrightarrow \quad z;y^{-1} \sqcap x = \bot \quad \Leftrightarrow \quad x^{-1};z \sqcap y = \bot$$



$$x;y \leq z^c \quad \Leftrightarrow \quad x \leq (z;y^{-1})^c \quad \Leftrightarrow \quad y \leq (x^{-1};z)^c$$

$$x;y \leq z \quad \Leftrightarrow \quad x \leq (z^c;y^{-1})^c \quad \Leftrightarrow \quad y \leq (x^{-1};z^c)^c$$

Definitions for binary relations: Math vs. Lean

Let X be a set and $R, S, T \in \mathcal{P}(X \times X)$ **binary relations** on X

```
import Mathlib.Data.Set.Basic
variable {X : Type u} (R S T : Set (X × X))
```

Define **composition** $R\,;S = \{(x, y) \mid \exists z,\ (x, z) \in R \wedge (z, y) \in S\}$.

```
def composition (R S : Set (X × X)) : Set (X × X) :=
  { (x, y) | ∃ z, (x, z) ∈ R ∧ (z, y) ∈ S }
```

Define the **inverse** of R by $R^{-1} = \{(y, x) \mid (x, y) \in R\}$

```
def inverse (R: Set (X×X)): Set (X×X) := {(y,x)|(x,y)∈R}
```

```
infixl:90 " ; "  => composition
postfix:100 "⁻¹" => inverse
```

A simple proof about representable RAs

```
lemma comp_assoc : (R ; S) ; T = R ; (S ; T) := by
  rw [Set.ext_iff]
  intro (a,b)
  constructor
  intro h
  rcases h with ⟨z, h₁, _⟩
  rcases h₁ with ⟨x,_,_⟩
  use x
  constructor
  trivial
  use z
  intro h₂
  rcases h₂ with ⟨x, h₃, h₄⟩
  rcases h₄ with ⟨y,_,_⟩
  use y
  constructor
  use x
  trivial
```

A database of finite integral relation algebras up to 5 atoms

An *atom* in a BA is a smallest element $\neq \bot$.

An atom a in a RA is *symmetric* if $a = a^{-1}$.

Let a, b, c, d be symmetric atoms $(x^{-1} = x)$ and r, s nonsymmetric

The number of RAs up to isomorphism is given below:

Cardinality	2	4	8	8	16	16	32	32	32
Atoms	$1'$	$1'a$	$1'rr^{-1}$	$1'ab$	$1'arr^{-1}$	$1'abc$	$1'rr^{-1}ss^{-1}$	$1'abrr^{-1}$	$1'abcd$
Number of RAs	1	2	3	7	37	65	83	1316	3013
Representable	1	2	3	7	26	45	**39**	**298**	**480**
Nonrepresent.	0	0	0	0	11	20	**29**	**783**	**2048**
Unknown	0	0	0	0	0	0	**15**	**235**	**485**

For the list of 83 there are 15 RAs not known to be (non)representable:

30,31,32,40,44,45,54,56,59,60,61,63,65,69,79 (see [Maddux 2006])

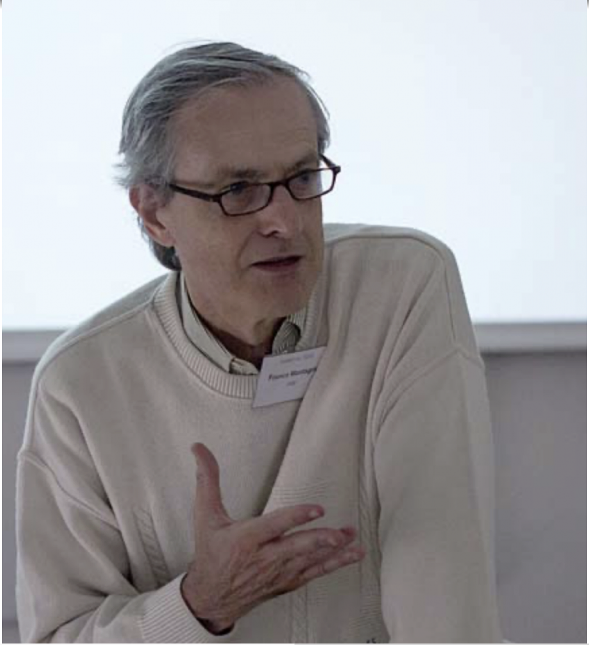
Conclusion

Residuated lattices and relation algebras can be formalized in Lean

Prover9/Mace4 and Python can find proofs or construct examples

An important application of proof assistants is to formalize results that are recorded in mathematical databases.

Thank you!



References

- A. Craig, P. Jipsen, C. Robinson: *Distributive involutive residuated lattices up to cardinality 8*, 2025, https://github.com/jipsen/Distributive-quasi-relation-algebras-and-DInFL/blob/main/DInFL1.pdf
- W. Fussner: *Poset Products as Relational Models*. Stud Logica 110, 95–120 (2022). https://doi.org/10.1007/s11225-021-09956-z
- N. Galatos, P. Jipsen: *Residuated lattices up to size 6*, 2017, https://math.chapman.edu/~jipsen/preprints/RLlist3.pdf
- P. Jipsen, F. Montagna: *The Blok-Ferreirim theorem for normal GBL-algebras and its application*. Algebra Universalis, 60, (2009), 381–404.
- L∃∀N Programming Language and Theorem Prover, https://lean-lang.org/
- L∃∀N Community and MathLib, https://leanprover-community.github.io/
- R. Maddux: *Relation Algebras.* Elsevier, Studies in Logic and the Foundations of Mathematics, Vol 150 (2006).
- W. McCune, L. Wos: *Otter*, Journal of Automated Reasoning, 18, (1997), 211–220.
- W. McCune: Prover9 and Mace4. https://www.cs.unm.edu/~mccune/prover9/, 2005–2010