Representability and formalization of relation algebras

Peter Jipsen and Pace Nelson

Chapman University

Annual Meeting of Association of Symbolic Logic

ASL 2025, May 13 – 16

New Mexico State University, Las Cruzes

Introduction

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

When is a RA **representable** as an algebra of binary relations?

Donald Monk (1964): the variety of representable RAs is **not axiomatized by finitely many formulas**.

Robin Hirsch and Ian Hodkinson (2001): it is **undecidable** whether a finite relation algebra is representable.

Roger Maddux (1983): n-dimensional bases to **prove nonrepresentability**.

Steve Comer (∼1980): one-point extension method to **prove representability** for some small RAs.

Finding and checking these proofs by hand is laborious but could now be done with the help of proof assistants.

Relation algebra 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`

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

Definition of relation algebra

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

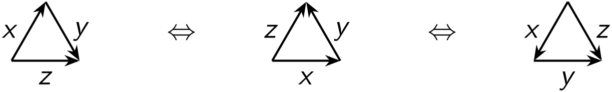
```
class RelationAlgebra (A : Type u) extends
  BooleanAlgebra A, Comp A, One A, Inv A where
  assoc : ∀ x y z : A, (x ; y) ; z = x ; (y ; z)
  rdist : ∀ x y z : A, (x ⊔ y) ; z = x ; z ⊔ y ; z
  comp_one  : ∀ x : A, x ; 1 = x
  conv_conv : ∀ x : A, x⁻¹⁻¹ = x
  conv_dist : ∀ x y : A, (x ⊔ y)⁻¹ = x⁻¹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

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



```
lemma top_conv : (⊤ : A)⁻¹ = ⊤ := by
  have : (⊤ : A)⁻¹ = (⊤ ⊔ ⊤⁻¹)⁻¹ := by simp
  have : (⊤ : A)⁻¹ = ⊤⁻¹ ⊔ ⊤ := by rw [conv_dist,
    conv_conv] at this; exact this
  have : (⊤ : A) ≤ ⊤⁻¹ := by rw [left_eq_sup] at this;
    exact this
  exact top_unique this

lemma ldist (x y z : A) : x ; (y ⊔ z) = x ; y ⊔ x ; z :=
    by
  calc
    x ; (y ⊔ z) = (x ; (y ⊔ z))⁻¹⁻¹ := by rw [conv_conv]
    _ = ((y ⊔ z)⁻¹ ; x⁻¹)⁻¹ := by rw [conv_comp]
    _ = ((y⁻¹ ⊔ z⁻¹) ; x⁻¹)⁻¹ := by rw [conv_dist]
    _ = (y⁻¹ ; x⁻¹ ⊔ z⁻¹ ; x⁻¹)⁻¹ := by rw [rdist]
    _ = ((x ; y)⁻¹ ⊔ (x ; z)⁻¹)⁻¹ := by rw [←conv_comp,
    ←conv_comp]
    _ = (x ; y) ⊔ (x ; z) := by rw [←conv_dist,
    conv_conv]
```

```
lemma comp_le_comp_right (z : A) {x y : A} (h : x ≤ y) :
    x ; z ≤ y ; z := by
  calc
    x ; z ≤ x ; z ⊔ y ; z := by simp
    _ = (x ⊔ y) ; z := by rw [←rdist]
    _ = y ; z := by simp [h]

lemma comp_le_comp_left (z : A) {x y : A} (h : x ≤ y) : z
    ; x ≤ z ; y := by
  calc
    z ; x ≤ z ; x ⊔ z ; y := by simp
    _ = z ; (x ⊔ y) := by rw [←ldist]
    _ = z ; y := by simp [h]

lemma conv_le_conv {x y : A} (h : x ≤ y) : x⁻¹ ≤ y⁻¹ :=
    by
  calc
    x⁻¹ ≤ x⁻¹ ⊔ y⁻¹ := by simp
    _ = (x ⊔ y)⁻¹ := by rw [←conv_dist]
    _ = y⁻¹ := by simp [h]
```

```
lemma conv_compl_le_compl_conv (x : A) : x⁻¹ᶜ ≤ xᶜ⁻¹ := by
  have : x ⊔ xᶜ = ⊤ := by simp
  have : (x ⊔ xᶜ)⁻¹ = ⊤⁻¹ := by simp
  have : x⁻¹ ⊔ xᶜ⁻¹ = ⊤ := by rw [conv_dist, top_conv]
    at this; exact this
  rw[join_eq_top_iff_compl_le] at this; exact this

lemma conv_compl_eq_compl_conv (x : A) : xᶜ⁻¹ = x⁻¹ᶜ := by
  have : x⁻¹⁻¹ᶜ ≤ x⁻¹ᶜ⁻¹ := conv_compl_le_compl_conv x⁻¹
  have : xᶜ ≤ x⁻¹ᶜ⁻¹ := by rw [conv_conv] at this; exact
    this
  have : xᶜ⁻¹ ≤ x⁻¹ᶜ⁻¹⁻¹ := conv_le_conv this
  rw [conv_conv] at this; exact le_antisymm this
    (conv_compl_le_compl_conv x)
```

```
lemma one_conv_eq_one : (1 : A)⁻¹ = 1 := by
  calc
    (1 : A)⁻¹ = 1⁻¹ ; 1 := by rw [comp_one]
    _ = (1⁻¹ ; 1)⁻¹⁻¹ := by rw [conv_conv]
    _ = (1⁻¹ ; 1⁻¹⁻¹)⁻¹ := by rw [conv_comp]
    _ = (1⁻¹ ; 1)⁻¹ := by rw [conv_conv]
    _ = 1 := by rw [comp_one, conv_conv]

lemma one_comp (x : A) : 1 ; x = x := by
  calc
    1 ; x = (1 ; x)⁻¹⁻¹ := by rw [conv_conv]
    _ = (x⁻¹ ; 1⁻¹)⁻¹ := by rw [conv_comp]
    _ = (x⁻¹ ; 1)⁻¹ := by rw [one_conv_eq_one]
    _ = x⁻¹⁻¹ := by rw [comp_one]
    _ = x := by rw [conv_conv]
```

```
lemma peirce_law1 (x y z : A) :
  x ; y ⊓ z = ⊥ ↔ x⁻¹ ; z ⊓ y = ⊥ := by
  constructor
  · intro h
    have : x ; y ≤ zᶜ := by rw [meet_eq_bot_iff_le_compl]
    at h; exact h
    have : z ≤ (x ; y)ᶜ := by rw [←compl_le_compl_iff_le,
     compl_compl] at this; exact this
    have : x⁻¹ ; z ≤ x⁻¹ ; (x ; y)ᶜ := comp_le_comp_left
    x⁻¹ this
    have : x⁻¹ ; z ⊓ y ≤ ⊥ := by calc
        x⁻¹ ; z ⊓ y ≤ x⁻¹ ; (x ; y)ᶜ ⊓ y :=
    inf_le_inf_right y this
        _ ≤ yᶜ ⊓ y := inf_le_inf_right y (schroeder x y)
        _ = ⊥ := by simp
    exact bot_unique this
```

```
  · intro h
    have : x⁻¹ ; z ≤ yᶜ := by rw
    [meet_eq_bot_iff_le_compl] at h; exact h
    have : y ≤ (x⁻¹ ; z)ᶜ := by
      rw [←compl_le_compl_iff_le, compl_compl] at this;
    exact this
    have : x⁻¹⁻¹ ; y ≤ x⁻¹⁻¹ ; (x⁻¹ ; z)ᶜ :=
    comp_le_comp_left x⁻¹⁻¹ this
    have : x⁻¹⁻¹ ; y ⊓ z ≤ ⊥ := by calc
      x⁻¹⁻¹ ; y ⊓ z ≤ x⁻¹⁻¹ ; (x⁻¹ ; z)ᶜ ⊓ z :=
    inf_le_inf_right z this
      _ ≤ zᶜ ⊓ z := inf_le_inf_right z (schroeder x⁻¹ z)
      _ = ⊥ := by simp
    have : x ; y ⊓ z ≤ ⊥ := by rw [conv_conv] at this;
    exact this
    exact bot_unique this

lemma peirce_law2 (x y z : A) :
  x ; y ⊓ z = ⊥ ↔ z ; y⁻¹ ⊓ x = ⊥ := by
  ...
```

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 $R^{-1} = \{(y, x) \mid (x, y) \in R\}$

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

```
theorem 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
```

Algebras of binary relations

An *algebra of binary relations* is a set of relations closed under the operations $\cup, \cap, ^c, ;, ^{-1}, 1'$.

Can prove the axioms of RAs hold for algebras of binary relations.

A relation algebra is **representable** if it is isomorphic to an algebra of binary relations.

Roger Lyndon [1956] found axioms that hold in all algebras of relations but not in all relation algebras.

Shortest axioms of Roger Lyndon

J: $t \le u; v \sqcap w; x$ and $u^{-1}; w \sqcap v; x^{-1} \le y; z$
$$\implies t \le (u; y \sqcap w; z^{-1}); (y^{-1}; v \sqcap x; z)$$

L: $x; y \sqcap z; w \sqcap u; v \le$
$x; (x^{-1}; u \sqcap y; v^{-1} \sqcap (x^{-1}; z \sqcap y; w^{-1}); (z^{-1}; u \sqcap w; v^{-1})); v$

M: $t \sqcap (u \sqcap v; w); (x \sqcap y; z) \le$
$v; ((v^{-1}; t \sqcap w; x); z^{-1} \sqcap w; y \sqcap v^{-1}; (u; y \sqcap t; z^{-1})); z$

```
theorem Jtrue : t ⊆ u;v ∩ w;x  ∧  u⁻¹;w ∩ v;x⁻¹ ⊆ y;z
    → t ⊆ (u;y ∩ w;z⁻¹);(y⁻¹;v ∩ z;x) := by
  intro h
  intro (a,b)
  intro h₁
  rcases h with ⟨h₂,h₃⟩
  have h₄ : (a, b) ∈ u ; v ∩ w ; x :=
    Set.mem_of_mem_of_subset h₁ h₂
  rcases h₄ with ⟨h₅, h₆⟩
  rcases h₅ with ⟨c, h₇, h₈⟩
  rcases h₆ with ⟨d, h₉, H₁⟩
  have H₂ : (c, a) ∈ u⁻¹ := by rw [inv]; dsimp; trivial
  have H₃ : (c, d) ∈ u⁻¹ ; w := by use a
  have H₄ : (b, d) ∈ x⁻¹ := by rw [inv]; dsimp; trivial
  have H₅ : (c, d) ∈ v ; x⁻¹ := by use b
  have H₆ : (c, d) ∈ u⁻¹ ; w ∩ v ; x⁻¹ := by
    constructor; trivial; trivial
  have H₇ : (c, d) ∈ y ; z := Set.mem_of_mem_of_subset H₆
    h₃
  rcases H₇ with ⟨e, H₈, H₉⟩
  ...
```

```
theorem Ltrue :
  x;y ∩ z;w ∩ u;v ⊆ x;((x⁻¹;z ∩ y;w⁻¹);(z⁻¹;u ∩ w;v⁻¹) ∩
     x⁻¹;u ∩ y;v⁻¹);v := by
  intro (a,b)
  intro h
  rcases h with ⟨h1, h2⟩
  rcases h1 with ⟨h3,h4⟩
  rcases h3 with ⟨e, h3, h5⟩
  rcases h4 with ⟨d, h3, h4⟩
  rcases h2 with ⟨c, h6, h7⟩
  use c
  constructor
  use e
  constructor
  trivial
  constructor
  constructor
  use d
  constructor
  constructor
  ...
```

```
theorem Mtrue :
  t ∩ (u ∩ v ; w) ; (x ∩ y;z) ⊆ v;((v⁻¹;t ∩ w;x);z⁻¹ ∩
    w;y ∩ v⁻¹;(u;y ∩ t;z⁻¹));z := by
  intro (a,b)
  intro h
  rcases h with ⟨h1,h2⟩
  rcases h2 with ⟨c,h1,h2⟩
  rcases h1 with ⟨h3,h4⟩
  rcases h4 with ⟨d,h5,h6⟩
  rcases h2 with ⟨h7,h8⟩
  rcases h8 with ⟨e,h9,h10⟩
  use e
  constructor
  use d
  constructor
  trivial
  constructor
  constructor
  use b
  constructor
  ...
```

Ralph McKenzie's 16-element relation algebra

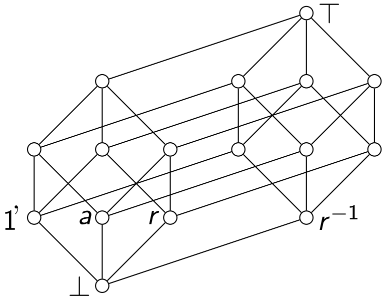
This algebra is named 14_{37} in Roger Maddux's book [4]

It is a **nonrepresentable** RA of smallest cardinality

with four atoms: $1', a, r, r^{-1}$ and top element $\top = 1' \sqcup a \sqcup r \sqcup r^{-1}$

;	$1'$	a	r	r^{-1}
$1'$	a	a	r	r^{-1}
a	a	$1' \sqcup r \sqcup r^{-1}$	$a \sqcup r$	$a \sqcup r^{-1}$
r	r	$a \sqcup r$	r	$\top$
r^{-1}	r^{-1}	$a \sqcup r^{-1}$	$\top$	r^{-1}

All 16 elements of McKenzie's algebra



McKenzie's algebra in Lean (as an atom structure)

```
inductive M : Type | e : M | a : M | r : M | r₁ : M
open M
def M.ternary : M → M → M → Prop := fun
| e, e, e => True | e, a, a => True | e, r, r => True
| e, r₁, r₁ => True | a, e, a => True | a, a, e => True
| a, a, r => True | a, a, r₁ => True | a, r, a => True
| a, r, r => True | a, r₁, a => True | a, r₁, r₁ => True
| r, e, r => True | r, a, a => True | r, a, r => True
| r, r, r => True | r, r₁, e => True | r, r₁, a => True
| r, r₁, r => True | r, r₁, r₁ => True | r₁, e, r₁ => True
| r₁, a, a => True | r₁, a, r₁ => True | r₁, r, e => True
| r₁, r, a => True | r₁, r, r => True | r₁, r, r₁ => True
| r₁, r₁, r₁ => True | _, _, _ => False
def M.inv : M → M := fun | e => e | a => a | r => r₁ |
    r₁ => r
def M.unary : M → Prop := fun | e => True | _ => False
```

McKenzie's algebra is nonrepresentable

Theorem [5] *McKenzie's algebra* 14_{37} *is not representable.*

Proof. The formula M fails in this algebra:

Let $t = a, u = r, v = a, w = a, x = r^{-1}, y = a, z = a$.

From the table we see $u \sqcap v; w = r \sqcap a; a = r \sqcap (1' \sqcup r \sqcup r^{-1}) = r$

and $x \sqcap y; z = r^{-1} \sqcap a; a = r^{-1} \sqcap (1' \sqcup r \sqcup r^{-1}) = r^{-1}$.

Hence the LHS $= a \sqcap r; r^{-1} = a \sqcap (1' \sqcup a \sqcup r \sqcup r^{-1}) = a$.

However the RHS $= a; ((a; a \sqcap a; r^{-1}); a \sqcap a; a \sqcap a; (r; a \sqcap a; a)); a$

$= a; (r^{-1}; a \sqcap a; a \sqcap a; r); a = a; \bot; a = \bot$ □

A database of finite integral relation algebras up to 5 atoms

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:

$1'$	$1'a$	$1'rr^{-1}$	$1ab$	$1'arr^{-1}$	$1'abc$	$1'rr^{-1}ss^{-1}$	$1'abrr^{-1}$	$1'abcd$
1	2	3	7	37	65	83	1316	3013

Their (non)representability has been decided up to size 16.

These results could benefit from formalization.

For the list of 83 there are 15 RAs that are not known to be (non)representable: 30,31,32,40,44,45,54,56,59,60,61,63,65,69,79 (see [4])

Conclusion

Relation algebras can be formalized in Lean using readable syntax

Algebras of binary relations can prove properties like J, L, M

A search for relational bases can be used to find deeper reasons for nonrepresentability

A compelling application of proof assistants is to formalize results that are recorded in mathematical databases.

References

- L∃∀N Programming Language and Theorem Prover, `https://lean-lang.org/`
- L∃∀N Community and MathLib, `https://leanprover-community.github.io/`
- R. Hirsch, I. Hodkinson: *Relation Algebras by Games.* North Holland/Elsevier Vol 147 (2002)
- R. Maddux, *Relation Algebras.* Elsevier Vol 150 (2006).
- R. McKenzie, *Representations of integral relation algebras.* Michigan Math. J., 17, (1970), 279–287.
- W. McCune, L. Wos, *Otter*, Journal of Automated Reasoning, 18, (1997), 211–220.

THANKS!