Theory of Algebraic Structure in Proof Assistant Systems

Akshobhya Katte Madhusudana

Supervisor: Dr. Jacques Carette

Department of Computing and Software

McMaster University

Contents

Introduction

Algebraic structure consists of a set A, a collection of operations on A, and a finite set of axioms, that these operations must satisfy.

Abstract algebra is the name that is commonly given to the study of algebraic structures. The general theory of algebraic structures has been formalized in universal algebra.

Introduction

Proof assistant system: Proof assistants are software tool to assist with the development of formal proofs by human-machine collaboration

Agda is a dependently typed programming language and a proof assistant system.

Agda has been used in various applications such as formal verification, program synthesis, theorem proving, and automated reasoning.

Agda standard library includes many useful definitions and theorems about basic data structures, such as natural numbers, lists, and vectors.

Definition

Equivalence Relation: A relation R on set X is a subset on $X \times X$ is equivalence if it is *reflexive*, *symmetric* and *transitive*.

- A relation R is ***reflexive*** if $R: \{(x, x): x \in X\}$
- A relation R is ***symmetric*** if $R: \forall x, y \in X{:}\ xRy \Longleftrightarrow\ yRx$
- *A relation R is **transitive*** if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$

Function: If in a relation, if every element in domain is mapped to only one element in the codomain, then we call it a function.

- A function f is ***injective*** if $f(x) = f(y) \Rightarrow\ x = y$.
- A function is called ***surjective*** if given $y \in Y$, there exists $x \in X$ such that $f(x) = y$.
- A function is called ***bijective*** if it is both injective and surjective.

Definition

Type: The type (or language) of the algebra is a set of function symbols. Each member of this set is assigned a positive number that is the arity of the member.

Morphism: If *A* and *B* are two algebras of same type *F*, then a homomorphism is defined as a mapping α from *A* to *B* such that: $\alpha f^{A}(a_1, a_2, \dots, an) = f^{B}(\alpha a_1, \alpha a_2, \dots, \alpha an)$

- For two algebras *A* and *B*, if $\alpha: A \to B$ is a homomorphism, and if *α* satisfies one-to-one mapping then the morphism *α* is called a ***monomorphism***
- For two algebras *A* and *B*, if $\alpha: A \to B$ is a monomorphism, and if *α* is a bijection from *A* to *B*, then *α* is called an ***isomorphism***.

Composition: For algebras *A*, *B*, and *C* the composition of morphisms $f: A \to B$ and $g: B \to C$ is denoted by the function $g \circ f: A \to C$ and is defined as $(g \circ f)a = g(fa), \forall a \in A$.

Proof Systems:

- Agda 2:Agda standard library – v1.7.1
- Coq: Mathematical components – 1.12.0
- Idris 2 – library code
- Lean 3 – Mathlib – 3.4.2

Experiment:

- Create a web crawler to skim the source code.
- Create a clickable table that takes to definition of the structures in the source code.

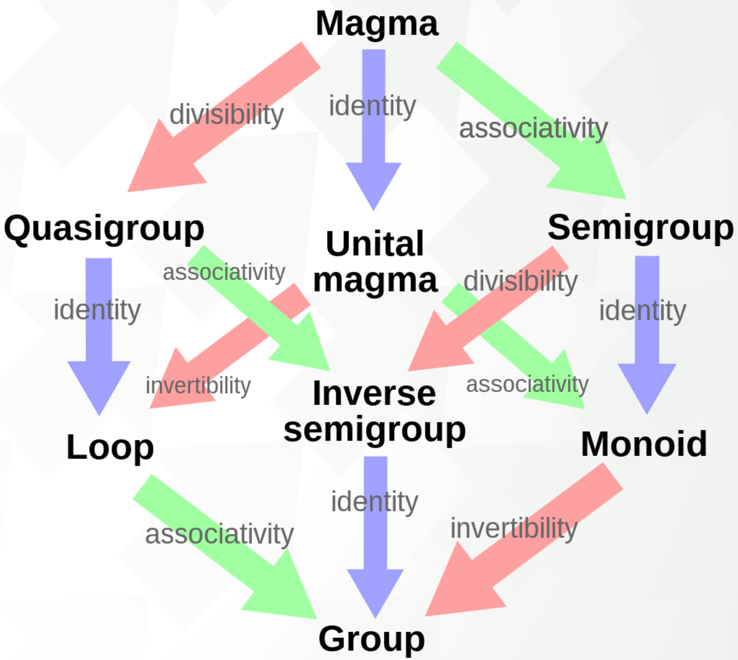
A ***magma*** has a set equipped with a single binary operation that must be closed by definition.

A ***quasigroup*** can be defined as a magma with left and right division identities

$$y = x \cdot (x \backslash y)$$
$$y = x \backslash (x \cdot y)$$
$$y = (y / x) \cdot x$$
$$y = (y \cdot x) / x$$

A ***loop*** is a quasigroup that has identity element. The identity axiom is given as:

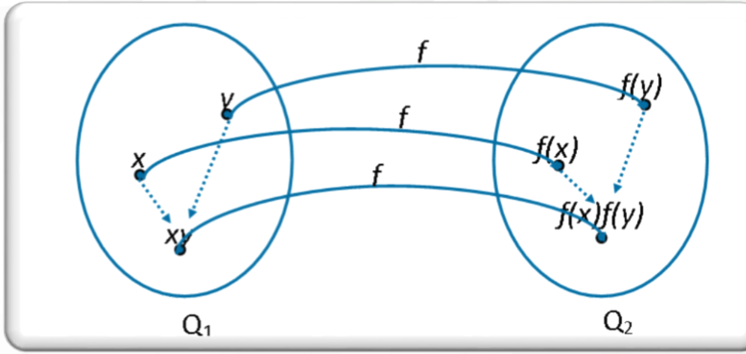
$$x \cdot e = e \cdot x = x$$



Homomorphism of quasigroup and loop:

```
record IsQuasigroupHomomorphism (⟦_⟧ : A → B) : Set (a ⊔ ℓ₁ ⊔ ℓ₂) where
  field
    isRelHomomorphism : IsRelHomomorphism _≈₁_ _≈₂_ ⟦_⟧
    ∙-homo           : Homomorphic₂ ⟦_⟧ _∙₁_ _∙₂_
    \\-homo           : Homomorphic₂ ⟦_⟧ _\\₁_ _\\₂_
    //-homo           : Homomorphic₂ ⟦_⟧ _//₁_ _//₂_
```

```
record IsLoopHomomorphism (⟦_⟧ : A → B) : Set (a ⊔ ℓ₁ ⊔ ℓ₂) where
  field
    isQuasigroupHomomorphism : IsQuasigroupHomomorphism ⟦_⟧
    ε-homo                   : Homomorphic₀ ⟦_⟧ ε₁ ε₂
```



Properties of Quasigroup

- Quasigroup is cancellative.
- If $x \cdot y = z$ then $y = x \backslash z$
- *If* $x \cdot y = z$ then $x = z / y$

Properties of Loop

- $x / x = e$
- $x \backslash x = e$
- $e \backslash x = x$
- $x / e = x$

Properties of Middle bol loop

- $x \cdot ((y \cdot x) \backslash x) = y \backslash x$
- $x \cdot ((x \cdot z) \backslash x) = x / z$
- $x \cdot (z \backslash x) = (x / z) \cdot x$
- $(x / (y \cdot z)) \cdot x = (x / z) \cdot (y \backslash x)$
- $(x / (y \cdot x)) \cdot x = y \backslash x$
- $(x / (x \cdot z)) \cdot x = x / z$

Properties of Moufang Loop

- Moufang loop is alternative.
- Moufang loop is flexible.
- $z \cdot (x \cdot (z \cdot y)) = ((z \cdot x) \cdot z) \cdot y$
- $x \cdot (z \cdot (y \cdot z)) = ((x \cdot z) \cdot y) \cdot z$
- $z \cdot ((x \cdot y) \cdot z) = (z \cdot (x \cdot y)) \cdot z$

Semigroup:
A semigroup is a Magma with associative property.

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

Commutative Semigroup:
A semigroup that satisfies commutative property is called commutative semigroup.

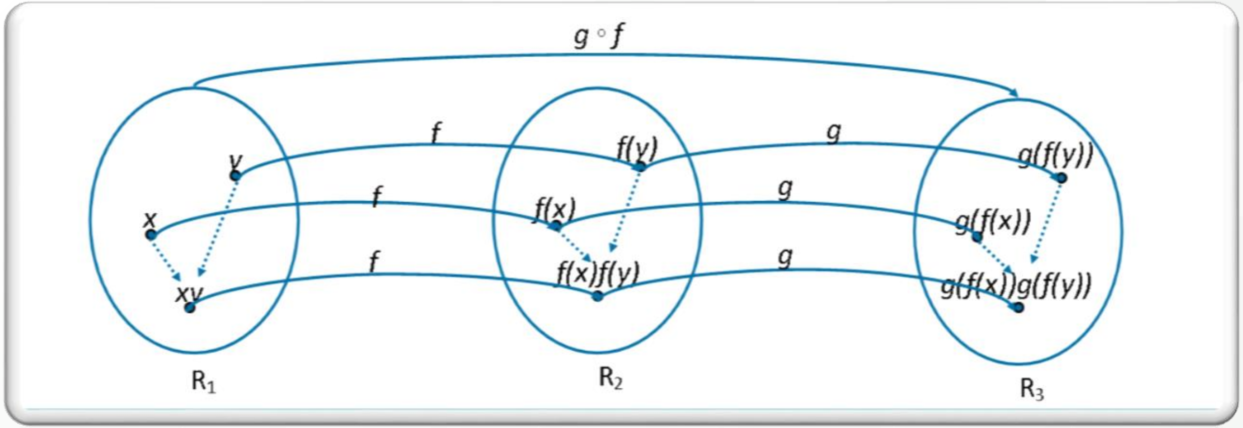
$$x \cdot y = y \cdot x$$

Ring $(\boldsymbol{R}, +, *, ^{-1}, \boldsymbol{0}, \boldsymbol{1})$

- + is an AbelianGroup:
  - · Associativity: $x + (y + z) = (x + y) + z$
  - · Identity: $(x + 0) = x = (0 + x)$
  - · Inverse: $(x + x^{-1}) = 0 = (x^{-1} + x)$
- * is a monoid
  - · Associativity: $x * (y * z) = (x * y) * z$
  - · Identity: $(x * 1) = x = (1 * x)$
- Multiplication distributes over addition:
  - · Left distributes $(x * (y + z)) = (x * y) + (x * z)$
  - · Right distributes $(x + y) * z = (x * z) + (y * z)$
- Annihilating zero: $(x * 0) = 0 = (0 * x)$

Composition of homomorphism is homomorphic:

- $g \circ f(x \cdot_1 y) = g\big(f(x) \cdot_2 f(y)\big) = g\big(f(x)\big) \cdot_3 g\big(f(y)\big) = g \circ f(x) \cdot_3 g \circ f(y)$
- $g \circ f(e_1) = g(e_2) = e_3$
- $g \circ f(x^{-1}) = g(x^{-1}) = x^{-1}$



Properties of semigroup

- Semigroup is alternative
- Semigroup is flexible
- $(x \cdot y) \cdot (x \cdot x) = x \cdot (y \cdot (x \cdot x))$

Properties of commutative semigroup

- Left semimedial: $(x \cdot x) \cdot (y \cdot z) = (x \cdot y) \cdot (x \cdot z)$
- Right semimedial: $(y \cdot z) \cdot (x \cdot x) = (y \cdot x) \cdot (z \cdot x)$
- Middle semimedial: $(x \cdot y) \cdot (z \cdot x) = (x \cdot z) \cdot (y \cdot x)$

Properties of Ring without one

- $-(x * y) = -x * y$
- $-(x * y) = x * -y$

Properties of Ring

- $-1 * x = -x$
- $x + x = 0 \Rightarrow x = 0$
- $x * (y - z) = x * y - x * z$
- $(y - z) * x = (y * x) - (z * x)$

Idempotent semiring:

- Addition + is an idempotent commutative monoid:
  - · Associativity: $x+(y+z)=(x+y)+z$
  - · Identity: $(x+0)=x=(0+x)$
  - · Commutativity: $(x+y)=(y+x)$
  - · Idempotent: $(x+x)=x$
- Multiplication · is a monoid:
  - · Associativity: $x\cdot(y\cdot z)=(x\cdot y)\cdot z$
  - · Identity: $(x\cdot 1)=x=(1\cdot x)$
- Addition distributes over multiplication :
  - · Left distributive: $(x\cdot(y+z))=(x\cdot y)+(x\cdot z)$
  - · Right distributive: $(x+y)\cdot z=(x\cdot z)+(y\cdot z)$
- Annihilating zero: $(x\cdot 0)=0=(0\cdot x)$

A ***Kleene Algebra*** is an idempotent semiring with * operator such that:

- $1+(x\cdot(x^*))\leq x^*$
- $1+(x^*)\cdot x\leq x^*$
- $b+a\cdot x\leq x\Rightarrow(a^*)\cdot b\leq x$
- $b+x\cdot a\leq x\Rightarrow b\cdot(a^*)\leq x$

Properties of Kleene Algebra:

- $0^* = 1$
- $1^* = 1$
- $1 + x^* = x$
- $x + x * x^* = x^*$
- $x + x^* * x = x^*$
- $x + x^* = x^*$
- $1 + x + x^* = x^*$
- $0 + x + x^* = x^*$
- $x^* * x^* = x^*$
- $x^{**} = x^*$
- $x = y \Rightarrow x^* = y^*$
- $a * x = x * b \Rightarrow a^* * x = x * b^*$
- $(x * y)^* * x = x * (y * x)^*$

Problem in Programming Algebra

Analyze 5 problems in programming algebra:

- Equivalent but structurally different e.g. Quasigroups
- Ambiguity in naming e.g. Ring and Rng, Nearring (*-semigroup/*-monoid).
- Redundant field in structural inheritance: e.g. semiring (+-commutativeMonoid and *-monoid).
- Identical structures e.g. Nearring
- Equivalent structures e.g. Bounded semilattice and Idempotent commutative monoid

Problem in Programming Algebra

1. Ambiguity – Ring vs Rng

```
record IsRing (+ * : Op₂ A) (-_ : Op₁ A) (0# 1# : A) : Set
(a ⊔ ℓ) where
  field
     +-isAbelianGroup : IsAbelianGroup + 0# -_
     *-cong           : Congruent₂ *
     *-assoc          : Associative *
     *-identity       : Identity 1# *
     distrib          : * DistributesOver +
     zero             : Zero 0# *
```

2. Equivalent but structurally different: Quasigroups

```
record IsQuasigroup (· : Op₂ A ) : Set (a ⊔ ℓ)
where
    field
    isMagma    : IsMagma ·
    LatinSquare : LatinSquare ·
```

```
record IsQuasigroup (· \\ // : Op₂ A) :
Set (a ⊔ ℓ) where
    field
        isMagma     : IsMagma ·
        \\-cong      : Congruent₂ \\
        //-cong      : Congruent₂ //
        leftDivides  : LeftDivides · \\
        rightDivides: RightDivides · //
```

Problem in Programming Algebra

3. Redundancy- When a structure is defined in terms of two or more structures there is a possibility of redundancy

```
record IsRing (+ * : Op₂ A) (-_ : Op₁ A) (0# 1# : A) :
Set (a ⊔ ℓ) where
  field
    +-isAbelianGroup : IsAbelianGroup + 0# -_
    *-isMonoid       : IsMonoid * 1#
```

```
record IsRing (+ * : Op₂ A) (-_ : Op₁ A) (0# 1# : A) :
Set (a ⊔ ℓ) where
  field
    +-isAbelianGroup : IsAbelianGroup + 0# -_
    *-cong           : Congruent₂ *
    *-assoc          : Associative *
    *-identity       : Identity 1# *
```

4. Identical structures– Same algebraic structure can be defined from two or more different structures

```
record IsNearring (+ * : Op₂ A) (0# 1# : A)
(_⁻¹ : Op₁ A) : Set (a ⊔ ℓ) where
  field
        isQuasiring : IsQuasiring + * 0# 1#
        +-inverse    : Inverse 0# _⁻¹ +
```

```
record IsNearring (+ * : Op₂ A) (0# 1# : A)
(_⁻¹ : Op₁ A) : Set (a ⊔ ℓ) where
    field
        +-isGroup   : IsGroup + 0# -_
        *-isMonoid : IsMonoid * 1#
```

Problem in Programming Algebra

5. Equivalent structures- Same algebraic structure can have various names.

```
record IsIdempotentCommutativeMonoid (· : Op₂ A)
(ε : A) : Set (a ⊔ ℓ) where
  field
        isCommutativeMonoid : IsCommutativeMonoid · ε
        idem                : Idempotent ·
```

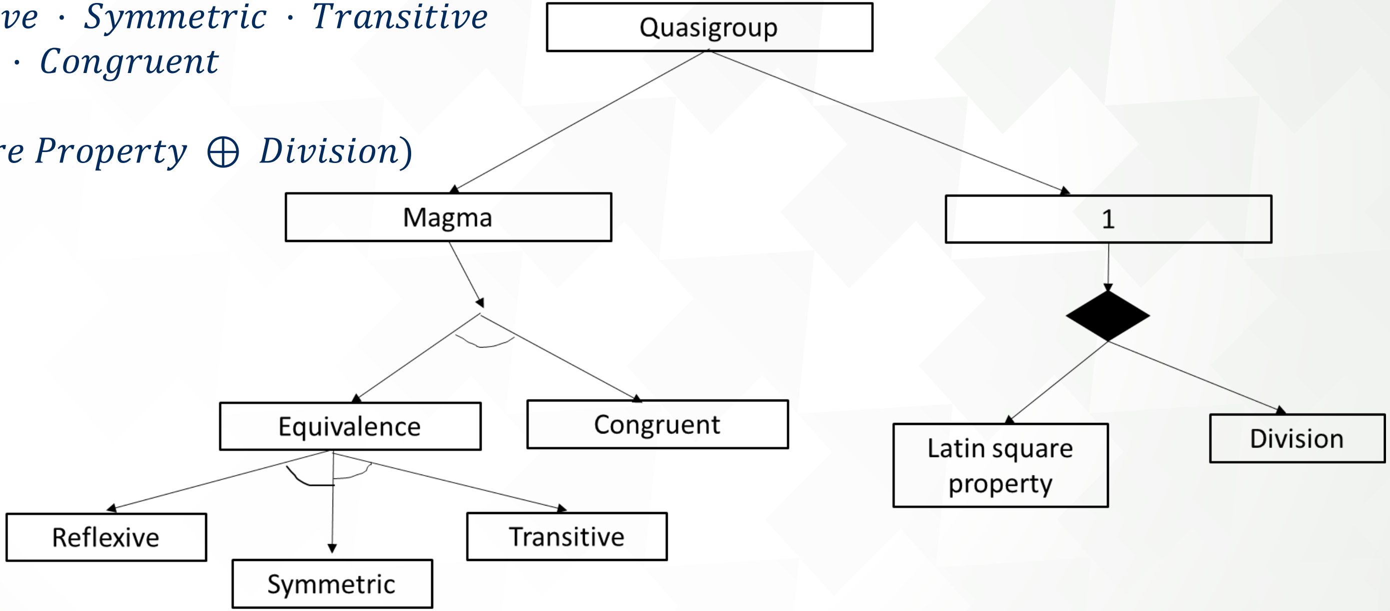
```
IsBoundedSemilattice =
                  IsIdempotentCommutativeMonoid

module IsBoundedSemilattice {· ε}
  (L : IsBoundedSemilattice · ε) where
    open IsIdempotentCommutativeMonoid L public
```

Problem in Programming Algebra

Product Family Algebra:

- $Equivalence = Reflexive \cdot Symmetric \cdot Transitive$
- $Magma = Equivalence \cdot Congruent$
- $Quasigroup =$
  $Magma \cdot (LatinSquare\ Property \oplus Division)$



Conclusion
- Define the scope by survey
- Theory of Algebraic structures in Agda
- Analyze problems that arise

Future work
- Extend product family algebra
- Generated libraries to standard library
- More concrete definitions of constructs

Questions?