Theory of Algebraic Structure in Proof Assistant Systems

Akshobhya Katte Madhusudana

Supervisor: Dr. Jacques Carette

Department of Computing and Software
McMaster University

Contents

- □Introduction
- □Definitions
- □Algebraic structure in proof systems - Survey
- □Theory of Quasigroup and Loop
- □Theory of Semigroup and Ring
- □Theory of Kleene Algebra
- □Problem in programming algebra
- □Conclusion

Introduction

Introduction
Definition
Survey
Quasigroup and Loop
Semigroup and Ring
Kleene Algebra
Programming algebra
Conclusion

By the early 19th century, mathematicians had discovered how to solve polynomial equation of up-to degree 4. Galois used group to determine the solvability of polynomial equations.

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

Introduction

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

Why proof?

- A proof convinces the reader that the statement is correct.
- A proof explains why the statement is correct.

We contribute to the Agda standard library, a proof assistant system, so it can be extended to other relevant fields of algebra.

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 $(g \ \circ \ f\ )a = g(fa), \forall a \in A$.

Algebraic Structures in Proof Systems - Survey

Proof Systems:

- Agda 2:Agda standard library
- Coq: Mathematical components
- Idris 2 – library code
- Lean 3 – Mathlib

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.

Theory of Quasigroup and Loop

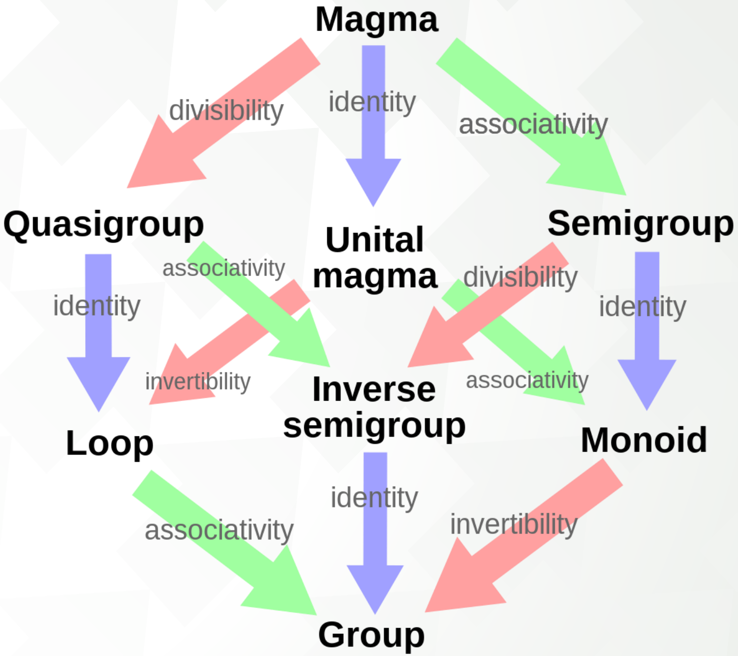
A ***magma*** has a set equipped with a single binary operation that must be closed by definition.

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

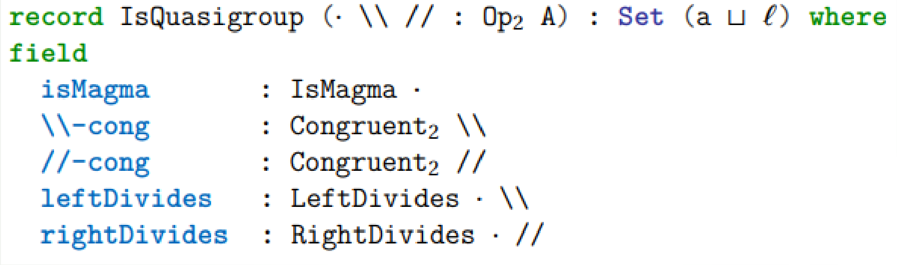
$$y = x \cdot (x \backslash y)$$
$$y = x \backslash (x \cdot y)$$
$$y = (y / x) \cdot x$$
$$y = (y \cdot x) / x$$

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

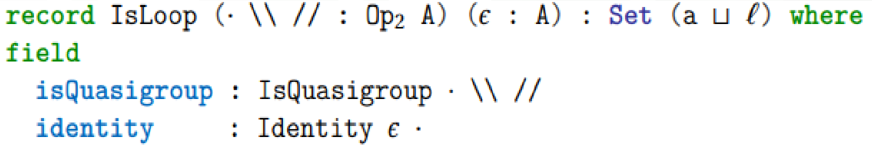
$$x \cdot e = e \cdot x = x$$



Theory of Quasigroup and Loop

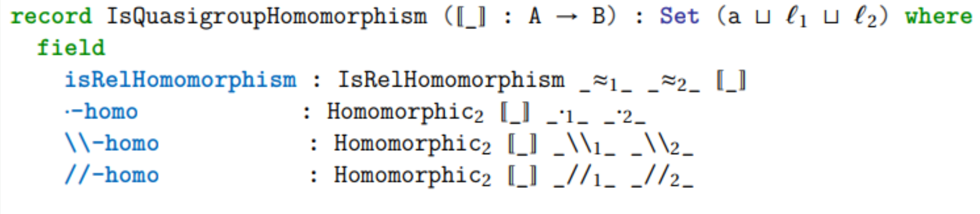


```
record IsQuasigroup (· \\ // : Op₂ A) : Set (a ⊔ ℓ) where
field
  isMagma       : IsMagma ·
  \\-cong       : Congruent₂ \\
  //-cong       : Congruent₂ //
  leftDivides   : LeftDivides · \\
  rightDivides  : RightDivides · //
```

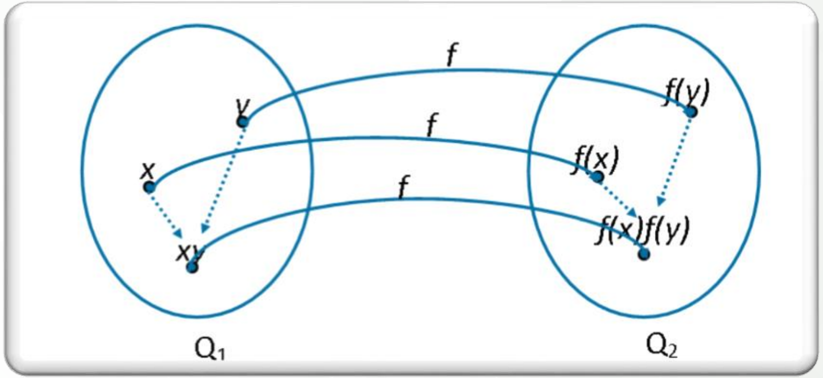


```
record IsLoop (· \\ // : Op₂ A) (ε : A) : Set (a ⊔ ℓ) where
field
  isQuasigroup : IsQuasigroup · \\ //
  identity     : Identity ε ·
```

Homomorphism of quasigroup



```
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_
```



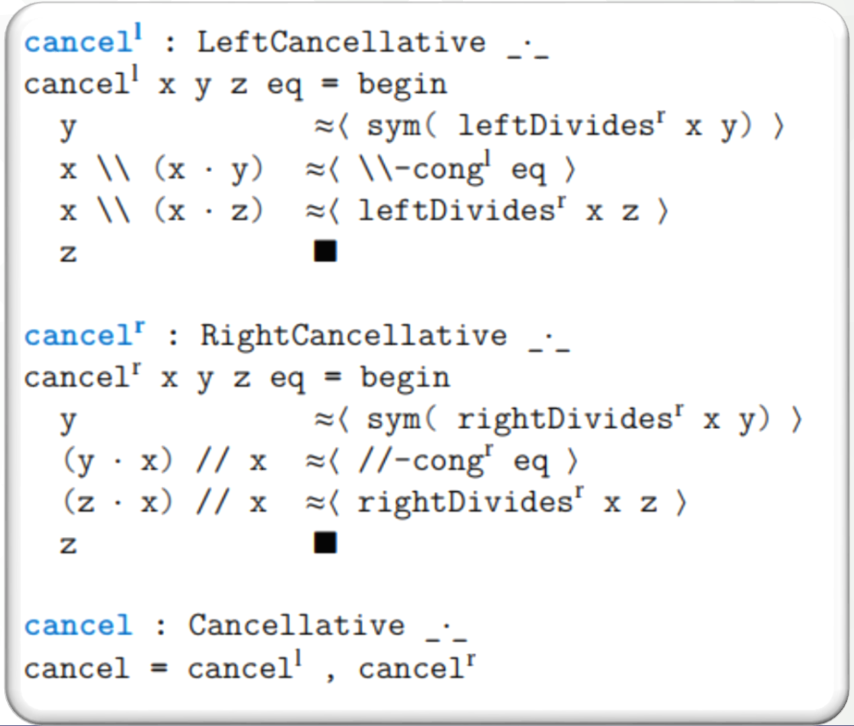
Theory of Quasigroup and Loop

Properties of Quasigroup

- Left cancellative: $x \cdot y = x \cdot z \Rightarrow y = z$
- Right cancellative: $y \cdot x = z \cdot x \Rightarrow y = z$
- $x \cdot y = z \;\Rightarrow y = x\backslash \mathrm{z}$
- $x \cdot y = z \Rightarrow x = z/y$

Properties of Loop

- $x/x = e$
- $x\backslash \mathrm{x} = e$
- $e\backslash \mathrm{x} = x$
- $x/e = x$



```
cancelˡ : LeftCancellative _·_
cancelˡ x y z eq = begin
  y              ≈⟨ sym( leftDividesʳ x y) ⟩
  x \\ (x · y)   ≈⟨ \\-congˡ eq ⟩
  x \\ (x · z)   ≈⟨ leftDividesʳ x z ⟩
  z              ∎

cancelʳ : RightCancellative _·_
cancelʳ x y z eq = begin
  y              ≈⟨ sym( rightDividesʳ x y) ⟩
  (y · x) // x   ≈⟨ //-congʳ eq ⟩
  (z · x) // x   ≈⟨ rightDividesʳ x z ⟩
  z              ∎

cancel : Cancellative _·_
cancel = cancelˡ , cancelʳ
```

Theory of Quasigroup and Loop

A loop is called middle bol loop if it satisfies the identity

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

Properties of Middle bol loop:

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

Theory of Quasigroup and Loop

A loop is called a ***right bol loop*** if:

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

A loop is called a ***left bol loop*** if:

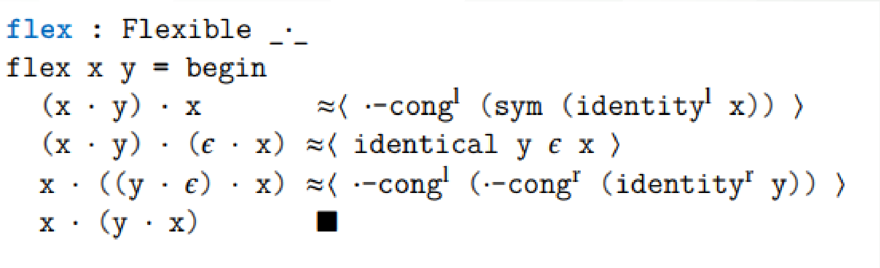
$$x \cdot (y \cdot (x \cdot z)) = (x \cdot (y \cdot x)) \cdot z$$

A left-right bol loop is called a ***moufang loop*** if:

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

Properties of Moufang Loop

- Left alternative: $(x \cdot x) \cdot y = x \cdot (x \cdot y)$
- Right alternative: $x \cdot (y \cdot y) = (x \cdot y) \cdot y$
- Flexible: $(x \cdot y) \cdot x = x \cdot (y \cdot x)$
- $z \cdot (x \cdot (z \cdot y)) = ((z \cdot x) \cdot z) \cdot y$
- $x \cdot (z \cdot (y \cdot z)) = ((x \cdot z) \cdot y) \cdot z$
- $z \cdot ((x \cdot y) \cdot z) = (z \cdot (x \cdot y)) \cdot z$



```
flex : Flexible _∙_
flex x y = begin
  (x ∙ y) ∙ x        ≈⟨ ∙-congˡ (sym (identityˡ x)) ⟩
  (x ∙ y) ∙ (ε ∙ x) ≈⟨ identical y ε x ⟩
  x ∙ ((y ∙ ε) ∙ x) ≈⟨ ∙-congˡ (∙-congʳ (identityʳ y)) ⟩
  x ∙ (y ∙ x)        ∎
```

Theory of Semigroup and Ring

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

- + 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: $(x * 0) = 0 = (0 * x)$

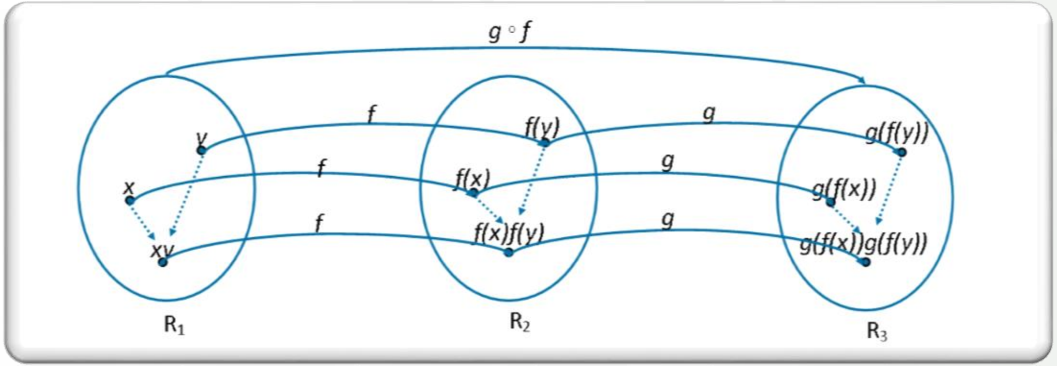
Theory of Semigroup and Ring

Composition of homomorphism is homomorphic:

- $$\begin{aligned} g \circ f(x \cdot_1 y) &= g\big(f(x) \cdot_2 f(y)\big) \\ &= g\big(f(x)\big) \cdot_3 g\big(f(y)\big) \\ &= g \circ f(x) \cdot_3 g \circ f(y) \end{aligned}$$

- $$\begin{aligned} g \circ f(e_1) &= g(e_2) \\ &= e_3 \end{aligned}$$

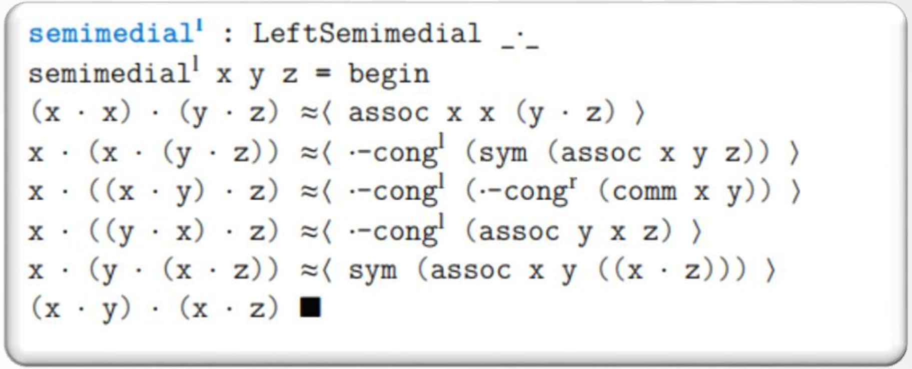
- $$\begin{aligned} g \circ f(x^{-1}) &= g(x^{-1}) \\ &= x^{-1} \end{aligned}$$



Theory of Semigroup and Ring

Properties of semigroup

- Left alternative: $(x \cdot x) \cdot y = x \cdot (x \cdot y)$
- Right alternative: $x \cdot (y \cdot y) = (x \cdot y) \cdot y$
- Flexible: $(x \cdot y) \cdot x = x \cdot (y \cdot x)$
- $(x \cdot y) \cdot (x \cdot x) = x \cdot (y \cdot (x \cdot x))$



```
semimedialˡ : LeftSemimedial _·_
semimedialˡ x y z = begin
(x · x) · (y · z) ≈⟨ assoc x x (y · z) ⟩
x · (x · (y · z)) ≈⟨ ·-congˡ (sym (assoc x y z)) ⟩
x · ((x · y) · z) ≈⟨ ·-congˡ (·-congʳ (comm x y)) ⟩
x · ((y · x) · z) ≈⟨ ·-congˡ (assoc y x z) ⟩
x · (y · (x · z)) ≈⟨ sym (assoc x y ((x · z))) ⟩
(x · y) · (x · z) ∎
```

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

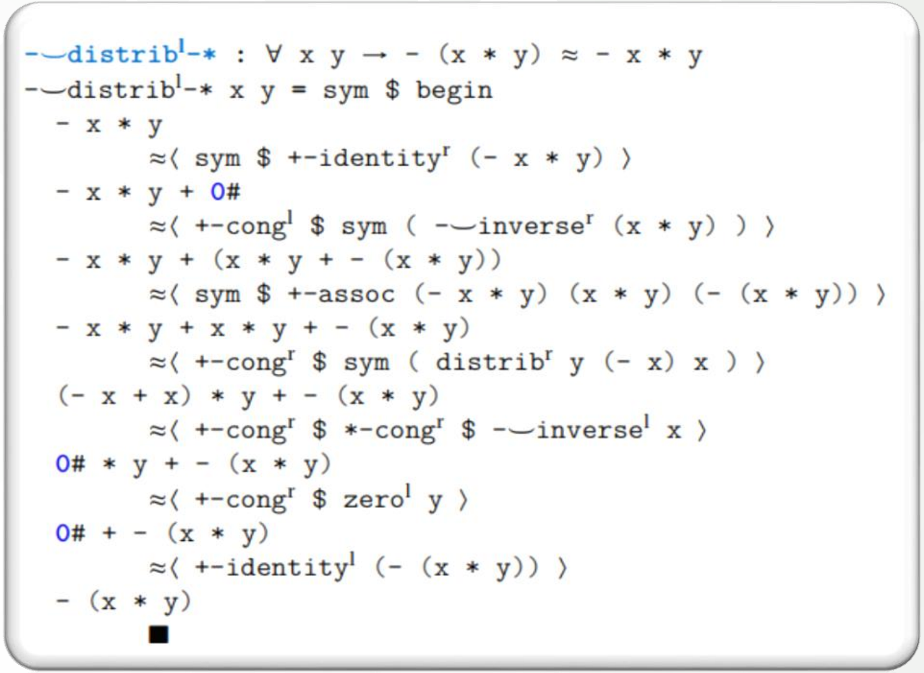
Theory of Semigroup and Ring

Properties of Ring without one

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

Properties of Ring

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



```
-‿distribˡ-* : ∀ x y → - (x * y) ≈ - x * y
-‿distribˡ-* x y = sym $ begin
  - x * y
        ≈⟨ sym $ +-identityʳ (- x * y) ⟩
  - x * y + 0#
        ≈⟨ +-congˡ $ sym ( -‿inverseʳ (x * y) ) ⟩
  - x * y + (x * y + - (x * y))
        ≈⟨ sym $ +-assoc (- x * y) (x * y) (- (x * y)) ⟩
  - x * y + x * y + - (x * y)
        ≈⟨ +-congʳ $ sym ( distribʳ y (- x) x ) ⟩
  (- x + x) * y + - (x * y)
        ≈⟨ +-congʳ $ *-congʳ $ -‿inverseˡ x ⟩
  0# * y + - (x * y)
        ≈⟨ +-congʳ $ zeroˡ y ⟩
  0# + - (x * y)
        ≈⟨ +-identityˡ (- (x * y)) ⟩
  - (x * y)
        ∎
```

Theory of Kleene Algebra

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: $(x \cdot 0) = 0 = (0 \cdot x)$

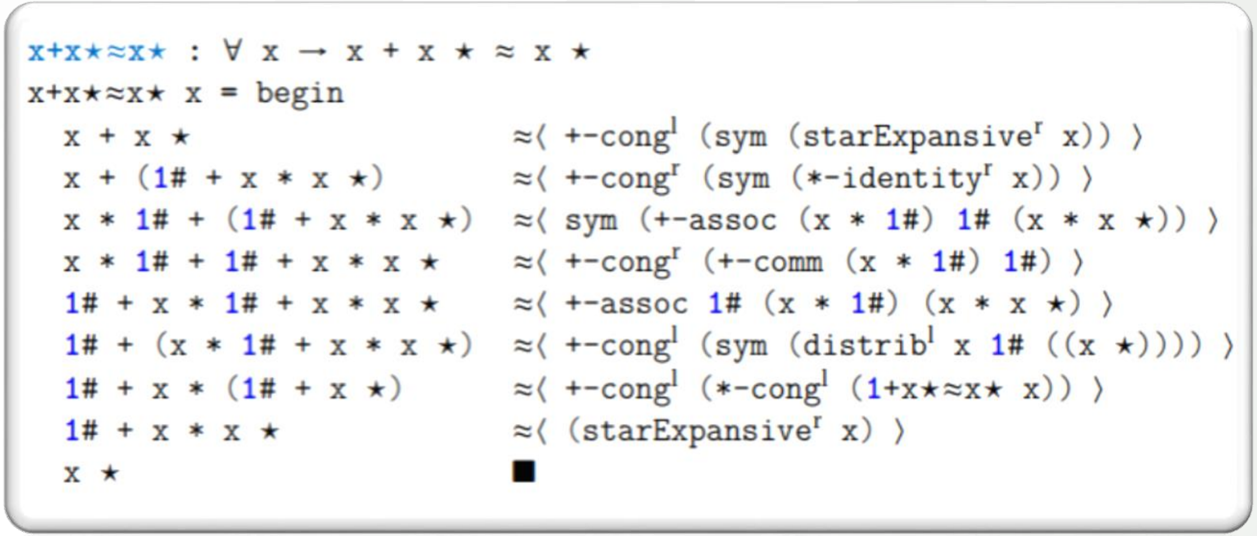
A ***Kleene Algebra*** is an idempotent semiring with * operator such that:

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

Theory of Kleene Algebra

Properties of Kleene Algebra:

- $0^* = 1$
- $1^* = 1$
- $1 + x^* = x^*$
- $x + x * x^* = x^*$
- $x + x^* * x = x^*$
- $x + x^* = x^*$



```
x+x★≈x★ : ∀ x → x + x ★ ≈ x ★
x+x★≈x★ x = begin
  x + x ★                   ≈⟨ +-congˡ (sym (starExpansiveʳ x)) ⟩
  x + (1# + x * x ★)        ≈⟨ +-congʳ (sym (*-identityʳ x)) ⟩
  x * 1# + (1# + x * x ★)   ≈⟨ sym (+-assoc (x * 1#) 1# (x * x ★)) ⟩
  x * 1# + 1# + x * x ★     ≈⟨ +-congʳ (+-comm (x * 1#) 1#) ⟩
  1# + x * 1# + x * x ★     ≈⟨ +-assoc 1# (x * 1#) (x * x ★) ⟩
  1# + (x * 1# + x * x ★)   ≈⟨ +-congˡ (sym (distribˡ x 1# ((x ★)))) ⟩
  1# + x * (1# + x ★)       ≈⟨ +-congˡ (*-congˡ (1+x★≈x★ x)) ⟩
  1# + x * x ★              ≈⟨ (starExpansiveʳ x) ⟩
  x ★                       ∎
```

Theory of Kleene Algebra

Properties of Kleene Algebra:

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

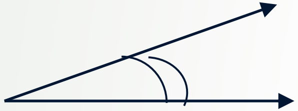
Problem in Programming Algebra

Analyze 5 problems in programming algebra:

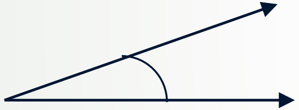
- Equivalent but structurally different e.g. Quasigroups
- Ambiguity in naming e.g. Ring and Rng, Nearring (*-semigroup/*-monoid).
- Redundant field in structural inheritance: e.g. semiring (+-commutativeMonoid and *-monoid).
- Identical structures e.g. Nearring
- Equivalent structures e.g. Bounded semilattice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

Problem in Programming Algebra

Product Family Algebra:

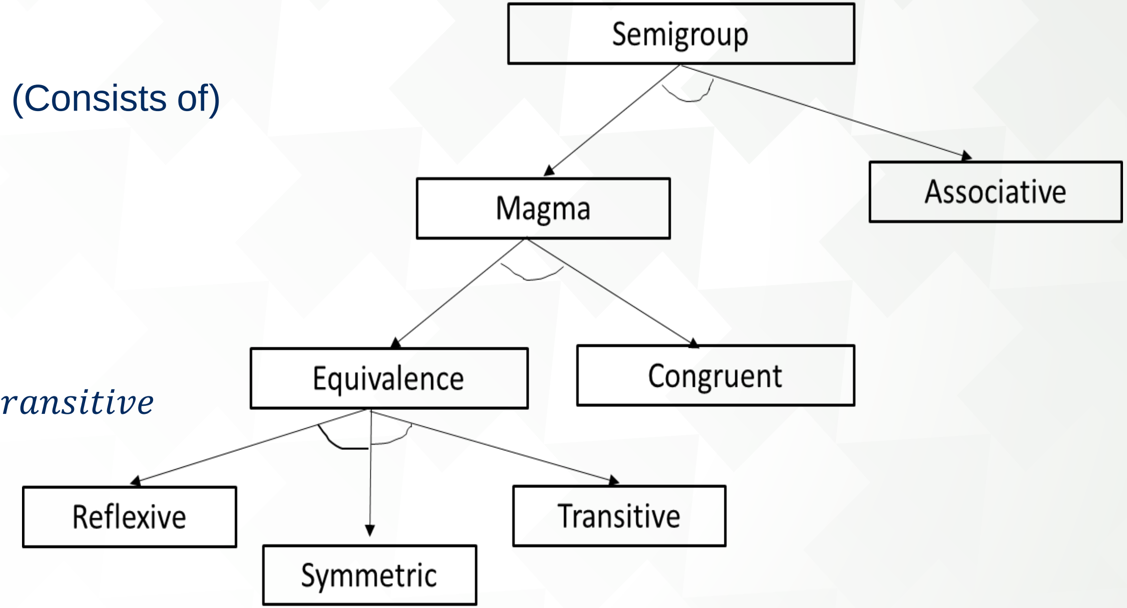


- AND decomposition (Consists of)



- OR decomposition

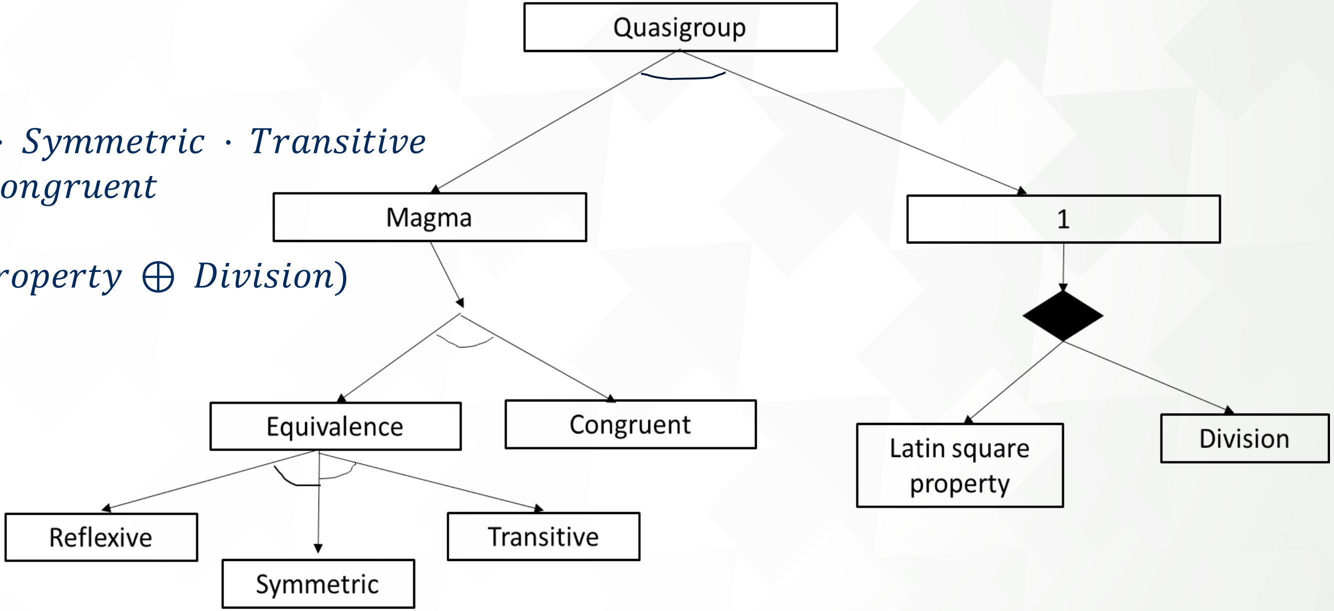
- $Equivalence = Reflexive \cdot Symmetric \cdot Transitive$
- $Magma = Equivalence \cdot Congruent$
- $Semigroup = Magma \cdot Associativity$



Problem in Programming Algebra

- XOR decomposition

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



Conclusion

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?