Universal Algebra in HoTT

Andreas Aagaard Lyngs
201710283

Supervised by
Bas Spitters

A thesis presented for the degree of
Bachelor of Mathematics

Department of Mathematics
Aarhus University
Denmark
17th January 2019

Abstract - english résumé

This report presents a universal algebra development in Coq for the Homotopy Type Theory (HoTT) library. Developments of universal algebra in Type Theory are commonly using setoids to support subsets. Setoids are best avoided because they complicate the implementation. This report shows that setoids are not needed in homotopy type theoretic universal algebra. The development in this report contains definitions of subalgebra, product algebra and quotient algebra. These definitions are verified for correctness using category theoretic techniques. Later they are used to prove the three isomorphism theorems, which can be seen as a milestone. A key theorem of the development shows that isomorphic algebras are in fact equal in HoTT. We therefore obtain equalities from the isomorphism theorems.

Abstract - dansk resumé

Denne rapport præsenterer en universel algebra implementering i Coq for Homotopi-type-teori (HoTT) biblioteket. Implementeringer af universel algebra i type-teori bruger ofte setoids til at understøtte delmængder. Setoids bør undgås fordi de komplicerer implementeringen. Denne rapport viser at setoids ikke er nødvendige i homotopi-type-teoretisk universel algebra. Implementeringen i denne rapport indeholder definitioner af under-algebra, produkt-algebra og kvotient-algebra. Disse definitioner er verificeret for korrekthed ved brug af kategori-teoretiske teknikker. Senere er definitionerne brugt til at bevise de tre isomorfi sætninger, hvilket kan anses som en milepæl. En nøglesætning i implementeringen viser at isomorfe algebraer er lig med hinanden i HoTT. Vi opnår derfor ligheder gennem isomorfi sætningerne.

Contents

Introduction

In this report I present the beginning of a universal algebra development in Coq for the Homotopy Type Theory (HoTT) library [1]. The Coq formalisation of this is located at `https://github.com/andreaslyn/hott-classes`. This is the first formalisation of universal algebra in HoTT. The work is based on the Math Classes library due to B. Spitters and E. van der Weegen [2], which was originally developed to serve as a basis for constructive analysis in Coq.

Universal algebra is important to mathematics because it provides general results about algebraic strucures. The isomorphism theorems in universal algebra are generalisations of the isomorphism theorems known from group theory and ring theory. In universal algebra, these theorems apply to a wide range of algebraic structures including groups and rings and even groups acting on sets, hence proving these theorems once and for all. In computer science, universal algebra is used to characterise algebraic data types (known from functional languages) as initial algebras in specific categories of algebras. Birkhoff used universal algebra to study regular languages as algebras [3].

Part I of this report introduces some background theory: Category Theory, universal algebra, and HoTT. The reader is assumed familiar with type theory. Part II presents the results of the universal algebra development for the Coq HoTT library.

The main literature I have studied during this development is:

- Type Classes for Mathematics in Type Theory by B. Spitters and E. van der Weegen [2].
- The HoTT Library: A formalization of homotopy type theory in Coq by A. Bauer and J. Gross and P. LeFanu Lumsdaine and M. Shulman and M. Sozeau and B. Spitters [1].
- Isomorphism is equality by T. Coquand and N. A. Danielsson [1].
- Chapter 1-3 and Chapter 6 of Homotopy Type Theory: Univalent Foundations of Mathematics (the HoTT book) [4].
- Chapter II of A Course in Universal Algebra by B. Stanley and H. P. Sankappanavar [5].

Problem

Universal algebra has been formalised in Coq by B. Spitters and E. van der Weegen [2], and in Agda by E. Gunther and A. Gadea and M. Pagano [6]. In order to implement quotient types and function extensionality, these developments are relying on setoids, a type together with an equivalence relation. Setoids complicate the theory because functions of setoids are required to respect the equivalence relations, and existing theorems relying on strict equality do not apply to setoids. Also, users of the library obtain results about setoids, which forces them to rely on setoids to some extent. This may escalate and add complexity to other developments as well.

The univalence axiom in HoTT implies function extensionality and higher inductive types can be used to define quotient types without the need for setoids.

In this report I develop universal algebra in HoTT using higher inductive types, so without relying on setoids. Section 6.3 contains a homotopy type theoretic definition of quotient algebra. A convenient practice in set theoretic foundations is to view isomorphic

objects as being equal. A key result, Theorem 5.6, states that isomorphic algebras are literally equal in HoTT. This is used in Section 7 to obtain equalities from the isomorphism theorems.

Part I
Background

1 Category Theory

This section introduces elementary notions from category theory. Readers familiar with category theory can safely skip this section. The section is based on Steve Awodeys category theory book [7]. Throughout the section we will be working in a set theoretical foundation (assuming large categories).

1.1 Definitions

Definition 1.1. A *category* $\mathbf{C}$ consists of

- a collection of *objects* $\mathbf{C}_0$,
- a collection of *morphisms* $\mathbf{C}_1$.

It is required that:

- For each $f \in \mathbf{C}_1$ there are objects $\mathrm{dom}(f) \in \mathbf{C}_0$ and $\mathrm{cod}(f) \in \mathbf{C}_0$ called the *domain* and *codomain* of f.
- There is a binary *composition* operator $\circ$ defined for morphisms $f \in \mathbf{C}_1$ and $g \in \mathbf{C}_1$ where $\mathrm{cod}(f) = \mathrm{dom}(g)$, such that $g \circ f \in \mathbf{C}_1$ and $\mathrm{dom}(g \circ f) = \mathrm{dom}(f)$ and $\mathrm{cod}(g \circ f) = \mathrm{cod}(g)$.
- For any $A \in \mathbf{C}_0$ there is an *identity morphism* $1_A \in \mathbf{C}_1$ with $\mathrm{dom}(1_A) = A$ and $\mathrm{cod}(1_A) = A$.

Furthermore, the following laws hold:

- For all f, g, h in $\mathbf{C}_1$ where $\mathrm{cod}(f) = \mathrm{dom}(g)$ and $\mathrm{cod}(g) = \mathrm{dom}(h)$,
$$h \circ (g \circ f) = (h \circ g) \circ f \qquad \text{(associativity law)}.$$
- For any $f \in \mathbf{C}_1$,
$$f \circ 1_{\mathrm{dom}(f)} = f = 1_{\mathrm{cod}(f)} \circ f \qquad \text{(unit laws)}.$$

△

Notation 1.2. Given a category $\mathbf{C}$, it is convenient to write $f : A \to B$ to mean a morphism $f \in \mathbf{C}_1$ with $\mathrm{dom}(f) = A$ and $\mathrm{cod}(f) = B$. When there is no danger of ambiguity we will write $A \in \mathbf{C}$ instead of $A \in \mathbf{C}_0$, and similarly for morphisms. △

Example 1.3.

(i) A basic category is the category $\mathbf{1}$ consisting of a single object $\star \in \mathbf{1}$ and a single morphism $1_\star \in \mathbf{1}$.

(ii) There is a category $\mathbf{0}$ with no objects and no morphisms.

(iii) An example of a bigger category is the category **Set** of all sets. In this category the objects $\mathbf{Set}_0$ are sets and the morphisms $\mathbf{Set}_1$ are functions. Morphism composition is defined to be function composition and the identity morphisms are the identity functions. ◊

Definition 1.4. An *isomorphism* in a category $\mathbf{C}$ is a morphism $f : A \to B$ in $\mathbf{C}$ for which there exists an *inverse* morphism $g : B \to A$ in $\mathbf{C}$, such that

$$g \circ f = 1_A \qquad \text{and} \qquad f \circ g = 1_B.$$

If there exists such an inverse morphism we say that A and B are *isomorphic*. △

Definition 1.5. A *functor* is a map $F : \mathbf{C} \to \mathbf{D}$ between categories $\mathbf{C}$ and $\mathbf{D}$, where every object $A \in \mathbf{C}$ is associated to an object $F(C) \in \mathbf{D}$ and every morphism $f : B \to C$ in $\mathbf{C}$ is associated to a morphism $F(f) : F(B) \to F(C)$ in $\mathbf{D}$. A functor $F : \mathbf{C} \to \mathbf{D}$ must preserve identity and composition in the sense that

$$F(1_A) = 1_{F(A)} \qquad \text{and} \qquad F(g \circ f) = F(g) \circ F(f).$$

△

Definition 1.6. A *natural transformation* $\alpha : F \to G$ between functors $F, G : \mathbf{C} \to \mathbf{D}$ consists of morphisms $\alpha_A : F(A) \to G(A)$ for each object $A \in \mathbf{C}$, such that for any morphism $f : A \to B$ in $\mathbf{C}$ the following square commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{\alpha_A} & G(A) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(B) & \xrightarrow{\alpha_B} & F(B) \end{array}$$

This means that α is required to satisfy $\alpha_B \circ F(f) = G(f) \circ \alpha_A$.

A *natural isomorphism* is a natural transformation α where each morphism α_A is an isomorphism. △

Definition 1.7. A category $\mathbf{C}$ gives rise to a *dual category* $\mathbf{C}^{\mathrm{op}}$. For each object $A \in \mathbf{C}$ there is a corresponding dual object $A^{\mathrm{op}} \in \mathbf{C}^{\mathrm{op}}$ and for each morphism $f : A \to B$ in $\mathbf{C}$ there is a corresponding dual morphism $f^{\mathrm{op}} : B^{\mathrm{op}} \to A^{\mathrm{op}}$. Notice that $(\mathbf{C}^{\mathrm{op}})^{\mathrm{op}} = \mathbf{C}$. △

1.2 Universal properties

Definition 1.8.

(i) An object 0 in a category $\mathbf{C}$ is *initial* iff for every object $A \in \mathbf{C}$ there is a unique morphism $0 \to A$.

(ii) An object 1 in a category $\mathbf{C}$ is *terminal* iff for every object $A \in \mathbf{C}$ there is a unique morphism $A \to 1$. △

Example 1.9. In **Set** the empty set is initial and any singleton set is terminal. ◊

Definition 1.10.

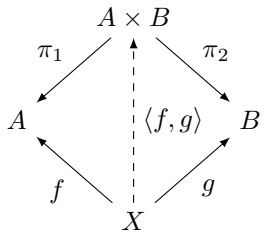
(i) A *diagram* of shape $\mathbf{J}$ is a functor $F : \mathbf{J} \to \mathbf{C}$.

(ii) A *cone* over a diagram $F : \mathbf{J} \to \mathbf{C}$ is a natural transformation $\alpha : A \to F$ with *summit* A an object in $\mathbf{C}$. For i object in $\mathbf{J}$, we refer to the morphisms $\alpha_i : A \to F(i)$ as the *legs* of the cone. △

Definition 1.11. Suppose $F : \mathbf{J} \to \mathbf{C}$ is a diagram. There is a category $\mathrm{Cone}(F)$ where the objects are cones over F. A morphism in $\mathrm{Cone}(F)$ from a cone $\alpha : A \to F$ to $\beta : B \to F$ corresponds to a morphism $\vartheta : A \to B$ in $\mathbf{C}$ satisfying $\alpha_i = \beta_i \circ \vartheta : A \to F(i)$ for all objects $i \in \mathbf{J}$. The identity morphism in $\mathrm{Cone}(F)$ of a cone $\alpha : A \to F$ is the identity morphism $1_A \in \mathbf{C}$, and composition in $\mathrm{Cone}(F)$ is composition in $\mathbf{C}$. △

Definition 1.12. A *limit* of a diagram $F : \mathbf{J} \to \mathbf{C}$ is a terminal object in the category $\mathrm{Cone}(F)$. △

Example 1.13. Consider a category $\mathbf{K}$ consisting of two objects 1 and 2, and the two required identity morphisms. Let $F : \mathbf{K} \to \mathbf{C}$ be a diagram and write $A = F(1)$ and $B = F(2)$. A limit of the diagram F is referred to as a *binary product*, we write $A \times B$ for the summit of the limit cone and $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ for the legs. If $f : X \to A$ and $g : X \to B$ are morphisms in $\mathbf{C}$ then there is a cone $\alpha : X \to F$ with $\alpha_1 = f$ and $\alpha_2 = g$. Hence, there is a unique map $\langle f, g \rangle : X \to A \times B$ which satisfies $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$, as indicated in the following diagram.



In the category **Set**, a binary product corresponds to the usual cartesian product where the projections $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$ are the legs of the limit cone. ◊

Example 1.14. A limit of a diagram $F : \mathbf{0} \to \mathbf{C}$ is a terminal object in $\mathbf{C}$. Any singleton set in **Set** is a terminal object. ◊

Definition 1.15. Given a functor $F : \mathbf{C} \to \mathbf{D}$ there is a *dual functor* $F^{\mathrm{op}} : \mathbf{C}^{\mathrm{op}} \to \mathbf{D}^{\mathrm{op}}$ defined on objects and morphisms by

$$F^{\mathrm{op}}(X^{\mathrm{op}}) = F(X)^{\mathrm{op}} \qquad \text{and} \qquad F^{\mathrm{op}}(f^{\mathrm{op}}) = F(f)^{\mathrm{op}}.$$

△

Definition 1.16. Let $F : \mathbf{J} \to \mathbf{C}$ be a diagram. The category of cones $\mathrm{Cone}(F^{\mathrm{op}})$ has a dual category of *cocones* $\mathrm{Cocone}(F) = \mathrm{Cone}(F^{\mathrm{op}})^{\mathrm{op}}$. △

Remark 1.17. Cocones $\alpha \in \mathrm{Cocone}(F)$ are natural transformations $\alpha : F \to A$ where $A \in \mathbf{C}$ is an object called the *nadir*. A morphism from cocone $\alpha : F \to A$ to $\beta : F \to B$ corresponds to a morphism $\vartheta : B \to A$ in $\mathbf{C}$ such that $\alpha_i = \vartheta \circ \beta_i : F(i) \to A$ for all objects $i \in \mathbf{J}$. ◊

Definition 1.18. A *colimit* is an initial object in the category of cocones $\mathrm{Cocone}(F)$. △

Remark 1.19. Since a limit in $\mathbf{C}^{\mathrm{op}}$ is a terminal object in the category $\mathrm{Cone}(F^{\mathrm{op}})$ it corresponds to an initial object in the dual category $\mathrm{Cocone}(F)$. Hence a limit in $\mathbf{C}^{\mathrm{op}}$ corresponds to a colimit in $\mathbf{C}$. ◊

Example 1.20. A colimit of $F : \mathbf{0} \to \mathbf{C}$ is an initial object in $\mathbf{C}$. For instance, in the category **Set** the empty set is initial. ◊

2 Universal algebra

This section presents set theoretic multi sorted universal algebra. Readers familiar with multi sorted universal algebra may want to just skim this section. The section is based on the Math Classes library [2] and the universal algebra book by B. Stanley and H. P. Sankap-panavar [5].

2.1 Definitions

Definition 2.1. A *signature* σ consists of:

- A set of *sorts* $\mathcal{S}_\sigma$.
- A set of *function symbols* $\mathcal{F}_\sigma$.
- For each function symbol $\alpha \in \mathcal{F}_\sigma$, a function symbol type, which is a finite sequence $\mathcal{T}_\alpha = (s_n)_{n \leq \text{ari}(\alpha)}$ of sorts $s_n \in \mathcal{S}_\sigma$, where $n \in \mathbb{N}_0$ and $\text{ari}(\alpha) \in \mathbb{N}_0$.

The number $\text{ari}(\alpha)$ is called the *arity* of the function symbol α. △

Definition 2.2. An *algebra* $\mathbf{A}$ for a signature σ consists of:

- A family of *carriers* $(\mathbf{A}_s)_{s \in \mathcal{S}_\sigma}$ indexed by $s \in \mathcal{S}_\sigma$.
- A family of *operations* $(\alpha^{\mathbf{A}})_{\alpha \in \mathcal{F}_\sigma}$ indexed by $\alpha \in \mathcal{F}_\sigma$. An operation for $\alpha \in \mathcal{F}_\sigma$ is an n-ary function (a constant when $n = 0$)

  $$\alpha^{\mathbf{A}} : (A_{s_1} \times A_{s_2} \times \cdots \times A_{s_n}) \to A_t$$

  where $n = \text{ari}(\alpha)$ is the arity of the function symbol $\alpha \in \mathcal{F}_\sigma$ and $(s_1, s_2, \ldots, s_n, t) = \mathcal{T}_\alpha$ is the function symbol type of α.

△

Example 2.3. Any group G is an algebra for a signature with just one sort. A group G has a binary operation $\cdot : G \times G \to G$, a unary operation $(-)^{-1} : G \to G$ and a constant $1 \in G$. ◊

Definition 2.4. Given algebras $\mathbf{A}$ and $\mathbf{B}$ for some signature σ. An algebra *homomorphism* $f : \mathbf{A} \to \mathbf{B}$ is a family of functions

$$(f_s : \mathbf{A}_s \to \mathbf{B}_s)_{s \in \mathcal{S}_\sigma}, \text{ indexed by } s \in \mathcal{S}_\sigma,$$

satisfying

$$f_t(\alpha^{\mathbf{A}}(a_1, \ldots, a_n)) = \alpha^{\mathbf{B}}(f_{s_1}(a_1), \ldots, f_{s_n}(a_n))$$

for all function symbols $\alpha \in \mathcal{S}_\sigma$, where $(s_1, \ldots, s_n, t) = \mathcal{T}_\alpha$ is the function symbol type. △

Definition 2.5. An algebra *isomorphism* is a homomorphism $(f_s)_s$ where f_s is bijective for all sorts $s \in \mathcal{S}_\sigma$. If there exists an isomorphism $\mathbf{A} \to \mathbf{B}$ then we say $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic*. △

Example 2.6. Group homomorphisms/isomorphisms are algebra homomorphisms/isomorphisms. ◊

Lemma 2.7. Let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be algebras a signature σ and suppose there exist homomorphisms $f = (f_s : \mathbf{A}_s \to \mathbf{B}_s)_s$ and $g = (g_s : \mathbf{B}_s \to \mathbf{C}_s)_s$. The family of composed functions

$$g \circ f := (g_s \circ f_s : \mathbf{A}_s \to \mathbf{C}_s)_{s \in \mathcal{S}_\sigma}$$

is a homomorphism $\mathbf{A} \to \mathbf{C}$. □

Definition 2.8. Let $\mathbf{A}$ and $\mathbf{B}$ be algebras for a signature σ. Then $\mathbf{B}$ is a *subalgebra* of $\mathbf{A}$ iff

- $\mathbf{B}_s \subseteq \mathbf{A}_s$ for all sorts $s \in \mathcal{S}_\sigma$,
- $\alpha^{\mathbf{B}} : (\mathbf{B}_{s_1} \times \cdots \times \mathbf{B}_{s_n}) \to B_t$ is the restriction of $\alpha^{\mathbf{A}} : (\mathbf{A}_{s_1} \times \cdots \times \mathbf{A}_{s_n}) \to A_t$ for all function symbols $\alpha \in \mathcal{F}_\sigma$ and $(s_1, \ldots, s_n, t) = \mathcal{T}_\alpha$. △

2.2 Isomorphism theorems

Normal subgroups play a central role in defining quotient groups and in the isomorphism theorems, which are fundamental to the development of group theory. Ideals play an analogous role in defining quotient rings and in the corresponding isomorphism theorems in ring theory. Given this parallel situation, it seems that there should be a general formulation of normal subgroup and ideal. In this subsection we will see that congruence is such a formulation, giving rise to generic versions of the isomorphism theorems.

Definition 2.9. Let $\mathbf{A}$ be an algebra for a signature σ. A family of equivalence relations $\sim_s \subseteq \mathbf{A}_s \times \mathbf{A}_s$, indexed by $s \in \mathcal{S}_\sigma$, is a *congruence* on $\mathbf{A}$ iff

$$\alpha^{\mathbf{A}}(a_1, \ldots, a_n) \sim_t \alpha^{\mathbf{A}}(b_1, \ldots, b_n), \text{ whenever } a_1 \sim_{s_1} b_1, \ldots, a_n \sim_{s_n} b_n,$$

for $\alpha \in \mathcal{F}_\sigma$ and $(s_1, \ldots, s_n, t) = \mathcal{T}_\alpha$ the function symbol type. △

Definition 2.10. Suppose $\sim = (\sim_s)_{s \in \mathcal{S}_\sigma}$ is a congruence on an algebra $\mathbf{A}$ for some signature σ. The *quotient algebra* $\mathbf{A}/\!\sim$ is the algebra for σ with

- carriers $\mathbf{A}_s/\!\sim_s$, for each $s \in \mathcal{S}_\sigma$, the quotient set of $\mathbf{A}_s$ by $\sim_s$;
- operations $\alpha^{(\mathbf{A}/\sim)}([a_1], \ldots, [a_n]) = [\alpha^{\mathbf{A}}(a_1, \ldots, a_n)]$, for $\alpha \in \mathcal{F}_\sigma$ and equivalence classes $[a_i] \in \mathbf{A}_{s_i}/\!\sim_{s_i}$.

This algebra is well defined. △

Example 2.11. If $\sim$ is a congruence on a group G with unit 1 then the equivalence class $N := [1] \in G/\!\sim$ is a normal subgroup of G, where the quotient group G/N and the quotient algebra $G/\!\sim$ coincide.

Conversely, if N is a normal subgroup of G then the relation $\sim$ given by

$$x \sim y \qquad \text{iff} \qquad xy^{-1} \in N$$

is a congruence on G where $N = [1] \in G/\!\sim$ and $G/N = G/\!\sim$. ◊

Definition 2.12. Let $\mathbf{A}, \mathbf{B}$ be algebras for a signature σ and suppose $f = (f_s : \mathbf{A}_s \to \mathbf{B}_s)_s$ is a homomorphism $\mathbf{A} \to \mathbf{B}$. The *kernel* $\ker(f)$ of f is a family of sets $\ker_t(f) \subseteq \mathbf{A}_t \times \mathbf{A}_t$, indexed by $t \in \mathcal{S}_\sigma$, defined by

$$\ker_t(f) = \{(a, b) \in \mathbf{A}_t \times \mathbf{A}_t \mid f_t(a) = f_t(b)\}.$$

△

Remark 2.13. A function $f_t : \mathbf{A}_t \to \mathbf{B}_t$ of an algebra homomorphism $f : \mathbf{A} \to \mathbf{B}$ is injective if and only if $\ker_t(f)$ is the identity relation. ◊

Theorem 2.14 (First isomorphism theorem)**.** Suppose $\mathbf{A}$ and $\mathbf{B}$ are algebras for a signature σ. Let $f : \mathbf{A} \to \mathbf{B}$ be a homomorphism.

(i) The homomorphic image $f(\mathbf{A}) := (f_s(\mathbf{A}_s))_{s \in \mathcal{S}_\sigma}$ induces a subalgebra of $\mathbf{B}$.
(ii) The kernel $\ker(f)$ is a congruence on $\mathbf{A}$.
(iii) The quotient algebra $\mathbf{A}/\ker(f)$ and the image algebra $f(\mathbf{A})$ are isomorphic. □

Theorem 2.15 (Second isomorphism theorem)**.** Let $\mathbf{A}$ and $\mathbf{B}$ be algebras with $\mathbf{B}$ a subalgebra of $\mathbf{A}$ and assume $\varphi = (\varphi_s)_{s\in\mathcal{S}_\sigma}$ is a congruence on $\mathbf{A}$. For $s \in \mathcal{S}_\sigma$, write

$$\varphi_s^{\mathbf{B}} = \varphi_s \cap (\mathbf{B} \times \mathbf{B}),$$
$$[\mathbf{B}]_s^{\varphi} = \{[a] \in \mathbf{A}_s/\varphi_s \mid [a] \cap \mathbf{B}_s \neq \emptyset\}.$$

(i) The family of relations $\varphi^{\mathbf{B}} := (\varphi_s^{\mathbf{B}})_{s\in\mathcal{S}_\sigma}$ is a congruence on $\mathbf{B}$.
(ii) The family of sets $[\mathbf{B}]^{\varphi} := ([\mathbf{B}]_s^{\varphi})_{s\in\mathcal{S}_\sigma}$ induces a subalgebra of $\mathbf{A}/\varphi$.
(iii) The algebras $\mathbf{B}/\varphi^{\mathbf{B}}$ and $[\mathbf{B}]^{\varphi}$ are isomorphic. □

Theorem 2.16 (Third isomorphism theorem)**.** Let φ, ϑ be congruences on some algebra $\mathbf{A}$ where $\vartheta_s \subseteq \varphi_s$ for all $s \in \mathcal{S}_\sigma$. Set

$$\varphi_s/\vartheta_s = \{([a],[b]) \in (\mathbf{A}_s/\vartheta_s) \times (\mathbf{A}_s/\vartheta_s) \mid \varphi_s(a,b)\}, \qquad \text{for } s \in \mathcal{S}_\sigma.$$

(i) The family of relations $\varphi/\vartheta := (\varphi_s/\vartheta_s)_{s\in\mathcal{S}_\sigma}$ is a congruence on $\mathbf{A}/\vartheta$.
(ii) The algebras $(\mathbf{A}/\vartheta)/(\varphi/\vartheta)$ and $\mathbf{A}/\varphi$ are isomorphic. □

3 Homotopy Type Theory

This section is based on the HoTT book [4]. Readers already familiar with HoTT may want to skip to section 3.3 and skim it.

HoTT is an alternative to ZFC set theory as a foundation of mathematics. HoTT is in particular distinguished from ZFC by being a type theory rather than a first order theory. Proofs are the same basic notion as other types like numbers.

HoTT allows for a convenient synthetic approach to homotopy theory: types are spaces, type inhabitants are points, identity types are paths. Promising research on cubical type theory is indicating that there is a computational interpretation of HoTT [8]. Another advantage of HoTT is that it formalises the natural mathematical practice of identifying isomorphic objects. Theorem 5.6 below formalises this by showing that isomorphic algebras are literally equal in HoTT.

Section 3.1 introduces the basic type theory the HoTT book is based on. Section 3.2 presents some of the elementary notions from homotopy type theory. Section 3.3 introduces a couple of higher inductive types.

3.1 Type Theory

A *universe* is a type of types. All universes $\mathcal{U}_n$ come with an associated level $n \in \mathbb{N}$. There is a cumulative hierarchy of universes

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$$

So universe $\mathcal{U}_n$ has type $\mathcal{U}_{n+k}$ for any $k \geq 1$. To simplify notation we leave the universe level implicit and write $\mathcal{U}$.

We use $\equiv$ for judgmental equality and $=$ for the identity type. The induction principle

for the identity type is

$$\mathrm{ind}_{=_A} : \prod_{\left(C:\prod_{(x,y:A)}(x=y)\to\mathcal{U}_i\right)} \Big(\prod_{(x:A)} C(x,x,\mathrm{refl}_x)\Big) \to \prod_{(x,y:a)} \prod_{(p:x=y)} C(x,y,p)$$

$$\mathrm{ind}_{=_A}(C,c,x,x,\mathrm{refl}_x) \equiv c(x),$$

where we write $f(a,b)$ for $f(a)(b)$ when the intention is clear.

Definition 3.1. Suppose $A : \mathcal{U}$ is a type and $x, y : A$ inhabitants. The identity type $x = y$ is called a *path* from x to y and the induction principle for the identity type is referred to as *path induction*. A term $p : x = y$ is viewed on as a path with *endpoints* x and y in a space A. △

Remark 3.2. The interpretation of identity types as paths is made precise in the simplicial model of univalent foundations [9]. ◊

Lemma 3.3. The path type is an equivalence relation. For let $x, y, z : A$ be inhabitants of a type $A : \mathcal{U}$, then

- $\mathrm{refl}_x : x = x$,
- $p : x = y$ implies $p^{-1} : y = x$,
- $p : x = y$ and $q : y = z$ implies $p \cdot q : x = z$. □

Definition 3.4. Let $p : x = y$ and $q : y = z$ be paths in some type A. We refer to $p^{-1} : y = x$ as the *inverse* path of p and $p \cdot q : x = z$ as the *composite* of p and q. △

3.2 Univalent foundations

The first definition in this section is fundamental and due to Voevodsky [10].

Definition 3.5. A type A is *contractible* if there exists a point $a : A$ and a dependent function $f : \prod_{(x:A)}(a = x)$ mapping $x : A$ to a path $a = x$,

$$\mathrm{isContr} : \mathcal{U} \to \mathcal{U}$$
$$\mathrm{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x).$$

△

Remark 3.6. It is tempting to use a propositions-as-types interpretation and read the type $\mathrm{isContr}(A)$ as: there exists a basepoint $a : A$ such that for all $x : A$ there is a path $a = x$ from a to x. This makes it sound like A is just path-connected. It actually says something stronger. For an intuition, let A be a set theoretic topological space and $I = [0,1]$ the unit interval. Suppose there exists a point $a \in A$ and a homotopy $f : A \times I \to X$ such that for all $x \in A$, $f(x,-) : I \to A$ is a path from a to x. Then f is a homotopy $a \simeq \mathrm{id}_A$ showing that the identity function on A is nullhomotopic. This says exactly that A is a contractible space. ◊

Example 3.7. The unit type $\mathbf{1}$ is a contractible type. Indeed

$$\mathrm{unitIsContr} : \mathrm{isContr}(\mathbf{1})$$
$$\mathrm{unitIsContr} :\equiv (\star,\ \mathrm{ind}_{\mathbf{1}}(\lambda x.\ \star = x,\ \mathrm{refl}_\star)),$$

where $\mathrm{ind}_{\mathbf{1}} : \prod_{(C:\star\to\mathcal{U})} C(\star) \to \prod_{(x:\mathbf{1})} C(x)$ is the induction principle for $\mathbf{1}$. ◊

Definition 3.8. A *mere proposition* is a type A for which $x = y : \mathcal{U}$ is contractible for all $x, y : A$,

$$\begin{aligned} &\mathrm{isProp} : \mathcal{U} \to \mathcal{U} \\ &\mathrm{isProp}(A) :\equiv \prod_{(x,y:A)} \mathrm{isContr}(x = y). \end{aligned}$$

△

Example 3.9.

(i) Any contractible type is a mere proposition,

$$\begin{aligned} &\mathrm{contrIsProp} : \prod_{(A:\mathcal{U})} \mathrm{isContr}(A) \to \mathrm{isProp}(A) \\ &\mathrm{contrIsProp}(A, (a, P))(x, y) :\equiv (P(x))^{-1} \cdot P(y), \qquad \text{since } P : \prod_{(z:A)} (a = z). \end{aligned}$$

(ii) The empty type $\mathbf{0}$ is a mere proposition, but it is not contractible.

(iii) Suppose $A : \mathcal{U}$ is a type and $B : A \to \mathcal{U}$ a type family such that $B(x)$ is a mere proposition for all $x : A$, then the dependent function type $\prod_{(x:A)} B(x)$ is a mere proposition as well. This is not the case for the Σ-type or coproduct. Section 3.3 below demonstrates how higher inductive types can be used to define a propositionally truncated Σ-type $\|\sum_{(x:A)} B(x)\|$, which is a mere proposition for any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$.

◊

Definition 3.10. A *set* is a type that satisfies the uniqueness of identity proofs property, if $p : x = y$ and $q : x = y$ then $p = q$,

$$\begin{aligned} &\mathrm{isSet} : \mathcal{U} \to \mathcal{U} \\ &\mathrm{isSet}(A) :\equiv \prod_{(x,y:A)} \mathrm{isProp}(x = y). \end{aligned}$$

△

Example 3.11.

(i) If $A : \mathcal{U}$ is a type and $B : A \to \mathcal{U}$ a type family where $B(x)$ is a set for all $x : A$, then the dependent function type $\prod_{(x:A)} B(x)$ is a set.

(ii) Let $A : \mathcal{U}$ be a type and $B : A \to \mathcal{U}$ a type family. If A is a set and $B(x)$ is a set for all $x : A$, then the Σ-type $\sum_{(x:A)} B(x)$ is a set. A similar statement holds for coproducts.

◊

Definition 3.12. Let $f : A \to B$ be a function and $x, y : A$ inhabitants. Define

$$\mathrm{ap}_f : x = y \to f(x) = f(y)$$
$$\mathrm{ap}_f(p) :\equiv \mathrm{ind}_{=_A}(C, c, x, y, p)$$

where

$$C : \prod_{(x,y:A)} \big(x = y \to \mathcal{U}\big), \qquad C(x, y, q) :\equiv f(x) = f(y)$$
$$c : \prod_{(x:A)} \big(f(x) = f(x)\big), \qquad c(x) :\equiv \mathrm{refl}_{f(x)}$$

△

Definition 3.13. Given a type family $P : A \to \mathcal{U}$ and a path $p : x = y$, where $x, y : A$. Then there is a function

$$\mathrm{transport}(P, p, -) : P(x) \to P(y)$$
$$\mathrm{transport}(P, p, -) :\equiv \mathrm{ind}_{=_A}(C, c, x, y, p).$$

where

$$C : \prod_{(x,y:A)} \big(x = y \to \mathcal{U}\big), \qquad C(x, y, q) :\equiv P(x) \to P(y)$$
$$c : \prod_{(x:A)} \big(P(x) \to P(x)\big), \qquad c(x)(h) :\equiv h$$

△

Definition 3.14. A function $f : A \to B$ is an *equivalence* if there exist functions $g, h : B \to A$ such that $f(g(x)) = x$ for all $x : B$ and $h(f(x)) = x$ for all $x : A$,

$$\mathrm{isequiv}(f) :\equiv \Big(\sum_{(g:B\to A)} \prod_{(x:B)} \big(f(g(x)) = x\big)\Big) \times \Big(\sum_{(h:B\to A)} \prod_{(x:A)} \big(h(f(x)) = x\big)\Big).$$

For $A, B : \mathcal{U}$ types, we define

$$(A \simeq B) :\equiv \sum_{(f:A\to B)} \mathrm{isequiv}(f).$$

When $A \simeq B$ then we say A and B are *equivalent*. △

Remark 3.15. Given types $A, B : \mathcal{U}$ there is a function $\mathrm{idtoequiv} : A = B \to A \simeq B$. The *univalence axiom* states that this function is an equivalence. ◊

Axiom 3.16 (Univalence axiom)**.** The function $\mathrm{idtoequiv} : A = B \to A \simeq B$ is an equivalence,

$$\mathrm{isequiv}(\mathrm{idtoequiv}).$$

□

Remark 3.17. So equality is equivalent to equivalence,

$$(A = B) \simeq (A \simeq B).$$

◊

The univalence axiom implies function extensionality:

Theorem 3.18. Assuming univalence there is an equivalence

$$(f = g) \simeq \Big(\prod_{(x:A)} \big(f(x) = g(x)\big)\Big), \qquad \text{for all } f, g : A \to B.$$

□

3.3 Higher inductive types

Higher inductive types are inductive types generated by constructors of inhabitants of the type, paths in the type and higher paths. This section introduces two higher inductive types that we will use in part II. Chapter 6 in the HoTT book [4] contains more information on higher inductive types.

Definition 3.19 (Propositional truncation)**.** Let A be any type. The propositional truncation $\|A\|$ of A is the higher inductive type with generating constructors:

(i) a function $|-| : A \to \|A\|$,
(ii) for all $x, y : \|A\|$, there is a path $\rho_{x,y} : x = y$.

There is an associated recursion principle. Given a type B and

- a function $g : A \to B$,
- for all $x, y : B$ there is a path $p_{x,y} : x = y$.

Then there is a function $f : \|A\| \to B$ such that $f(|a|) \equiv g(a)$ for all $a : A$. △

The propositional truncation type has an induction principle as well, but the recursion principle for propositional truncation implies the induction principle.

Example 3.20. The constructor (ii) of the propositional truncation type says that $\|A\|$ is a proposition. Using propositional truncation we have a mere proposition $\|\sum_{(x:A)} P(x)\|$ for any $P : A \to \mathcal{U}$. If there is a term $t : \|\sum_{(x:A)} P(x)\|$ and $B : \mathcal{U}$ is a mere proposition, by the recursion principle, we may assume an inhabitant $a : \sum_{(x:A)} P(x)$ to prove B. ◊

Definition 3.21. Let $f : A \to B$ be a function. We say that f is *surjective* iff

$$\prod_{(b:B)} \| \sum_{(a:A)} \big(f(a) = b\big)\|.$$

△

Remark 3.22. The above definition of surjective is a mere proposition. This would not generally be the case if we omitted the propositional truncation in the definition. ◊

Definition 3.23 (Set-quotient)**.** Let A be a type and $R : A \to A \to \mathcal{U}$ a family of mere propositions, such that $R(x, y)$ is a mere proposition for all $x, y : A$. The *set-quotient* A/R is the higher inductive type generated by the constructors:

(i) a function $q : A \to A/R$;
(ii) for $a, b : A$ such that $R(a, b)$, there is a path $q(a) = q(b)$;
(iii) if $x, y : A/R$ and $r, s : x = y$ then $r = s$. △

The set-quotient has a recursion principle and an induction principle, but we will not need the details. The constructor (i) of the set-quotient gives a quotient map $q : A \to A/R$. The constructor (ii) says that elements $a, b : A$ for which $R(a, b)$ holds are identified in A/R. Constructor (iii) implies that A/R is a set.

Part II
Universal algebra in HoTT

This part of the report presents the universal algebra development for the Coq HoTT library. The formalisation can be found at `https://github.com/andreaslyn/hott-classes`. The formalisation contains proofs of all the lemmas and theorems presented below. The start of the formalisation is part of a Coq project which was supervised by B. Spitters. That project is attached as Appendix A. It contains a complete proof of Theorem 5.6 below.

In section 6, Category Theory is used as a tool to verify our definitions. For example, we want a binary product of two algebras to be a binary product in the category with algebras and algebra homomorphisms.

From hereon we switch to a pseudo code notation close to the Coq UTF-8 syntax. This makes it easier to relate the report to the formalisation. The notation `x` ≡ `y` will denote `x` is judgmentally equal to `y` and `x = y` is the path type.

4 Algebra

This section gives the main definitions in the universal algebra development. They are explained in more detail in appendix A. The definitions are similar to those in Section 2, but they are homotopy type theoretic in this section. Before defining signature and algebra we will introduce a non-empty list datatype.

Definition 4.1 (`ne_list`)**.** *Non-empty list* is defined by

```
Inductive ne_list (T : Type) : Type :=
  | one : T → ne_list T
  | cons : T → ne_list T → ne_list T.
Arguments one {T}.
Arguments cons {T}.
```
△

Notation 4.2. The `Arguments one {T}` statement above means that the `T:Type` argument to `one` should be left implicit. Hence `one : (∏ {T:Type}, T → ne_list T)` where curly braces in the type indicate implicit arguments. △

Definition 4.3. We will use the notation:

```
Global Notation "[: x :]" := (one x) : ne_list_scope.
Global Notation "[: x ; .. ; y ; z :]"
    := (cons x .. (cons y (one z)) ..) : ne_list_scope.
Global Infix ":::"
    := cons (at level 60, right associativity) : ne_list_scope.
```
△

The non-empty list is used to define the function symbol type of function symbols.

Definition 4.4 (`Signature`)**.** A *signature* is defined by

```
Record Signature : Type := BuildSignature
  { Sort : Type
  ; Symbol : Type
  ; symbol_types : Symbol → ne_list Sort }.
```

```
Definition SymbolType (σ : Signature) := ne_list (Sort σ).
Global Coercion symbol_types : Signature >-> Funclass.
```
△

Notation 4.5. The above `Global Coercion` allows for using a signature `σ : Signature` as a function `σ u ≡ symbol_types σ u`, for all function symbols `u : Symbol σ`. △

The next definition is used to convert `σ u ≡ symbol_types σ u` into the type of the algebra operation corresponding to `u`.

Definition 4.6. The `Operation` function has type

```
Operation : ∏ {σ : Signature}, (Sort σ → Type) → SymbolType σ → Type.
```

For `A : Sort σ → Type` and `w : SymbolType σ` a symbol type, it is defined by

```
Operation A w := A s₁ → A s₂ → ⋯ → A sₙ → A t
```

where `w ≡ [:s₁; s₂; …; sₙ; t:]` and `s₁ s₂ ⋯ sₙ t : Sort σ` are all sorts. △

Definition 4.7 (`Algebra`)**.** An *algebra* is defined by

```
Record Algebra {σ : Signature} : Type := BuildAlgebra
  { carriers : Sort σ → Type
  ; operations : ∏ (u : Symbol σ), Operation carriers (σ u)
  ; hset_carriers_algebra : ∏ (s : Sort σ), IsHSet (carriers s) }.
Arguments Algebra : clear implicits.
```

We also introduce an implicit coercion and notation:

```
Global Coercion carriers : Algebra >-> Funclass.
Global Notation "u ^^ A" := (operations A u) (at level 60, no associativity).
```
△

Notation 4.8. The `Arguments Algebra : clear implicits` notation means that the `σ : Signature` argument to `Algebra` should not be implicit. Otherwise it would be implicit because it was given inside curly braces. The `σ : Signature` argument is still implicit for `carriers`, etc. △

An algebra `A : Algebra σ` for a signature `σ` consists of a type `A s ≡ carriers A s` for each sort `s : Sort σ`, and an *operation* `u^^A ≡ operations A u : Operation A (σ u)` for each function symbol `u : Symbol σ`. We require `carriers A s` be a set for any `s : Sort σ` to obtain the uniqueness of identity proofs property for algebras.

5 Homomorphism and isomorphism

In this section we let `A B: Algebra σ` denote two algebras for a signature `σ:Signature`.

Definition 5.1 (`Homomorphism`)**.** Let `f : (∏ (s : Sort σ), A s → B s)` be a family of functions. Suppose `α : Operation A w` and `β : Operation B w` are operations of types given by `w`, see Definition 4.6. We define `OpPreserving f α β : Type` to be the type:

For all `x₁ : A s₁, x₂ : A s₂, …, xₙ : A sₙ`,

`f t (α x₁ x₂ ⋯ xₙ) = β (f s₁ x₁) (f s₂ x₂) ⋯ (f sₙ xₙ)`

where `[:s₁; s₂; …; sₙ; t:] ≡ σ u` is the symbol type of `u`.

A *homomorphism* is defined by

```
Record Homomorphism {σ} {A B : Algebra σ} : Type
  := BuildHomomorphism
      { def_hom : ∏ (s : Sort σ), A s → B s
      ; is_hom : ∏ (u : Symbol σ) OpPreserving def_hom (u^^A) (u^^B) }.
Arguments Homomorphism {σ}.
Arguments BuildHomomorphism {σ} {A B : Algebra σ} def_hom {is_hom}.
Global Coercion def_hom : Homomorphism >-> Funclass.
```
△

Definition 5.2 (`IsIsomorphism`)**.** For `f : Homomorphism A B` a homomorphism, `IsIsomorphism f : Type` is defined as the type:

For all `s : Sort σ`, `f s` is both surjective and injective.

By surjective we mean Definition 3.21 and by injective we mean

```
∏ (x y : A s), f s x = f s y → x = y.
```

We say that `f` is an *isomorphism* if `IsIsomorphism f` holds. △

Remark 5.3. This definition of isomorphism is similar to that in set theoretic universal algebra. Since surjective is a mere proposition one can show that `IsIsomorphism` is a mere proposition. This is proven in the formalisation. ◊

Lemma 5.4 (`equiv_carriers_isomorphism`)**.** Assume `f : Homomorphism A B` and `IsIsomorphism f`. The family of functions `f : (∏ s, A s → B s)` is a family of equivalences

```
f : ∏ s, A s ≃ B s
```
□

Lemma 5.5.

(i) There is an *identity* homomorphism `hom_id :Homomorphism A A` satisfying

```
hom_id (s : Sort σ) (x : A s) ≡ x
```

The identity homomorphism is an isomorphism `IsIsomorphism hom_id`.

(ii) Suppose `f : Homomorphism A B` and `IsIsomorphism f`. Equivalences have inverse functions, so by Lemma 5.4 there is a family of inverse functions

```
λ (s : Sort σ), (f s)^-1.
```

There is an *inverse* homomorphism `hom_inv : Homomorphism B A` satisfying

```
hom_inv (s : Sort σ) ≡ (f s)^-1.
```

This homomorphism is an isomorphism `IsIsomorphism hom_inv`.

(iii) With `g : Homomorphism B C` and `f : Homomorphism A B` there is a *composition* homomorphism `hom_compose : Homomorphism A C` satisfying

```
hom_compose (s : Sort σ) ≡ g s ∘ f s.
```

If both `g` and `f` are isomorphisms then `hom_compose` is an isomorphism. □

Isomorphisms have an important property in HoTT:

Theorem 5.6 (`path_isomorphism`)**.** Let `f : Homomorphism A B` be a homomorphism and suppose it is an isomorphism `IsIsomorphism f`. Then `A = B`.

Proof. Theorem 5.6 in appendix A. □

6 Algebras from algebras

6.1 Subalgebra

Definition 6.1 (SubalgebraPredicate)**.** Let `A : Algebra` σ be an algebra for a signature σ `: Signature` and suppose `P : (∏ (s : Sort` σ`), A s → Type)` such that `P s x` is a mere proposition for all `s` and `x`. Assume moreover that there is a term

$$\Theta \ \texttt{:} \prod \texttt{(x}_1 \texttt{ : A s}_1\texttt{) (x}_2 \texttt{ : A s}_2\texttt{)} \cdots \texttt{(x}_n \texttt{ : A s}_n\texttt{),}$$
$$\texttt{P s}_1 \texttt{ x}_1 \to \texttt{P s}_2 \texttt{ x}_2 \to \cdots \to \texttt{P s}_n \texttt{ x}_n \to \texttt{P t ((u^^A) x}_1 \texttt{ x}_2 \cdots \texttt{ x}_n\texttt{)}$$

for all function symbols `u : Symbol` σ, where `[:s`$_1$`; s`$_2$`; ...; s`$_n$`; t:] ≡` σ `u` is the symbol type of `u`. Then we will refer to `P` as a *subalgebra predicate* for `A`. △

Definition 6.2 (Subalgebra)**.** Let σ `: Signature` and `A : Algebra` σ and suppose `P : (∏ (s : Sort` σ`), A s → Type)` is a subalgebra predicate for `A`. Then there is a *subalgebra* `A&P : Algebra` σ of `A`. The `carriers` of the subalgebra `A&P` are

$$\texttt{(A\&P) (s : Sort } \sigma\texttt{)} \equiv \sum \texttt{x, P s x}$$

For each `u : Symbol` σ, the operation `u^^(A&P) : Operation (A&P) (`σ `u)` satisfies

$$\texttt{(u^^(A\&P)) (x}_1\texttt{; p}_1\texttt{) (x}_2\texttt{; p}_2\texttt{)} \cdots \texttt{(x}_n\texttt{; p}_n\texttt{)}$$
$$\equiv \texttt{((u^^A) x}_1 \texttt{ x}_2 \cdots \texttt{ x}_n \texttt{ ; } \Theta \texttt{ x}_1 \texttt{ x}_2 \cdots \texttt{ x}_n \texttt{ p}_1 \texttt{ p}_2 \cdots \texttt{ p}_n\texttt{)}$$

where `[:s`$_1$`; s`$_2$`; ...; s`$_n$`; t:]` $\equiv$ σ `u` is the symbol type of `u` and `(_ ; _)` is notation for the Σ-type constructor, so that `(x`$_i$`; p`$_i$`) : (A&P) s`$_i$. △

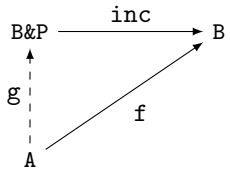
Remark 6.3. We can think of the subalgebra carriers `(A&P) s : (∑ x, P s x)` as a subset of `A s`, for each `s : Sort` σ. ◊

Lemma 6.4 (hom_inclusion_subalgebra)**.** Let σ `: Signature` and let `P : (∏ (s : Sort` σ`), A s → Type)` be a subalgebra predicate for an algebra `A : Algebra` σ. There is an inclusion homomorphism `inc : Homomorphism (A&P) A`,

```
inc (s : Sort σ) ((x; p) : (A&P) s) ≡ x.
```

The function `inc s : (A&P) s → A s` is an injection for all `s : Sort` σ. □

The following lemma shows that the subalgebra together with the above inclusion homomorphism behaves in the expected way. It says that for any subalgebra predicate `P : (∏ s, B s → Type)` and homomorphism `f : Homomorphism A B`, such that `P s (f s x)` holds for all `s : Sort` σ and `x : A s`, there exists a unique homomorphism `g : Homomorphism A (B&P)` making the following diagram commute:



Lemma 6.5. Suppose `A B : Algebra` σ are algebras for a signature σ and `P : (∏ s, B s → Type)` is a subalgebra predicate. There is an equivalence

```
Homomorphism A (B&P) ≃ (∑ (f : Homomorphism A B), ∏ s x, P s (f s x))
```

induced by postcomposition with the inclusion homomorphism

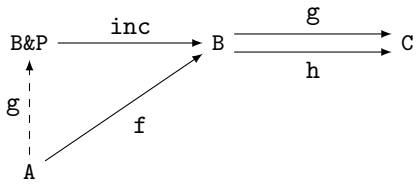
```
inc : Homomorphism (B&P) B
```

from lemma 6.4. □

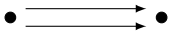
Remark 6.6. This can be expressed in category theoretic terms. For given a signature `σ : Signature` there is a category where the objects are algebras `Algebra` σ and the morphisms are homomorphisms. Suppose `g h : Homomorphism B C` are morphisms in this category. There is a subalgebra predicate `P : (∏ s, B s → Type)` satisfying

```
P s x ≡ (g s x = h s x).
```

Given a morphism `f : Homomorphism A B` where `hom_compose g f = hom_compose h f`, then `P s (f s x)` holds for all `s : Sort` σ and `x : A s`. So by the preceding lemma we have a commutative diagram:



The above subalgebra `B&P` is summit of a limit cone over a diagram of type



Such a limit cone is called an equaliser. This shows that the category of algebras for any signature `σ : Signature` has all equalisers. ◊

6.2 Product algebra

Definition 6.7 (`ProdAlgebra`)**.** Let `A : I → Algebra` σ be a family of algebras for some `σ : Signature` and `I : Type` an index type. The *product algebra* `Prod A : Algebra` σ has `carriers`

```
Prod A (s : Sort σ) ≡ ∏ (i:I), A i s.
```

For all `u : Symbol` σ, the operation `u^^(Prod A) : Operation (Prod A)(σ u)` satisfies

$$\texttt{(u\^\^(Prod A))}\ \texttt{(p}_1\ \texttt{:}\ \texttt{Prod A s}_1\texttt{)}\ \texttt{(p}_2\ \texttt{:}\ \texttt{Prod A s}_2\texttt{)}\ \cdots\ \texttt{(p}_n\ \texttt{:}\ \texttt{Prod A s}_n\texttt{)}$$
$$\equiv \lambda\ \texttt{(i:I), (u \^\^ A i) (p}_1\ \texttt{i) (p}_2\ \texttt{i)}\ \cdots\ \texttt{(p}_n\ \texttt{i)}$$

with `[:s₁; s₂; …; sₙ; t:] ≡` σ `u` the symbol type of `u`. △

Lemma 6.8 (`hom_projection_prod_algebra`)**.** Let `Prod A :Algebra` σ be the product algebra of a family of algebras `A : I → Algebra` σ. For each `i:I` there is a projection homomorphism π `i : Homomorphism (ProdAlgebra I A) (A i)`,

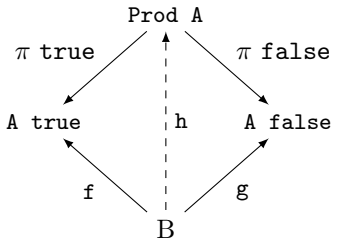
```
π (i:I) (s : Sort σ) (p : (Prod A) s) ≡ p i.
```

□

Remark 6.9. Suppose that `Prod A :Algebra` σ is the product algebra of `A : Bool` $\to$ `Algebra` σ. If there is a pair of homomorphisms `f : Homomorphism B (A true)` and `g : Homomorphism B (A false)`, where `B : Algebra` σ. Then there is a homomorphism `h : Homomorphism B (Prod A)` satisfying

```
h (s : Sort σ) (i:Bool) ≡ if i then f x else g x
```

This homomorphism `h` is the unique homomorphism making the following diagram commute.



From the above diagram we see that the category of algebras for any signature σ has all binary products `Prod A`, for any `A : Bool → Algebra` σ. More generally, let `I:Type` be a type assigned discrete category structure. Then a family of algebras `A : I → Algebra` σ is a diagram, and `Prod A` is the limit of the diagram. This limit is called a product, so the category of algebras for σ has all products. This is stated in the following lemma. It is an indication that our definition of product algebra is correct. ◊

Lemma 6.10 (`ump_prod_algebra`)**.** Let `A : I → Algebra` σ be a family of algebras and `I:Type` an indexing type. There is an equivalence

```
Homomorphism B (Prod A) ≃ (∏ i, Homomorphism B (A i))
```

induced by mapping `f : Homomorphism B (Prod A)` to the family of homomorphisms

```
λ (i:I), hom_compose (π i) f
```

where `π i : Homomorphism (Prod A) (A i)` is the `i`th projection homomorphism. □

6.3 Quotient algebra

Notation 6.11. For `R : relation X` a relation on some type `X`,

```
is_mere_relation X R ≡ ∏ (x y : X), IsHProp (R x y)
```

where `IsHProp` is isProp from Definition 3.8. △

Definition 6.12 (`Congruence`)**.** Let `A : Algebra` σ be an algebra for a signature σ `: Signature`. A family of relations `Φ : (∏ (s : Sort` σ`), relation (A s))` satisfies `HasCongruenceProperty A Φ` if for all function symbols `u : Symbol` σ,

$$\Phi\ s_1\ x_1\ y_1\ *\ \Phi\ s_2\ x_2\ y_2\ *\ \cdots\ *\ \Phi\ s_n\ x_n\ y_n$$

implies

$$\Phi\ t\ ((u\hat{}A)\ x_1\ x_2\ \cdots\ x_n)\ ((u\hat{}A)\ y_1\ y_2\ \cdots\ y_n)$$

where $[:s_1; s_2; \ldots; s_n; t:] \equiv \sigma\ u$ is the symbol type, and x_i `: A` s_i and y_i `: A` s_i.

A *congruence* is a family of mere equivalence relations satisfying `HasCongruenceProperty`,

```
Record Congruence {σ : Signature} (A : Algebra σ) : Type := BuildCongruence
  { relation_congruence
    : ∏ (s : Sort σ), relation (A s)
  ; is_mere_relation_congruence
    : ∏ (s : Sort σ), is_mere_relation (A s) (relation_congruence s)
  ; equivalence_congruence
    : ∏ (s : Sort σ), Equivalence (relation_congruence s)
  ; property_congruence
    : HasCongruenceProperty A relation_congruence }.
Global Coercion relation_congruence : Congruence >-> Funclass.
```
△

Definition 6.13. Suppose `R : relation X` is a relation on some type `X:Type`. In the Coq HoTT library `quotient R` is the name for the set-quotient from Definition 3.23. △

Definition 6.14 (`QuotientAlgebra`)**.** Let `σ : Signature` be a signature. Given an algebra `A : Algebra σ` and a congruence `Φ : Congruence A` the *quotient algebra* `A/Φ` has `carriers`

$$\texttt{(A/Φ) (s : Sort σ)} \equiv \texttt{quotient (Φ s)}$$

The `operations` of the quotient algebra `A/Φ` satisfy

$$\texttt{(u^^(A/Φ)) [x}_1\texttt{] [x}_2\texttt{]} \cdots \texttt{[x}_n\texttt{]} = \texttt{[(u^^A) x}_1\ \texttt{x}_2 \cdots \texttt{x}_n\texttt{]}$$

for all `u : Symbol σ` and `x`$_i$ `: A s`$_i$, where `[:s`$_1$`; s`$_2$`; ...; s`$_n$`; t:] ≡ σ u` is the symbol type of `u` and `[x`$_i$`] : quotient (Φ s`$_i$`)` is the equivalence class of `x`$_i$, Definition 3.23(i). △

Lemma 6.15 (`hom_quotient`)**.** For any algebra `A : Algebra σ` and congruence `Φ : Congruence A` there is a homomorphism `hom_quotient :Homomorphism A (A/Φ)` satisfying

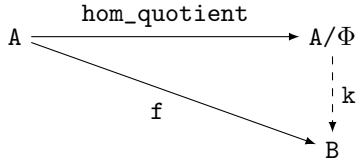
$$\texttt{hom\_quotient (s : Sort σ) (x : A s)} \equiv \texttt{[x]}$$

where `[x] : (A/Φ) s` is the equivalence class of `x`. This homomorphism is surjective. □

Remark 6.16. The quotient algebra `A/Φ` has the following universal property. Let `f : Homomorphism A B` be a homomorphism respecting the congruence `Φ` in the sense that `Φ s x y` implies `f s x = f s y`, for all `s : Sort σ` and `x y : A s`. There is a unique homomorphism `k : Homomorphism (A/Φ) B` such that

$$\texttt{hom\_compose k hom\_quotient} = \texttt{f}$$

as indicated in the following diagram:



◊

Lemma 6.17. Let Φ : Congruence A be a congruence on an algebra A : Algebra σ. Let B : Algebra σ. There is an equivalence

```
Homomorphism (A/Φ) B
  ≃ (∑ (f : Homomorphism A B), ∏ s x y, Φ s x y → f s x = f s y)
```

induced by precomposition with hom_quotient : Homomorphism A (A/Φ). □

This is the universal property one would expect from a quotient algebra, hence indicating the quotient algebra definition is correct.

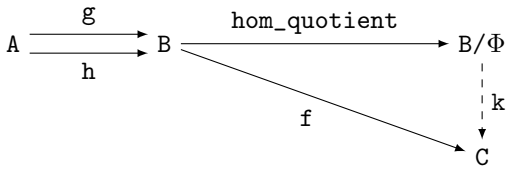
Remark 6.18. For the categorical point of view, suppose g h : Homomorphism A B are homomorphisms. There is a congruence Φ : Congruence B satisfying

```
Φ (s : Sort σ) (x : B s) (y : B s)
  ≡ ∏ (Ψ : Congruence B),
      (∏ (t : Sort σ) (a : A t), Ψ t (g t a) (h t a)) → Ψ s x y
```

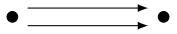
This is the least congruence where Φ s (g a) (h a) for all s : Sort σ and a : A s. Let f : Homomorphism B C such that hom_compose f g = hom_compose f h. There is another congruence Ψ : Congruence B where

```
Ψ (s : Sort σ) (x : B s) (y : B s) ≡ f s x = f s y.
```

It follows from hom_compose f g = hom_compose f h that Ψ s (g a) (h a) holds for all a : A s. Thus Φ s x y implies f s x = f s y. By Lemma 6.17 there exists a unique homomorphism k : Homomorphism (B/Φ) C making the following diagram commute.



The above quotient algebra B/Φ is nadir of a colimit over a diagram of type



This colimit is called a coequaliser. It is an equaliser in the dual category. The category of algebras for any signature σ has all coequalisers. ◊

7 Isomorphism theorems

This section presents homotopy type theoretic versions of the isomorphism theorems. Section 2.2 introduced the set theoretic isomorphism theorems. The isomorphism theorems in universal algebra are generalisations of the fundamental isomorphism theorems known from group theory and ring theory. Proofs of the theorems in this section can be found in the formalisation, `https://github.com/andreaslyn/hott-classes`, in the `theory` directory. Before stating the theorems we will need a couple of definitions.

Definition 7.1. The term hexists : (∏ {X:Type}, (X → Type) → Type) is the Coq HoTT library name for the propositional truncation (Definition 3.19 above) of the Σ-type,

```
hexists P := ‖∑ (x:X), P x‖.
```

△

Definition 7.2. Let `X:Type` be a type and `R : X → X → Type` an equivalence relation where `R x y` is a mere proposition for all `x y : X`. Then there is a mere proposition `in_class :quotient R → X → Type` such that `in_class C x` holds if and only if `x:X` is in the equivalence class `C : quotient R`. △

Theorem 7.3 (`hom_first_isomorphism`)**.** Let `A B : Algebra σ` be algebras for a signature `σ : Signature` and let `f : Homomorphism A B` be a homomorphism.

(i) There is a *kernel* congruence `cong_ker : Congruence A` such that

```
cong_ker (s : Sort σ) (x : A s) (y : A s) ≡ (f s x = f s y).
```

(ii) Define `in_image_hom :(∏ s, B s → Type)` by

```
in_image_hom s y ≡ hexists (λ x, (f s x) = y).
```

This is a subalgebra predicate for `B`.

(iii) There is an isomorphism

```
Homomorphism (A/cong_ker) (B&in_image_hom).
```

(iv) This isomorphism induces a path

```
A/cong_ker = B&in_image_hom.
```

□

The first isomorphism theorem in this section is similar to that of section 2.2, `cong_ker` corresponds to $\ker(f)$ from the first isomorphism theorem in Section 2.2, `B&in_image_hom` corresponds to the homomorphic image $f(\mathbf{A})$. In HoTT we have the additional part (iv), which follows from Theorem 5.6.

Theorem 7.4 (`hom_second_isomorphism`)**.** Let `σ : Signature` be a signature and `A : Algebra σ` an algebra for σ. Suppose `P : (∏ s, A s → Type)` is a subalgebra predicate for `A` and `Φ : Congruence A` is a congruence on `A`. Let `inc : Homomorphism (A&P) A` denote the inclusion homomorphism from Lemma 6.4.

(i) There exists a *trace* congruence `cong_trace : Congruence (A&P)` where

```
cong_trace (s : Sort σ) (x y : (A&P) s) ≡ Φ s (inc s x) (inc s y).
```

(ii) There is a subalgebra predicate `in_subquotient: (∏ s, (A/Φ) s → Type)` where

```
in_subquotient (s : Sort σ) (x : (A/Φ) s)
  ≡ hexists (λ (y : (A&P) s), in_class x (inc s y)).
```

(iii) There exists an isomorphism

```
Homomorphism ((A&P) / cong_trace) ((A/Φ) & in_subquotient).
```

(iv) Thus there is a path

```
((A&P) / cong_trace) = ((A/Φ) & in_subquotient).
```

□

Here `cong_trace` corresponds to $\varphi^{\mathbf{B}}$ from the second isomorphism theorem in Section 2.2, and `((A/Φ) & in_subquotient)` corresponds to $[\mathbf{B}]^{\varphi}$. In HoTT we have the equality (iv), which we do not have in set theory.

In the formalisation there are two different proofs of the second isomorphism theorem. One proof uses a direct approach and the other proof uses the path from the first isomorphism theorem, Theorem 7.3(iv). Afterwards the resulting isomorphisms are shown to be equal.

Theorem 7.5 (hom_third_isomorphism)**.** Let σ : Signature be a signature and A : Algebra σ an algebra. Suppose Φ Ψ : Congruence A are two congruences on A such that Ψ s x y implies Φ s x y, for all s : Sort σ and x y : A s.

(i) There is a congruence cong_quotient : Congruence (A/Ψ) where

```
cong_quotient (s : Sort σ) (a b : (A/Ψ) s)
  ≡ ∏ (x y : A s), in_class a x → in_class b y → Φ s x y.
```

(ii) There is an isomorphism

```
Homomorphism (A/Ψ/cong_quotient) (A/Φ).
```

(iii) So there is a path

```
(A/Ψ/cong_quotient) = (A/Φ).
```

□

Here cong_quotient corresponds to φ/ϑ from the third isomorphism theorem in Section 2.2. In HoTT we additionally get the path (iii).

As for the second isomorphism theorem, there are two proofs of the third isomorphism theorem. A direct proof and another proof which uses the path from the first isomorphism theorem, Theorem 7.3(iv).

8 Conclusions

This report has demonstrated that one can develop universal algebra in HoTT without using setoids. We have seen subalgebra, product algebra, quotient algebra, and verified that they have the expected universal properties.

Higher inductive types were used to define quotient algebra using the set-quotient type. An alternative to using higher inductive types is to define equivalence classes, as in set theory,

```
[a] := ∑ (x:A), R a x
```

where A:Type and a:A and R : A → A → Type is a mere equivalence relation. Then for all x y : A,

```
R x y ↔ R y x    iff    R x y ≃ R y x    iff    R x y = R y x
```

where the first "iff" follows from R x y being a mere proposition, for all x y : A, and the last "iff" comes from the univalence axiom. This implies that [x] = [y] iff R x y holds, and we have an alternative quotient type. Using this quotient type we may need to assume the propositional resizing axiom. Sections 3.5 and 6.10 in the HoTT book [4] elaborates on this.

Towards the end of the report we saw the isomorphism theorems. An appealing aspect of HoTT is that we obtain equalities from the isomorphism theorems, since isomorphic algebras are equal.

9 Future work

A way to proceed from here is to implement support for varieties. A variety is a category of algebras satisfying a particular set of identities. For example, the category of groups and group homomorphisms forms a variety satisfying the group axioms/identities.

In this development we have just considered 0-truncated universal algebra, where the carrier types are sets. In HoTT there is the notion of an n-type, see the HoTT book [4] Section 3.1 for 1-types and Section 7.1 for the more general n-type. A way to continue the development is to consider what happens in 1-truncated universal algebra, where the carrier types are 1-types. One can even consider a general universal algebra, where the carrier types are arbitrary types.

References

[1] A. Bauer, J. Gross, P. L. Lumsdaine, M. Shulman, M. Sozeau, and B. Spitters, "The hott library: A formalization of homotopy type theory in coq," in *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, ser. CPP 2017. ACM, 2017, pp. 164–172. [Online]. Available: http://doi.acm.org/10.1145/3018610.3018615

[2] B. Spitters and E. van der Weegen, "Type classes for mathematics in type theory," *MSCS, special issue on 'Interactive theorem proving and the formalization of mathematics'*, vol. 21, pp. 1–31, 2011.

[3] G. Birkhoff and J. D. Lipson, "Heterogeneous algebras," *Journal of Combinatorial Theory*, vol. 8, no. 1, pp. 115 – 133, 1970. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002198007080014X

[4] T. Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations for Mathematics*. Institute for Advanced Study: http://homotopytypetheory.org/book, 2013.

[5] S. Burris and H. P. Sankappanavar, *A Course in Universal Algebra*. Springer-Verlag, 2012.

[6] E. Gunther, A. Gadea, and M. Pagano, "Formalization of universal algebra in agda," *Electronic Notes in Theoretical Computer Science*, vol. 338, pp. 147–166, 10 2018.

[7] S. Awodey, *Category Theory*, 2nd ed. New York, NY, USA: Oxford University Press, Inc., 2010.

[8] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg, "Cubical type theory: a constructive interpretation of the univalence axiom," *CoRR*, vol. abs/1611.02108, 2016. [Online]. Available: http://arxiv.org/abs/1611.02108

[9] C. Kapulkin and P. LeFanu Lumsdaine, "The Simplicial Model of Univalent Foundations (after Voevodsky)," *arXiv e-prints*, p. arXiv:1211.2851, Nov. 2012.

[10] V. Voevodsky, "Univalent foundations project," October, 2010, 12 pages. [Online]. Available: http://www.math.ias.edu/vladimir/files/univalent_foundations_project.pdf

Appendices

Appendix A Universal algebra homomorphisms and isomorphisms in HoTT

I have uploaded the appendix as a separate file. The appendix is a self-contained report for a Coq project supervised by Bas Spitters.