Proving Univalence for Generic Higher Structures and Specific Monoids on Sets

Raul Santana Trejo – Group 24
(R.SantanaTrejo@student.tudelft.nl)

Type Theory and Math

Traditionally mathematics has been written by hand using set theory.

Type theory is the mathematical field of using Types and Type Checkers as we know them in Computer Science to redefine mathematics.

Type theory is understandable by computers, leading to computer assisted proofs and the formalization of large parts of mathematics.

Homotopy Type Theory

Homotopy Type Theory (HoTT) answers the question "What is a Type?" by interpreting **Types as Spaces**

Some notions become intuitive:
A member of a Type is a point in the Space

Two members are equal if there is a path between them

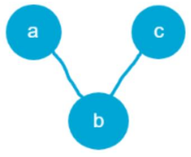

Figure 1: topological visualization of a = b= c

A big question is: when are two entire types equal? This is where most HoTT research is.

Univalence

One of the advantages of seeing types as spaces is our ability to **redefine equality**

Traditional point-wise equality is **too-strict** for many purposes

A recent breakthrough in HoTT is the addition of the **Univalence Axiom** (UA)

The UA states that if two spaces can be **continuously transformed** between them, then they are **equal**

Figure 2: topological visualization of an isomorphism

These transformations are called "**Isomorphisms**".

For an example of traditional equality being too strict see these two monoids:

$$M1 = (\mathbf{N}, \lambda mn.m + n, 0))$$
$$M2 = (\mathbf{N} \setminus \{0\}, \lambda mn.m + n - 1, 1)$$

M1 and M2 are isomorphic, we can transform them using:

$$eq_M = \lambda n.n + 1$$

M1 and M2 behave the same in most situations, but **traditionally they are not be equal**

UA for Higher Structures

The paper that I surveyed [1] explores how to **define these monoids and other complex structures so that we can prove them equal using univalence**

This is the resulting structure:

$$monoid \ : \ Code$$
$$monoid = ((b, e), laws)$$

$$b = (id \to id \to id)$$
$$e = id$$
$$laws = \lambda C(\_ \bullet \_, e).$$
$$((Is - Set\ C)\times$$
$$(\forall x.e \bullet x \equiv x)\times$$
$$(\forall x.x \bullet e \equiv x)\times$$
$$(\forall x\ y\ z.x \bullet (y \bullet z) \equiv (x \bullet y) \bullet z)$$

This is a Code for Monoids, it contains all the monoid data: *b* is the binary operator, *e* is the identity element, and *laws* contains the laws for monoids.

This formulation allows us to **proof equality** for monoids using the following proof:

$$(1)\ Isomorphic\ a\ X\ Y \ \leftrightarrow$$
$$(2)\ \Sigma eq : C \simeq D.\ resp\ a\ eq\ x \ \equiv y \leftrightarrow$$
$$(3)\ \Sigma eq : C \simeq D.\ subst\ (El\ a)\ (\simeq \Longrightarrow \equiv eq)\ x \equiv y \leftrightarrow$$
$$(4)\ \Sigma eq : C \equiv D.\ subst\ (El\ a)\ eq\ x \equiv y \leftrightarrow$$
$$(5)\ X \equiv Y$$

We know that the structures **respect isomorphism**, so we can substitute them and thus they are equal, the Carrier Type is also equal because of univalence, so the entire Instance is equal.

Modifying the Monoid Code

We can modify the Code to create a set with **multiple monoid structures**, this is an example with two operators:

$$b = ((id \to id \to id) \times (id \to id \to id))$$
$$e = (id \times id)$$
$$laws = \lambda C(\_ \bullet \_, e_\bullet, \_ \triangle \_, e_\triangle).$$
$$((Is - Set\ C)\times$$
$$(\forall x.e_\bullet \bullet x \equiv x)\times$$
$$(\forall x.x \bullet e_\bullet \equiv x)\times$$
$$(\forall x\ y\ z.x \bullet (y \bullet z) \equiv (x \bullet y) \bullet z)\times$$
$$(\forall x.e_\triangle \triangle x \equiv x)\times$$
$$(\forall x.x \triangle e_\triangle \equiv x)\times$$
$$(\forall x\ y\ z.x \triangle (y \triangle z) \equiv (x \triangle y) \triangle z)$$

This pattern can be repeated to add an arbitrary number of operators and identity elements to the set, showing how the **Code system is easily expandable**.

Conclusion

The system in the reference paper works well and is expandable, further research needs to be made into seeing how this definition adapts to other structures, such as groupoids or monoids not built on sets.

The implications of proving equality from isomorphism are great, including recent research in using a simpler less optimized implementation to make proofs about a faster and more complex implementation. [2]

References

[1] Thierry Coquand and Nils Anders Danielsson. Isomorphism is equality. Indagationes Mathematicae, 24(4):1105–1120, 2013.

[2] Nicolas Tabareau et al. The marriage of univalence and parametricity. J. ACM, 68(1), jan 2021.