Identity Proofs of Higher Homotopy Explained for Programmers

by Sven Nilsen, 2019

When trying to learn what Homotopy Type Theory is all about, it is easy to get confused by paths between paths and so on. In this paper I explain the same semantics using functions and physics, in a way that makes it clearer why it works.

Normally, in programming one thinks of an object as a member of some type:

$x : T$

Sometimes, e.g. in the physical world, it can be very hard to reason formally about objects that way.

For example, the position of an object changes over time. How you measure the position is relative to the frame of reference of choice. With other words, there are many ways to view the same object. One has to make sure not to mix up these ways of viewing objects by accident.

By reading this paper, I want you to learn thinking about objects as a function of this form:

$x : \mathbb{R}^N \to T$

For example, a position that changes over time will have the type:

$x : \mathbb{R}^1 \to \mathbb{R}^3$

It takes \`$\mathbb{R}^1$\` because there time has one dimension and returns \`$\mathbb{R}^3$\`, the 3 dimensions of space.

The position of an object is a way of identifying that object. If one imagines the physical world like a movie following a path through space, one can slice up the world into frames (still image of a movie):

$\text{world} : \mathbb{R}^1 \to \text{frame}$

For any object in the physical world, it's position can be thought of as a function:

$\text{position\_of\_some\_object} : \text{frame} \to \mathbb{R}^3$

The composition of this function with the world has the type:

$\text{position\_of\_some\_object} \cdot \text{world} : \mathbb{R}^1 \to \mathbb{R}^3$

This means that there is an interpretation of a special case of \`$\mathbb{R}^N \to T$\` where \`$N = 1$\` and \`$T = \mathbb{R}^3$\`. The interpretation is slightly different that what programmers are used to, but it is also familiar.

We have already seen that it holds for `N = 0`, because:

$$\mathbb{R}^0 \to T \quad <=> \quad () \to T \quad <=> \quad T$$

Which is the normal way we think of objects in programming:

$$x : T \quad <=> \quad x : \mathbb{R}^0 \to T$$

My point is that for any `N`, there exists a valid interpretation. I will show that this holds for any `N`.

Imagine two observers that are traveling along different paths in empty space, watching the stars around them. Since it is an empty space, there exist a way to continuously deform one path into the other. From this deformed path, one would view the stars from a slightly different viewpoint.

The world as a whole in this context can be thought of as a function:

$$\text{world} : \mathbb{R}^2 \to \text{frame}$$

One dimension is time, and the other is the deformation of the path from one observer to the other. For simplicity one can put the deformation first, such that:

world(0) : $\mathbb{R}^1 \to$ frame	First observer
world(1) : $\mathbb{R}^1 \to$ frame	Second observer
world(0.5) : $\mathbb{R}^1 \to$ frame	Deformed observer path between observers

The composition in this interpretation has the following type:

$$\text{position\_of\_some\_object} \cdot \text{world} : \mathbb{R}^2 \to \mathbb{R}^3$$

With other words, for a time traveler that can move along the deformation of the path from one observer to another, the same object can be viewed from a location placed on an abstract 2D surface. The coordinate in the abstract 2D surface tells exactly how the world looks like.

When composing the way of identifying the position of an object with the world, we get a function that gives us the position of the object from the coordinate in the abstract 2D surface.

The fact that a such interpretation exists follows from recognition of an object's position from a single frame and the law of function composition. Remember, we have not changed the function:

$$\text{position\_of\_some\_object} : \text{frame} \to \mathbb{R}^3$$

We only changed how frames were generated.

For every frame, or a way of viewing the world, the same star is recognizable. Even stars look the same from far away, when you view them through a very powerful telescope or travel close to them, they look different. So, there is a way to recognize stars even if there are many positions to view them from.

Normally, when thinking about all positions of a space to view the world, you might just use a 3D coordinate plus time, creating a space-time:

$$\text{world} : \mathbb{R}^4 \to \text{frame}$$

The problem is, how would you recognize how the coordinates are measured?

Instead of thinking of the physical world as fixed relative to a 4D space-time, one can think about it as seen from the viewpoint of observers. This is also more consistent with more complex world views such as the multi-verse, where the viewpoint of an observer depends on the observer state. In a world where you make different choices, the world would look differently, but it also would look differently in a specific way. The space describing this viewpoint is constructed by creating paths between paths.

In a different timeline, the two observers traveled a slightly different path. By creating a way of deforming the 2D surface from one timeline into the other timeline, we now have the following world:

$$\text{world} : \mathbb{R}^3 \to \text{frame}$$

This just adds an extra deform dimension, where:

$\text{world}(0, 0) : \mathbb{R}^1 \to \text{frame}$	First timeline, first observer
$\text{world}(0, 1) : \mathbb{R}^1 \to \text{frame}$	First timeline, second observer
$\text{world}(1, 0) : \mathbb{R}^1 \to \text{frame}$	Second timeline, first observer
$\text{world}(1, 1) : \mathbb{R}^1 \to \text{frame}$	First timeline, first observer

The composition in this interpretation has the following type:

$$\text{position\_of\_some\_object} \cdot \text{world} : \mathbb{R}^3 \to \mathbb{R}^3$$

By using every more clever and more imaginative ways of generating frames, we know in general:

$$\text{position\_of\_some\_object} \cdot \text{world} : \mathbb{R}^N \to \mathbb{R}^3$$

This way of reasoning is called “induction”. We have established a way of generalizing this way of thinking to more complex worlds, and we know we can continue doing this forever by taking the new worlds and constructing new ones that are even more complex.

From this argument we know that there exists an interpretation for all `N`.

However, there can be more than one way to identify an object. For example, the color:

$$\text{color\_of\_some\_object} \cdot \text{world} : \mathbb{R}^N \to \text{color}$$

So, we know that this holds for all types:

$$\text{...\_of\_some\_object} \cdot \text{world} : \mathbb{R}^N \to T$$

Or, written this way:

$$x : \mathbb{R}^N \to T$$

This shows that you can think about objects as functions of this form.

There is a tiny step that you can make this interpretation even powerful:

Any continuous deformation of an object into another state is a way of identifying it.

Instead of thinking of objects in the world as static determined by frames in a movie, or viewed by observers from far away, one can think about objects as going through any physically realistic transformations: They wear down, they explode, they heat up, they move etc.

The world does not have to be fixed like a movie!

We get this interpretation for free: For any way of viewing the world through observers, one gets a way of viewing the world through objects that themselves are changing across some abstract surface.

This interpretation is free of time and choices. However, be careful, because such objects has different semantics than the construction of computer programs to prove theorems, due to frames of reference.

Notice that this method used in this paper requires only two things:

- A way to identify an object within a frame
- A way to generate frames from higher and higher dimensions

Frames are generic sets, that makes this possible. A technical term for this property is that frames are “identity proofs”. They define the semantics of identity of objects. The rest is trivial by using function composition.

Homotopy Type Theory uses this same semantics, except that the identity proof of an object is not a function of the form `ℝ^N → frame` but using a type `Eqv<A, B>` where `A` is the “first observer” and `B` is the “second observer”. This can be thought of as a path in a geometric space:

path : Eqv<A, B>   path ~ 0 : A   path ~ 1 : B

Here, `A` and `B` corresponds to different frames. Any object of same identity in either frame has some equivalence from one to the other. This is stated as `Eqv<A, B>`, which itself is an equivalence. Equal objects can be identified by each other, so they are equivalent. Therefore, the univalence axiom is true.

When proving theorems using programs, an equivalence has a map `A → B` and a map `B → A`.
If we have a function of type `A → T`, then we can transform it to a function `B → T` and vice versa.

The maps `A → B` and `B → A` is extra information that goes beyond the axioms of equivalence, but it makes it possible to construct computer programs that proves theorems that satisfy these axioms. The univalence axiom is about which computers programs can be constructed from a such equivalence, by stating that a proof of equality is equivalent to a proof of equivalence `(A = B) ~= (A ~= B)`.