Avatar Idempotency

by Sven Nilsen, 2025

In this paper I introduce a special higher order function for avatar idempotency, that introduces fix-points for all values by default until one can show that this leads to a contradiction given a set of axioms where axioms of avatar idempotency is not included. This function allows one to use equations to describe the full behavior of some functions where the input type is an Avatar Extension of the output type.

Assume that there is a function `f` of the type:

$$f : T \to U$$

Where `T` is an Avatar Extension of `U`. This means, there is some map `g : U → T` that allows one to construct the semantically analogue of `x : U` in `T` by `g(x) : T`. The object `g(x)` shares the same properties as `x` locally, but might be part of a larger universe in `T` than in `U`.

For example, for every natural number `x : ℕ`, there is a rational number `x : ℚ`. In Type Theory, this is not valid property, so one must use a 1-avatar to construct a valid rational number from the natural number. Let us call this 1-avatar `q`, so `q'(x) : ℚ`. In standard mathematical notation it is universally accepted that `x : ℚ`, however, standard mathematical notation is not formal enough to satisfy Type Theory and likewise any formal foundation of mathematics. Also, at the moment, Avatar Extensions is not accepted by the mathematical community, because the community is biased toward Inside theories of mathematics and Avatar Extensions is an Outside theory. The mathematical community at the moment has not even a concept of the distinction between Inside and Outside theories of mathematics. Yet, in Path Semantics this is standard terminology, because Path Semanticists have to deal with actual mathematical language design and not pretend we know it all using the arbitrary cultural tradition of human species caused by historical accidents.

Basically, a Path Semanticist says `x : ℚ`, just like in standard mathematical notation, but with the additional step of mentioning that this is an Avatar Extension, so other Path Semanticists can agree. While Avatar Extensions is a theory full of philosophical existential nightmares, it is at least more precise to use language this way than the technical naivity of the current mathematical community. Yes, we know this is existentially horrible, but at least we admit that it is existential horrible. It is wrong to pretend everything is fine. So, a Path Semanticist is some person that accepts standard mathematical notation while accepting that the foundation of mathematics is existential horrible.

Now, for `f` there might be a set of axioms that a mathematician use to talk about it. Mathematicians tend to think differently about this than programmers. A programmer might simply define `f` using some programming language and call it a day. The problem that the mathematician has is that one would like to prove properties of `f` that either gives some kind of Platonic biased knowledge, or to verify that an implementation of `f` is correct. Other times, the mathematician might want to reason about a more general class of objects satisfying these axioms than some particular implementation. This leads to a problem for Path Semanticists, since the mathematical description of `f` might not be a unique solution, without the mathematician nor the programmer noticing. Therefore, we have to appeal to Avatar Extensions to fill in the gaps between the implementation and axioms. Some logicians call it an "interpretation", but this is just vague language overlapping with other technicalities of interpretations in logic. One such example are interpretations of the Peano axioms. Are you trying to start a new religion or something? People, please just stop doing this. Instead, try think about what a galactic civilization would do instead.

OK, so if there is a set of axioms talking about `f`, then what computer scientists do (because mathematicians rarely care beyond the Platonic biased knowledge they are interested in), is to throw in idempotency:

f . f <=> f

Since this allows one to prove that:

f(x) = y => f(y) = y

The motivation is that one can pretend to know something and call oneself a “computer scientist” as opposed to other labels like “programmer” or “mathematician”. However, again, this is existentially horrible, because for Path Semanticists this is not good enough. First, you need a constraint `c`:

f{c} . f{c} <=> f{c}

c : T → 𝔹 ∀ x : U { c(g’(x)) }

Remember `g`? You can not get here without a 1-avatar in the first place. Think about that.

Second, computer scientists tend to believe that this defines the implementation of `f`, from being able to prove the above property `f(y) = y` if `y` is returned by `f`. In Path Semantics, there is a distinction between Platonism and Seshatism. We call it “Platonism vs Seshatism” to distinguish it from other forms of Platonism, to emphasize that this is about Platonism relative to Seshatism and Seshatism relative to Platonism, and not something else. In this notion of Platonism, one credits knowledge by abstraction, like the theorems one can prove using various axioms. However, this does not imply knowledge by causality, which gets credited in Seshatism. What computer scientists tend to do here, is confusing one form of knowledge, Platonism, from another form of knowledge Seshatism and thinking they are one and the same. They are not. Here, you do not know that `f` is implemented the way you think it does by looking at the very appealing property of idempotence.

Let us write this differently. Instead of:

f(x) = y => f(y) = y

Write this using the existential path of `f`, which is `∃f` (think of it as the codomain):

(∃f)(y) => f(y) = y

What is the existential path of `f`? You do not know!

If you knew precisely what `f` was, then you would know `∃f`. Since you do not know `∃f`, you can not know `f`, right? I am just using logic here. Modus tollens (look it up). With other words, just because people call themselves computer scientists does not imply that they know what they are doing. I am not saying they are not computer scientists. It is OK to not know what you are doing, sometimes. Path Semanticists understand this very well, as existential horrible everything is.

So, what is happening here? What is the right terminology to express what the computer scientists are actually trying to accomplish? The answer is: None. There is no terminology available. This is a hole in the current collective knowledge of humanity. You can not know what `f` outputs, without, at least in some sense, have a well defined idea of what `f` does. However, if you do not know what `f` does, then you can not know what `f` outputs. It is a circular paradox with no solution.

“Hmm… but that’s why I just code it up.” says the programmer.

Yes! Yes! That is why programmers are the people who actually solves problems and can currently get a job everywhere, while computer scientists and mathematicians hardly can get a job anywhere. Kudos to the programmers that know the solution sometimes is by being more pragmatic about it.

However, as pragmatic programmers can be and how practical their way of thinking is, this is not anywhere close to good enough for a galactic civilization. Path Semanticists do not think about galactic civilizations only because it is fun to think about. Thinking about galactic civilizations is the way we have to think as a human species to make progress in Path Semantics. If we can reason our way to arguing for what a galactic civilization might do, then by Zen-consistency in Naive Zen Logic we should believe that this is the way to do it, up to the confidence in the argument. Alternatively, you can use the theoretical model of a Polite Zen Robot if this was a problem of ethical or moral importance, but for math it is often just simpler to imagine a galactic civilization.

With other words, we need a way to express what we are trying to do here. Since there is no way to express it currently, we just make it up:

    ava_idem(f)

Here is your axiom. This axiom says that if you take the other axioms, excluding this axiom and other similar axioms that might cause recursive problems, one can prove the following:

    !!(f(q’(x)) = x)   =>   f(q’(x)) = x

With other words, if you can prove that by assuming `f(q’(x)) != x` leads to a contradition, then this can be lifted up into a proof of `f(q’(x)) = x`. So, by finding a contradiction with any other axioms, by default, `f` returns `x` for `q’(x)`. However, by induction this is not strong enough to cover a well-defined `f`. One needs to make it co-inductive. With other words, even if you can not find any contradiction, then you are allowed to define the output this way. All this is expressed by the simple expression `ava_idem(f)`, which stands for Avatar Idempotency. Basically, it means the most obvious thing a pragmatic programmer would do when implementing `f` that makes sense from the perspective of Seshatism. Thus, `ava_idem` is Seshatic biased.

Does that mean that we know what the pragmatic programmer would do in every case? No, we are not claiming to know that, because there are infinite cases and who knows whether two programmers might disagree sometimes. Instead, this is an expression of our ignorance. Instead of pretending that everything is fine and trusting the programmers, we are saying explicitly that this is one place that might go wrong if people are not careful. Watch out for errors.

I know what you are thinking right now. This is existential horrible. Yes! That is what Path Semanticists have been saying all the time! The foundation of mathematics is simply existential horrible and there is nothing you can do about it. That is why there is a field of Path Semantics in the first place, because in the modern Wittgensteinean synthesis between language and logic, there is a demand for people to explain and teach other people about all the existential horribleness.

(shrugs)

Q.E.D.