Natural One-Avatar

by Sven Nilsen, 2020

In this paper I show that the natural choice of a 1-avatar is the identity element of a product.

Avatar Extensions[1] is a technique of abstract generalization by exploiting symmetries inside "smaller" theories. One problem when working with avatar extensions is to choose the smallest possible theory that generalizes naturally. Since Avatar Graphs[2] can be thought of as a category[3] with initial objects and products, it is a good idea to take a closer look at how products define natural initial objects.

The identity element `1` of a product has the property:

∀ x { mul(1, x) = mul(x, 1) = x }

When creating a 1→1 avatar map `p`, one would like a natural 1-avatar `e` such that:

p'(e) ~= e `p'(e)` is isomorphic to `e` (for all `p`)

More, one would like a shorthand syntax such that `p` refers to a `p'(e)`.

When using the identity element `1` as `e`, the product itself might be thought of as a way to create 1→1 avatar maps. First, one introduces a new symbol `p`, which is not in the domain[6] of `mul`:

(∀mul)(p) = false

Second, one states that for all avatar extensions of `mul`, the law of identity element `1` holds:

∀ x : [∀.{mul}] true { .{mul}(1, x) = .{mul}(x, 1) = x }

Here, I use `.{mul}` by borrowing from the notation of Naive Zen Logic[4], where `.{mul}` describes a "smarter" `mul`, or, in other words, any `mul` that is extended using avatar extensions. Curly braces are used to disambiguate from `.q'(b) = b` in Avatar Logic[5], since `.f(x)` can evaluate to `.q'(b)`.

A shorter version version of the statement above:

.{mul}(1) <=> (.{mul} 1) <=> id

Since the "old" domain does not contain `p`, the "old" co-domain[6] does not contain `p` either:

(∃mul)(p) = false

Products must be defined for the following extended `mul`:

mul(x, p) mul(p, x) mul(p, p)

Where `x` is the "old" values in the domain of `mul`.

The identity element `1` has the property that products only need to be defined for the case:

mul(p, p)

The cases `mul(p, 1)` and `mul(1, p)` are already covered.

For example, for normal-, split- and dual complex numbers, this is the standard definition:

$\mathbf{i}^2 = -1$
$\mathbf{j}^2 = 1$
$\boldsymbol{\varepsilon}^2 = 0$

However, since negation is added before imaginary numbers (-1), it is natural to use the avatar cover[7]:

$\mathrm{mul}[\mathrm{neg}]_a \Leftrightarrow \mathrm{xor}$

Which can be generalized to any avatar extension:

$.\{\mathrm{mul}\}[\mathrm{neg}]_a \Leftrightarrow \mathrm{xor}$

The avatar cover “covers” the left side of the cases:

$-\mathbf{i}^2 = 1$
$-\mathbf{j}^2 = -1$
$-\boldsymbol{\varepsilon}^2 = 0$

Due to negation being structure preserving, hence representing an isomorphism.
There is no way to represent the left side in an avatar graph.
Another intuition is that normal paths naturally reduces from the left to the right when optimising.

It is therefore natural to define imaginary numbers using adjoint operators instead:

$(-\mathbf{i}) \cdot \mathbf{i} = \mathbf{i} \cdot (-\mathbf{i}) = 1$ See the paper “Imaginary Adjoint Operators”[8]
$(-\mathbf{j}) \cdot \mathbf{j} = \mathbf{j} \cdot (-\mathbf{j}) = -1$ See the paper “Split Adjoint Operators”[9]
$(-\boldsymbol{\varepsilon}) \cdot \boldsymbol{\varepsilon} = \boldsymbol{\varepsilon} \cdot (-\boldsymbol{\varepsilon}) = 0$ See the paper “Dual Adjoint Operators”[10]

Notice that dual numbers also requires zero, which also can be defined for avatar extensions:

$\forall\ x\ \{\ .\{\mathrm{mul}\}(0, x) = .\{\mathrm{mul}\}(x, 0) = 0\ \}$

Zero might be thought of as being created by the product after `1` is already defined.
This makes it unnecessary to create a custom $1 \rightarrow 1$ avatar to introduce `1` after zero.

In summary, the natural choice of a 1-avatar is: 1.

References:

[1] “Avatar Extensions”
AdvancedResearch – reading sequence on Path Semantics
https://github.com/advancedresearch/path_semantics/blob/master/sequences.md#avatar-extensions

[2] “Avatar Graphs”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/avatar-graphs.pdf

[3] “Category theory”
Wikipedia
https://en.wikipedia.org/wiki/Category_theory

[4] “Naive Zen Logic”
Sven Nilsen, 2018
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/naive-zen-logic.pdf

[5] “Avalog”
AdvancedResearch – an experimental implementation of Avatar Logic
https://github.com/advancedresearch/avalog

[6] “Constrained Functions”
Sven Nilsen, 2017
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/constrained-functions.pdf

[7] “Avatar Covers”
Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/avatar-covers.pdf

[8] “Imaginary Adjoint Operators”
Adam Nemecek, Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/imaginary-adjoint-operators.pdf

[9] “Split Adjoint Operators”
Adam Nemecek, Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/split-adjoint-operators.pdf

[10] “Dual Adjoint Operators”
Adam Nemecek, Sven Nilsen, 2020
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip/dual-adjoint-operators.pdf