The following is an attempt to write a denotational semantics for Nu's scripting system, based on https://www.youtube.com/watch?v=bmKYiUOEo2A. This presentation uses a more improvised, 'constructive' style of syntax than the Conal's, however.

<u>Axiomatic semantics</u> – When a μ is defined in terms of itself, we consider it axiomatic and irreducible in this context.

```
μ:Value<a> =
| μ:Value<μ:Relation>
| μ:Value<μ:Address>
| μ:Value<μ:Name>
| μ:Value<μ:String>
| μ:Value<μ:Bool>
| μ:Value<μ:Unit>

μ:Values<a> = μ:Values<a>

μ:Effect = μ:Effect
```

<u>Derived Semantics</u>

```
μ:Get<a> = μ:Name -> μ:Relation -> μ:Value<a>
μ:Set<a> = μ:Name -> μ:Relation -> μ:Value<a> -> μ:Effect
μ:Cmd<a> = μ:Value<a> -> μ:Effect
μ:Fold<a b> = (μ:Value<a> -> b) -> μ:Values<a> -> b

μ:Define<a> =
| μ:Name -> μ:Value<a>
| μ:Name -> μ:Name -> μ:Relation -> μ:Value<a>

μ:Stream<a> =
| μ:Values<a>
| μ:Address -> μ:Values<a>
| μ:Name -> μ:Relation -> μ:Values<a>
| μ:Stream<a> -> (a -> b) -> μ:Values<b>
| μ:Stream<a> -> μ:Stream<b> -> μ:Stream (a * b)
| μ:Stream<a> -> μ:Stream<b> -> μ:Stream (a | b)

μ:Variable<a> = μ:Name -> μ:Stream a

μ:Equate<a> = \name -> \rel -> μ:Stream<a> -> μ:Fold (μ:Set<a> name rel)

μ:Handle<a> = μ:Stream<a> -> μ:Fold μ:Cmd<a>
```