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 style of syntax than the Conal’s, however.

// A value abstraction.

**type Value<a> =**

| Relation of Value<Relation>

| Address of Value<Address>

| Name of Value<Name>

| Unit of Value<Unit>

get<a> : Name -> Relation -> Value<a> = ...

set<a> : Name -> Relation -> Value<a> -> Effect<a> = ...

// Augments the environment with a new definition of type a.

**type Declare<a>**

declare<a> : Name -> a -> Declare<a> = ...

// An effect on the environment parameterized with a value of type a.

**type Effect**

effect<a> : Value<a> -> Effect = ...

// A stream abstraction.

**type Stream<a>**

foldStream<a, b> : (Value<a> -> b) -> Stream<a> -> b = ...

mapStream<a> : (a -> b) -> Stream<a> -> Stream<b> = ...

filterStream<a> : (a -> Bool) -> Stream<a> -> Stream<a> = ...

productStream<a, b> : Stream<a> -> Stream<b> -> Stream<a \* b> = ...

sumStream<a, b> : Stream<a> -> Stream<b> -> Stream<a | b> = ...

eventStream<a> : Address -> Stream<a> = ...

propertyStream<a> : Name -> Relation -> Stream<a> = ...

// Domain-level functions.

define name value = declare name value

variable name stream = declare name stream

equate name relation stream = foldStream (set name relation) stream

handle command stream = foldStream effect stream