# A Framework for the Denotational Semantics of Multi-Composition

Zachary Flores, Angelo Taranto, Eric Bond    Two Six Technologies

## A Question

Given a language $\mathcal{L}$ that can meaningfully compose high-level abstractions of domain-specific languages (DSLs), what is a possible framework for the denotational semantics of $\mathcal{L}$?

Such denotational semantics provide a means for formalization of $\mathcal{L}$ that offers high levels of assurance that compositions in $\mathcal{L}$ are correct.

## Motivation and an Example

Our interest in composing high-level abstractions of DSLs stems from wanting to create a language $\mathcal{L}$ that can serve as a backbone for a meta-language for DSLs abstracted from legacy code that can have several dependencies.

Our main goal is to ensure that any DSL composition taking place in $\mathcal{L}$ is provably correct. To accomplish this, we construct an algebraic framework for the denotational semantics of $\mathcal{L}$, so this framework will provide a mathematical underpinning for our meta-language.

What is the mathematical definition of our high-level abstractions of DSLs?

**Definition 2.** *A DSL $\mathcal{D}$ is a collection of types, $\mathcal{D}_T$ and a collection of finite-arity functions, $\mathcal{D}_F$, on those types that can be composed to form new functions in $\mathcal{D}$.*

**Example 3.** *Let $\mathcal{D}_T := \{nat, str\}$, and $\mathcal{D}_F := \{$**print**, **hash**$\}$. The code **print** $n\,m$ takes in numbers $n, m$ and returns the first $n$ digits of $m$; **hash** $str$ computes a hash of a given string.*

*In $\mathcal{D}$, we can create the function **firstn** that prints the first $n$ digits of a hash with the composition **print** $n$ (**hash** $str$) (where $n$ is fixed).*

The language $\mathcal{L}$ needs to be able to compose finitely many high-level abstractions of DSLs, and we first discuss a feasible way to do that for two DSLs.

## Email

zachay.flores@twosixtech.com,
angelo.taranto@twosixtech.com,
eric.bond@twosixtech.com

## Combining DSLs

We begin with the case of two DSLs, $\mathcal{D}'$ and $\mathcal{D}''$. We want to combine $\mathcal{D}'$ and $\mathcal{D}''$ with respect to a DSL $\mathcal{Z}$, to create another DSL, $\mathcal{D}$.

Mathematically, this means is if we specify our high-level abstractions of DSLs as objects in a *category* $\mathbb{O}$, then *with respect to* $\mathcal{Z}$ means there are maps:

$$\mathcal{D}' \xleftarrow{f} \mathcal{Z} \xrightarrow{g} \mathcal{D}'';$$

and $\mathcal{D}$ is constructed by computing the *categorical pushout along* $\mathcal{Z}$ in $\mathbb{O}$. That is, there are maps $i' : \mathcal{D}' \to \mathcal{D}$ and $i'' : \mathcal{D}'' \to \mathcal{D}$, such that the diagram,

$$\begin{array}{ccc} \mathcal{D} & \xleftarrow{i'} & \mathcal{D}' \\ \uparrow{\scriptstyle i''} & & \uparrow{\scriptstyle g} \\ \mathcal{D}'' & \xleftarrow[f]{} & \mathcal{Z}, \end{array}$$

commutes, and $(\mathcal{D}, i', i'')$ is *universal* with respect to this diagram.

To construct such a $\mathcal{D}$ for our definition of a DSL, if we let $\mathcal{D}_\bullet$ be either $\mathcal{D}_T$ or $\mathcal{D}_F$, then,

$$\mathcal{D}_\bullet := (\mathcal{D}'_\bullet \sqcup \mathcal{D}''_\bullet) \,/\, \sim,$$

where $\sim$ is the finest equivalence relation such that $f(z) \sim g(z)$ for all $z \in \mathcal{Z}$. That is, we identify points in $\mathcal{D}$ with common preimage. This construction is in the category of sets, as any object involved is a set.

We ask: **(1)** What mathematical object can we identify our definition of a DSL with? **(2)** Given an identification, can we form a category of these objects in which categorical pushouts exist? **(3)** Is our definition of a DSL sufficient?

## Supplementary



## Operads

We identify our definition of a DSL with the following.

**Definition 1.** *An operad $\mathcal{O}$ consists of a collection of types, which we will denote by $T$, such that for each $n \geq 1$, $d \in T$, sequence of types in $T$, $\underline{c} := c_0, \ldots, c_{n-1}$, a collection of terms $\mathcal{O}\binom{d}{\underline{c}}$ for which:*

- *for each $c \in T$, an element $\mathbb{1}_c \in \mathcal{O}\binom{c}{c}$ called the $c$-colored unit;*
- *for each $0 \leq i \leq n-1$, a function,*

$$\circ_i : \mathcal{O}\binom{d}{\underline{c}} \times \mathcal{O}\binom{c_i}{\underline{b}} \to \mathcal{O}\binom{d}{\underline{c} \bullet_i \underline{b}},$$

*where $\underline{c} \bullet_i \underline{b}$ is the sequence $c_0, \ldots, c_{i-1}, \underline{b}, c_{i+1}, \ldots, c_{n-1}$; $\mathcal{O}$ also comes with axiomatic constraints for associativity of the $\circ_i$, unitary, and symmetry conditions.*

*A morphism of operads, $F : \mathcal{O} \to \mathcal{O}'$ consists of a map between types and on terms that commutes with colored units, the $\circ_i$, and all axiomatic constraints. This turns operads into a category that we denote by $\mathbb{O}$.*

Pushouts can be formed in the category $\mathbb{O}$, and we discuss next how to clearly identify a DSL $\mathcal{D}$ as an operad. With this identification, we are also providing formal structure to what we allow in function composition by imposing constraints using the associativity axioms for an operad.

## A Mathematical Definition of $\mathcal{L}$

**Example 4.** *Let $T$ be a collection of types, and let $d \in T$ and $\underline{c} := c_0, c_1, \ldots, c_{n-1}$ be a sequence in $T$. Let $\mathcal{T}\binom{d}{\underline{c}}$ denote the function type:*

$$c_0 \to c_1 \to \cdots \to c_{n-1} \to d.$$

*Then $\mathcal{T}\binom{d}{\underline{c}}$ is the type of all $n$-ary functions with return type $d$. Moreover, $\mathcal{T}$ is an operad with $\circ_i$ defined as follows: $f \in \mathcal{T}\binom{d}{\underline{c}}, g \in \mathcal{T}\binom{c_i}{\underline{b}}$, then $f \circ_i g \in \mathcal{T}\binom{d}{\underline{c} \bullet_i \underline{b}}$ is the function:*

$$(x_0, \ldots, x_{i-1}, \underline{y}, x_{i+1}, \ldots, x_{n-1}) \mapsto f(x_0, \ldots, x_{i-1}, g(\underline{y}), x_{i+1}, \ldots, x_{n-1}).$$

We give a concrete example to see how this works by re-examining Example 3 in this context.

**Example 5.** *If $T = \{nat, str\}$, then, **print** $\in \mathcal{T}\binom{nat}{nat, nat}$, **hash** $\in \mathcal{T}\binom{nat}{str}$, and **firstn** $=$ **print** $\circ_1$ **hash** $\in \mathcal{T}\binom{nat}{nat, str}$*

To give a mathematical basis for DSL composition in $\mathcal{L}$, we first make some definitions. A *diagram of shape* $\mathbb{J}$ *in a category* $\mathbb{C}$ is a covariant functor $D : \mathbb{J} \to \mathbb{C}$, in which $\mathrm{Ob}(\mathbb{J})$ is a finite set.

**Example 6.** *Let $\mathbb{J}$ be the category with objects $-\mathbf{1}, \mathbf{0}, \mathbf{1}$ whose non-identity morphisms are given by the diagram $-\mathbf{1} \leftarrow \mathbf{0} \to \mathbf{1}$. We define the image of the diagram $D : \mathbb{J} \to \mathbb{O}$ to be $\mathcal{O}' \leftarrow \mathcal{Z} \to \mathcal{O}''$ in $\mathbb{O}$.*

If we let $\mathrm{Diag}(\mathbb{O})$ denote the category of *all* diagrams of *any* shape in $\mathbb{O}$, then the *colimit*, a natural transformation between $\mathrm{Diag}(\mathbb{O})$ and $\mathbb{O}$, provides the desired mathematical definition for DSL composition in $\mathcal{L}$.

A concrete example of a colimit is a *pushout*. In Example 6, we have $\mathrm{colim}(D) = \mathcal{O}$, where $\mathcal{O}$ is the pushout of $\mathcal{O}', \mathcal{O}''$ along $\mathcal{Z}$.