Operation through Enrichment

John C. Baez and Christian Williams

University of California, Riverside

4th August 2018

1 Introduction

Formal systems are often defined without intrinsic connection to how they actually *operate* in practice. In everyday computation, the *structure* of the program is separate from the *dynamics* - but their disparity is the primary source of error. If these are unified in one mathematical object, the system can be proven *correct by construction.* **Operational semantics** [13] is an essential tool in language design and verification, which formally specifies program behavior by *labelled transition systems*, or labelled directed graphs:

$$(\lambda x.x + x\ ,\ 2) \xrightarrow{\beta} 2 + 2 \xrightarrow{+} 4$$

The idea is to *reify* operational semantics via **enrichment** [6]. In the categorical representation of an algebraic theory, the objects are *types*, and the morphisms are *terms*; hence to represent the actual *process* of computation, we need the higher-level notion of *rewriting* one term into another - the hom-object or "thing of morphisms" between two terms should be not a set but a *category*-like structure, where these 2-morphisms represent *rewrites.* For instance, the SK-combinator calculus is the "abstraction-free" λ-calculus, generated by two basic rewrite rules (see §8):

$$\begin{tikzcd} t^3 \arrow[d, bend right=50, ""{name=A}, "(((Sx)y)z)"'] \arrow[d, bend left=50, ""'{name=B}, "((xz)(yz))"] \arrow[from=A, to=B, Rightarrow, shorten=2pt] \\ t \end{tikzcd} \qquad \begin{tikzcd} t^2 \arrow[d, bend right=50, ""{name=C}, "((Kx)y)"'] \arrow[d, bend left=50, ""'{name=D}, "x"] \arrow[from=C, to=D, Rightarrow, shorten=2pt] \\ t \end{tikzcd}$$

A **Lawvere theory** [8] defines an algebraic structure abstractly, as a category $\mathcal{T}$ generated by powers of a single object t, for "term", and morphisms $t^n \to t$ representing n-ary operations, satisfying equations. This presents the *theory* of a kind of algebra, which can be modelled in a category $\mathcal{C}$ by a power-preserving functor $\mu : \mathcal{T} \to \mathcal{C}$. This is a very general notion of "algebra" - computational formalisms are also presented by generators and relations: in particular, a **term calculus** represents a *formal language* by sorts, term constructors, and congruence rules.

Using enriched Lawvere theories for operational semantics has been explored in the past. This was studied in the case of categories by Seely [17], posets by Ghani and Lüth [11], and others, for various related purposes. Here we allow quite general enrichments, to incorporate these approaches in a common framework - but we focus attention on graph-enriched Lawvere theories, which have a clear connection to the original idea of operational semantics:

sorts	: generating object t
term constructors	: generating morphisms $t^n \to t$
structural congruence	: commuting diagrams
$*$ rewrite rules	: generating hom-edges $*$

There are many other useful enriching categories. Better yet, there are functors between them that allow the seamless *transition* between different kinds of operational semantics. There is a *spectrum* of enriching categories which forms a gradient of resolution for the semantics of term calculi. For an enriching category $\mathcal{V}$, a **$\mathcal{V}$-theory** is a $\mathcal{V}$-enriched Lawvere theory with natural number arities (see §4):

Graphs: Gph-theories represent "small-step" operational semantics *
- a hom-graph edge represents a *single* term rewrite.

Categories: Cat-theories represent "big-step" operational semantics:
- identity and composition represent the *reflexive-transitive* closure of the rewrite relation.

Posets: Pos-theories represent "full-step" operational semantics:
- a hom-poset boolean represents the *existence* of a big-step rewrite.

Sets: Set-theories represent denotational semantics (provided the calculus is *confluent*, see [18]):
- a hom-set element represents an *equivalence class* of the symmetric closure of the big-step relation.

(We will use *reflexive* graphs, meaning every vertex has a distinguished self-loop; this is not the convention, but it is needed for the "free category" functor to be a change-of-semantics (§6).)

Operational semantics is unified by enriched Lawvere theories and canonical functors between enriching categories. This provides a more systematic categorical representation of computation.

A motivating example is "logic as a distributive law" [19], an algorithm for deriving a spatial-behavioral type system from a formal presentation of a computational calculus. Essential properties such as soundness be proven, a powerful query language is generated, and modalities can be "built in" to express principles of the system.

This idea impacts more than computation, and even though these concepts have been well-known for decades, the practical significance has not been fully realized. With the modern paradigm of "programs $\simeq$ proofs", this method applies to many subjects throughout mathematics. We demonstrate this idea with several examples, and consider future potential.

We gratefully acknowledge the support of Pyrofex Incorporated.

2 Lawvere Theories

Computer science loves monads, but they are widely regarded as somewhat mysterious. They are almost too elegant; it is difficult to "grok" how they work before working with them significantly. This is fairly ironic, because most are actually equivalent to something much more straightforward: Lawvere theories.

The "theory of monoids" can be defined without any reference to sets:

$$\begin{array}{rl}
\text{an object} & M \\
\text{an identity element} & e : 1 \to M \\
\text{and multiplication} & m : M^2 \to M \\
\text{with associativity} & m \circ (m \times M) = m \circ (M \times m) \\
\text{and unitality} & e \circ M = M = M \circ e
\end{array}$$

Lawvere theories formalize this idea. They were originally called *finite product* theories: a skeleton N of the category of finite sets FinSet is the free category with finite coproducts on 1 - every finite set is equal to the disjoint union of copies of $\{*\}$; conversely, N^{op} is the free category with finite *products* on 1. So, a category with finite products $\mathcal{T}$ equipped with a strictly product-preserving bijective-on-objects functor $\tau : \mathrm{N}^{\mathrm{op}} \to \mathcal{T}$ is essentially a category generated by one object $\tau(1) = M$ and n-ary operations $M^n \to M$, as well as the projection and diagonal morphisms of finite products. Lawvere theories form a category Law, with finite-product functors $f : \mathcal{T} \to \mathcal{T}'$ such that $f\tau = \tau'$.

The abstraction of this definition is powerful: the syntax encapsulates the algebraic theory, *independent* of semantics, and then one is free to realize M as almost any formal object. For another category with finite products $\mathcal{C}$, a **model** of the Lawvere theory in $\mathcal{C}$ is a product-preserving functor $\mu : \mathcal{T} \to \mathcal{C}$. By the "free" property above, this functor is determined by $\mu(\tau(1)) = \mu(M) = X \in \mathcal{C}$. The models of $\mathcal{T}$ in $\mathcal{C}$ form a category $[\mathcal{T}, \mathcal{C}]_{fp}$, in which the morphisms are natural transformations. The general theory can be thereby modelled in many useful ways. For example, ordinary groups are models $\mathcal{T}_{\text{Grp}} \to \text{Set}$, while functors $\mathcal{T}_{\text{Grp}} \to \text{Top}$ are topological groups.

Lawvere theories and *finitary monads* provide complementary representations of algebraic structures and computation, as discussed by Hyland and Power in [5], and they were proven to be equivalent by Linton in [9]. For every Lawvere theory $\mathcal{T}$, there is an adjunction:

$$\begin{tikzcd}
\text{Set} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & {[\mathcal{T}, \text{Set}]_{fp}} \arrow[l, bend left, "U"]
\end{tikzcd}$$

There is the *underlying set* functor $U : [\mathcal{T}, \text{Set}]_{fp} \to \text{Set}$ which sends each model μ to the image of the generating object, $\mu(\tau(1)) = X$ in Set. There is the *free model* functor $F : \text{Set} \to [\mathcal{T}, \text{Set}]_{fp}$ which sends each finite set n to the representable functor $\mathcal{T}(|n|, -) : \mathcal{T} \to \text{Set}$, and in general a set X to the functor which sends $t^n \in \mathcal{T}$ to the set of all n-ary operations on X: $\{f(x_1, ..., x_n) | f \in \mathcal{T}(n, 1), x_i \in X\}$ - this is the *filtered colimit* of representables indexed by the poset of finite subsets of X [16], which pertains to conditions of finitude (see §3). These form the adjunction:

$$\text{Mod}(F(n), \mu) \cong \mu(n) \cong \text{Set}(n, U(\mu))$$

The left isomorphism is by the Yoneda lemma, and the right isomorphism is by the universal property of $(-)^n$ in Set. Essentially, these are opposite ways of representing the n-ary operations of a model.

This adjunction induces a monad T on Set, which sends each set X to the set of all terms in the theory on X up to equality - the integral symbol is a *coend*, essentially a coproduct quotiented by the equations of the theory:

$$T(X) = \int^{n \in \text{N}} \mathcal{T}(n, 1) \times X^n$$

Conversely, for a monad T on Set, its *Kleisli category* is the category of all *free algebras* of the monad. There is a "comparison" functor $k : \text{Set} \to Kl(T)$ which is the identity on objects and preserves products, so restricting the domain of k to N forms the canonical Lawvere theory corresponding to the monad. This restriction is what limits the equivalence to *finitary* monads (§3). There is a good explanation of all this in Milewski's categorical computation blog [12].

The correspondence of Lawvere theories and finitary monads forms an equivalence between the category of Lawvere theories and the category of finitary monads on Set, as well as the categories of models and algebras for every corresponding pair $(\mathcal{T}, T)$:

$$\text{Law} \cong \text{Mnd}_f$$

$$\text{Mod}(\mathcal{T}) \cong \text{Alg}(T)$$

This generalizes to arbitrary *locally finitely presentable* modelling categories $\mathcal{C}$ (§3). The previous references suffice; we do not need further details.

3 Enrichment

We generalize *sets* of morphisms to *objects* of morphisms, to endow formal systems with operational information. Let $(\mathcal{V}, \otimes, I)$ be a monoidal category [7], the "enriching" category.

A $\mathcal{V}$**-category** or $\mathcal{V}$-enriched category $\mathcal{C}$ is:

$$\begin{array}{rl} \text{a collection of objects} & Obj(\mathcal{C}) \\ \text{a hom-object function} & \mathcal{C}(-,-) : Obj(\mathcal{C}) \times Obj(\mathcal{C}) \to Obj(\mathcal{V}) \\ \text{composition morphisms} & \circ_{a,b,c} : \mathcal{C}(b,c) \otimes \mathcal{C}(a,b) \to \mathcal{C}(a,c) \quad \forall a,b,c \in Obj(\mathcal{C}) \\ \text{identity elements} & i_a : I \to \mathcal{C}(a,a) \quad \forall a \in Obj(\mathcal{C}) \end{array}$$

such that composition is associative and unital.

A $\mathcal{V}$**-functor** $F : \mathcal{C} \to \mathcal{D}$ is:

$$\begin{array}{rl} \text{a function} & F_0 : Obj(\mathcal{C}) \to Obj(\mathcal{D}) \\ \text{hom-morphisms} & F_{ab} : \mathcal{C}(a,b) \to \mathcal{D}(Fa,Fb) \quad \forall a,b \in \mathcal{C} \end{array}$$

such that F is compatible with composition and identity.

A $\mathcal{V}$**-natural transformation** $\alpha : F \Rightarrow G$ is:

$$\text{a family} \quad \alpha_a : I \to \mathcal{D}(Fa,Ga) \quad \forall a \in Obj(\mathcal{C})$$

such that α is "natural" in a. Hence there is a *2-category* $\mathcal{V}$Cat of $\mathcal{V}$-categories, $\mathcal{V}$-functors, and $\mathcal{V}$-natural transformations. See [6] for reference.

Let $\mathcal{V}$ be a **closed symmetric monoidal category**, providing

$$\begin{array}{rl} \text{internal hom} & [-,-] : \mathcal{V}^{\text{op}} \otimes \mathcal{V} \to \mathcal{V} \\ \text{symmetry braiding} & \tau_{a,b} : a \otimes b \cong b \otimes a \quad \forall a,b \in Obj(\mathcal{C}) \\ \text{tensor-hom adjunction} & \mathcal{V}(a \otimes b, c) \cong \mathcal{V}(a,[b,c]) \quad \forall a,b,c \in Obj(\mathcal{V}) \end{array}$$

Then $\mathcal{V}$ is *itself* a $\mathcal{V}$-category, denoted $\tilde{\mathcal{V}}$, with internal hom as the hom-object function. The tensor-hom adjunction is the all-important *currying*; the counit is *evaluation*, in the fundamental sense. This generalizes to an *action* of $\mathcal{V}$ on any $\mathcal{V}$-category $\mathcal{C}$: for $x \in Obj(\mathcal{V})$ and $a, b \in Obj(\mathcal{C})$, the **power** of b by x and the **copower** of a by x are objects of $\mathcal{C}$ which represent the adjunction:

$$\mathcal{C}(a \odot x, b) \cong [x, \mathcal{C}(a,b)] \cong \mathcal{C}(a, x \pitchfork b) \tag{1}$$

and $\mathcal{C}$ is $\mathcal{V}$-powered or copowered if all powers or copowers exist.

These are the two basic forms of *enriched limit* and *colimit*, which are not especially intuitive; but they are a direct generalization of a familiar idea in the category of sets. In Set, the power is the "exponential" function set and the copower is the product. To generalize this to an action on other Set-categories, note that:

$$X \pitchfork Y = \quad Y^X \quad \cong \textstyle\prod_{x \in X} Y$$

$$X \odot Y = \quad X \times Y \quad \cong \textstyle\coprod_{y \in Y} X$$

So, categories are canonically Set-powered or copowered by indexed products or coproducts of copies of an object, provided that these exist. Even though the definition of Lawvere theory seems to be all about products, it is actually about *powers*, because these constitute the *arities* of the operations. This is precisely what is generalized in the enriched form.

(We will use exponential notation $x \pitchfork b = b^x$, and denote the unit I by 1, because the enriching categories under consideration are cartesian.)

There are just a few more technicalities. Given a $\mathcal{V}$-category $\mathcal{C}$, one often considers the *Yoneda embedding* into the $\mathcal{V}$-presheaf category $[\mathcal{C}^{\text{op}}, \mathcal{V}]$, and it is important if certain subcategories are representable; generally, some properties of $\mathcal{C}$ depend on a condition of "finitude" [1]. A category is **locally finitely presentable** if it is the category of models for a *sketch*, a theory with not only products but general *limits*, and an object is finitely presentable or *finite* if its representable functor is **finitary**, or preserves filtered colimits. A $\mathcal{V}$-category $\mathcal{C}$ is locally finitely presentable if the underlying category $\mathcal{C}_0$ is LFP, $\mathcal{C}$ has finite powers, and $(-)^x : \mathcal{C}_0 \to \mathcal{C}_0$ is finitary for all finitely presentable x.

The details are not crucial - all categories to be considered are locally finitely presentable. Denote by $\mathcal{V}_f$ the subcategory of $\mathcal{V}$ of finite objects - in Gph, these are simply graphs with finitely many vertices and edges.

4 Enriched Lawvere theories

All of these abstract definitions culminate in the central concept: for a symmetric monoidal closed category $(\mathcal{V}, \otimes, I)$, a *$\mathcal{V}$-enriched Lawvere theory* à la Power [14] is a finitely-powered $\mathcal{V}$-category $\mathcal{T}$ equipped with a strictly power-preserving bijective-on-objects $\mathcal{V}$-functor $\tau : \mathcal{V}_f^{\mathrm{op}} \to \mathcal{T}$. A *model* of a $\mathcal{V}$-theory is a finite-power $\mathcal{V}$-functor $\mu : \mathcal{T} \to \mathcal{V}$, and $\mathcal{V}$-natural transformations between them form the $\mathcal{V}$-category of models $[\mathcal{T}, \mathcal{V}]_{fp}$. The monadic adjunction and equivalence of §2 generalize to $\mathcal{V}$-theories, as originally formulated by Power.

However, this requires $\mathcal{T}$ to have *all* powers of $\mathcal{V}_f$, i.e. the theory must have arities for every finite object of $\mathcal{V}$. It is potentially very useful to include these generalized arities, but this introduces the question of how to *present* such a theory; this is much more subtle and abstract than n-ary operations. However, this is not needed for our purposes - we only need *natural number* arities, while still retaining enrichment.

A very general and useful definition of enriched algebraic theory was introduced by Lucyshyn-Wright [10], which allows for theories to be parameterized by a **system of arities**, a full subcategory inclusion $j : \mathcal{J} \hookrightarrow \mathcal{V}$ containing the monoidal unit and closed under tensor.

A $\mathcal{V}$-enriched algebraic theory with j-arities or **$\mathcal{J}$-$\mathcal{V}$ theory** $(\mathcal{T}, \tau)$ is a $\mathcal{V}$-category $\mathcal{T}$ equipped with a $\mathcal{J}$-power preserving bijective-on-objects $\mathcal{V}$-functor $\tau : \tilde{\mathcal{J}}^{\mathrm{op}} \to \mathcal{T}$. A **model** of this theory in a $\mathcal{V}$-category $\mathcal{C}$ is a finite-power preserving $\mathcal{V}$-functor $\mathcal{T} \to \mathcal{C}$.

A $\mathcal{J}$-$\mathcal{V}$ theory is essentially a $\mathcal{V}$-category with objects being $\mathcal{J}$-powers t^J of a generating object t, for $J \in \mathcal{J}$ - note that t itself is t^I. In the same way that every $n \in \mathrm{N}^{\mathrm{op}}$ is a power of $1 \in \mathrm{Set}$, every $J \in \tilde{\mathcal{J}}$ is a power of the monoidal unit $I \in \mathcal{V}$ (using equation 1):

$$\tilde{\mathcal{J}}(J \odot I, J) \cong [I, \tilde{\mathcal{J}}(J, J)] \cong \tilde{\mathcal{J}}(J, J^I)$$

This is just the direct generalization of the usual isomorphisms $J \times I \simeq J \simeq J^I$. Since a τ preserves $\mathcal{J}$-powers, this implies that every object of $\mathcal{T}$ is a power of $t = \tau(I)$.

Of course, these form categories: $\mathcal{J}$-$\mathcal{V}$ theories and $\mathcal{J}$-power preserving $\mathcal{V}$-functors $f : \mathcal{T} \to \mathcal{T}'$ such that $f\tau = \tau'$ give the category $\mathcal{V}$Law; and for every theory $\mathcal{T}$ and every $\mathcal{V}$-category $\mathcal{C}$ with $\mathcal{J}$-powers, there is the category $\mathrm{Mod}(\mathcal{T}, \mathcal{C})$ of $\mathcal{V}$-functors and $\mathcal{V}$-natural transformations between them. (Note: if $\mathcal{V}$ is a *cosmos*, i.e. complete and cocomplete, then $\mathcal{V}$Cat has *enriched functor categories* - hence $\mathcal{V}$Law and $\mathrm{Mod}(\mathcal{T}, \mathcal{C})$ are actually $\mathcal{V}$-categories. This is potentially useful, and the "operational" $\mathcal{V}$'s of this paper are indeed cosmoi.)

Here is an overview of the concepts:

$$\begin{array}{rcclr} j : & \mathcal{J} & \hookrightarrow & \mathcal{V} & \text{arities} \\ & & & \frown & \text{enrichment} \\ \tau : & \tilde{\mathcal{J}}^{\mathrm{op}} & \to & \mathcal{T} & \text{theory} \\ & & & (\downarrow) & \text{models} \\ & & & \mathcal{C} & \text{semantics} \end{array}$$

This parameterization is quite general; for example, Power's definition is the case $\mathcal{J} = \mathcal{V}_f$. A system of arities is **eleutheric** if left Kan extensions along j exist and are preserved by $\mathcal{V}(K, -)$ for all $K \in Ob(\mathcal{J})$. This is what is needed to have the essential *monadicity* theorems: Lucyshyn-Wright proved that any $\mathcal{J}$-$\mathcal{V}$ theory for an eleutheric system of arities has a category of models for $\mathcal{C} = \mathcal{V}$ which is *monadic* over $\mathcal{V}$, and the induced $\mathcal{V}$-monad is "j-ary" in that it "conditionally preserves $\mathcal{J}$-flat colimits", i.e. can be thought of as a monad with $\mathcal{J}$ arities.

The usual kinds of arities are eleutheric: in particular, finite cardinals. Hence, N-$\mathcal{V}$ theories have all of the nice relations with monads as ordinary Lawvere theories, and the same arities - but they now have the rich "operational" information of $\mathcal{V}$, and this $\mathcal{V}$ is adaptable.

5 Change of Base

We propose a general framework in which one can *transition* seamlessly between different forms of operational semantics: small-step, big-step, full-step, denotational:

$$\begin{tikzcd}
\text{Gph} \arrow[r, bend left, "F_C"] \arrow[r, phantom, "\perp"] & \text{Cat} \arrow[l, bend left, "U_G"] \arrow[r, bend left, "F_P"] \arrow[r, phantom, "\perp"] & \text{Pos} \arrow[l, bend left, "U_C"] \arrow[r, bend left, "U_S"] \arrow[r, phantom, "\top"] & \text{Set} \arrow[l, bend left, "F_P"]
\end{tikzcd}$$

This is effected by a **monoidal functor** - a functor

$$(F, \lambda, \upsilon) : (\mathcal{V}, \otimes_{\mathcal{V}}, I_{\mathcal{V}}) \to (\mathcal{W}, \otimes_{\mathcal{W}}, I_{\mathcal{W}})$$

which transfers the tensor and unit via the *laxor* and *unitor*

$$\begin{array}{rl} \lambda : & F(a) \otimes_{\mathcal{W}} F(b) \to F(a \otimes_{\mathcal{V}} b) \\ \upsilon : & I_{\mathcal{W}} \to F(I_{\mathcal{V}}) \end{array}$$

such that λ is natural in a, b and associative, and unital relative to υ.

This induces a **change of base** functor $F_* : \mathcal{V}\text{Cat} \to \mathcal{W}\text{Cat}$ [3]. This is the strange but elegant operation on enriched categories whereby the objects remain unchanged, but the hom-objects are transformed by the functor between enriching categories. If $f : \mathcal{C} \to \mathcal{D} \in \mathcal{V}\text{Cat}$ is a $\mathcal{V}$-functor, then $F_*(f)_{obj} = f_{obj}$ and $F_*(f)_{hom} = F \circ f_{hom}$, and $F_*(\mathcal{C})$ is defined:

$$\begin{array}{rl} \text{objects} & Obj(\mathcal{C}) \\ \text{hom-function} & F \circ \mathcal{C}(-,-) \\ \text{composition} & F(\circ_{a,b,c}) \circ \lambda \\ \text{identity} & F(i_a) \circ \upsilon \end{array}$$

The change of base operation forms a 2-functor (or "Cat-functor"):

$$\begin{array}{ccc} \text{MonCat} & \xrightarrow{(-)_*} & \text{2Cat} \\ (F : \mathcal{V} \to \mathcal{W}) & \mapsto & (F_* : \mathcal{V}\text{Cat} \to \mathcal{W}\text{Cat}) \end{array}$$

In particular, there is an important correspondence of adjunctions:

$$\begin{tikzcd}
\text{Set} \arrow[r, bend left, "- \odot I"] \arrow[r, phantom, "\perp"] & \mathcal{V} \arrow[l, bend left, "\mathcal{V}(I,-)"] \arrow[r, mapsto] & \text{Cat} \arrow[r, bend left, "(- \odot I)_*"] \arrow[r, phantom, "\perp"] & \mathcal{V}\text{Cat} \arrow[l, bend left, "(\mathcal{V}(I,-))_*"]
\end{tikzcd}$$

Each set X is represented in $\mathcal{V}$ as the X-indexed coproduct of the unit object, and conversely, each object a of $\mathcal{V}$ is represented in Set by the hom-set from the unit to a. This process induces the change of base whereby ordinary Set-categories are converted to $\mathcal{V}$-categories, denoted $\mathcal{C} \mapsto \tilde{\mathcal{C}}$.

This is precisely what is needed: the "arity" category N sits inside many enriching categories under various guises: as *finite discrete graphs*, *categories*, *posets*, etc. For each $\mathcal{V}$ we can define the arity subcategory $\mathrm{N}_{\mathcal{V}}$ to be the full subcategory of finite coproducts (copowers) of the unit object, and this remains essentially unchanged by the change-of-base above to the $\mathcal{V}$-category $\tilde{\mathrm{N}}_{\mathcal{V}}$.

We only need to show that everything is simplified by restricting to this particular $\mathcal{J}$.

6 Simplify with N-arities

Most of the enriched algebraic theory literature deals with generalized arities; these will be important in time, but for present applications, we would like the benefits of enrichment with the simplicity of natural number arities. Here we provide some lemmas for this simplification.

Let $(\mathcal{V}, \times, I_{\mathcal{V}})$ be a cartesian closed category with finite coproducts. Define $\mathrm{N}_{\mathcal{V}}$ to be the full subcategory of finite coproducts of the unit object:

$$n_{\mathcal{V}} = \coprod_{n \in \mathrm{N}} I_{\mathcal{V}}$$

which is the *copower* of $I_{\mathcal{V}}$ by a finite set $n \in \mathrm{N}$, characterized by the universal property

$$\mathcal{V}(n_{\mathcal{V}}, a) = \mathcal{V}(I_{\mathcal{V}} \odot n, a) \simeq \mathrm{Set}(n, \mathcal{V}(I_{\mathcal{V}}, a))$$

This is our "system of arities", the full monoidal subcategory $\mathcal{J} \hookrightarrow \mathcal{V}$.

We will call N-$\mathcal{V}$ theories **$\mathcal{V}$-theories** for simplicity. The point is that instead of thinking about fancy enriched powers, we just want to think about good old products:

Lemma 1. Let $\mathcal{V}$ and $\mathrm{N}_{\mathcal{V}}$ be as above. Then $n_{\mathcal{V}}$-powers in $\tilde{\mathcal{V}}$ are isomorphic to n-powers, i.e. n-fold products, in $\mathcal{V}$.

Proof.

$$\begin{array}{rcll} \mathcal{V}(a, [n_{\mathcal{V}}, b]) & \simeq & \mathcal{V}(a \times n_{\mathcal{V}}, b) & \text{hom-tensor adjunction} \\ & = & \mathcal{V}(a \times (\coprod_n I_{\mathcal{V}}), b) & \text{definition of } n_{\mathcal{V}} \\ & \simeq & \mathcal{V}(\coprod_n (a \times I_{\mathcal{V}}), b) & \text{distributivity} \\ & \simeq & \mathcal{V}(\coprod_n a, b) & \text{unitality} \\ & \simeq & V(a, \prod_n b) & \text{co/continuity of hom} \end{array}$$

Each of these isomorphisms is *natural* in a; so by the Yoneda lemma, this implies $[n_{\mathcal{V}}, b] \simeq \prod_n b$. □

So, the *full* sub-$\mathcal{V}$-category $\tilde{\mathrm{N}}_{\mathcal{V}}$ has hom-objects which are essentially sets:

$$[n_{\mathcal{V}}, m_{\mathcal{V}}] \simeq \textstyle\prod_n (\coprod_m I_{\mathcal{V}}) \;\; (\text{“} \simeq m^n \text{”})$$

In $\mathcal{V}$Cat, the objects of the theory $\mathcal{T}$ are $n_{\mathcal{V}}$-powers of a generating object s. Alas, we cannot simply say that "$s^{n_{\mathcal{V}}} \simeq \prod_n s$", because the latter does not make sense in the $\mathcal{V}$-category $\mathcal{T}$; products are characterized by a Set-enriched universal property. However, we only need:

Lemma 2. Let $\mathcal{T}$ be a $\mathcal{V}$-category with $\mathrm{N}_{\mathcal{V}}$-powers. Then homs into $s^{n_{\mathcal{V}}}$ are isomorphic to n homs into s, because

$$\mathcal{T}(a, s^{n_{\mathcal{V}}}) \simeq [n_{\mathcal{V}}, \mathcal{T}(a, s)] \simeq \textstyle\prod_n \mathcal{T}(a, s)$$

by definition of power, and Lemma 1.

If the functor $F : \mathcal{V} \to \mathcal{W}$ induces a change of base $F_* : \mathcal{V}\mathrm{Cat} \to \mathcal{W}\mathrm{Cat}$ which preserves $\mathcal{V}$-theories - i.e. every $\mathcal{V}$-theory $\tau_{\mathcal{V}}$ corresponds to a $\mathcal{W}$-theory $\tau_{\mathcal{W}}$ - then F is a *change of semantics.* Since the powers $s^{n_{\mathcal{V}}}$ are the only objects of $\mathcal{T}$, it suffices to determine when the above universal property is preserved. Because the homs of base change are defined

$$F_*(\mathcal{T})(a, s^{n_{\mathcal{V}}}) = F(\mathcal{T}(a, s^{n_{\mathcal{V}}}))$$

we only need F to preserve finite products:

Lemma 3. Let $F : \mathcal{V} \to \mathcal{W}$ preserve finite products, and let $\mathrm{N}_{\mathcal{V}}$, $\mathrm{N}_{\mathcal{W}}$ be defined as above. If $f : \mathcal{C} \to \mathcal{D}$ is a $\mathcal{V}$-functor which preserves $\mathrm{N}_{\mathcal{V}}$-powers, then $F_*(f) : F_*(\mathcal{C}) \to F_*(\mathcal{D})$ is a $\mathcal{W}$-functor which preserves $\mathrm{N}_{\mathcal{W}}$-powers.

Proof.

$$\begin{array}{rcll} F_*(\mathcal{D})(F_*(f)(a), F_*(f)(s^{n_\mathcal{V}})) & = & F(\mathcal{D}(f(a), f(s^{n_\mathcal{V}})) & \text{definition of base change} \\ & \simeq & F(\mathcal{D}(f(a), f(s)^{n_\mathcal{V}}) & f \text{ preserves } \mathrm{N}_\mathcal{V}\text{-powers} \\ & \simeq & F(\prod_n \mathcal{D}(f(a), f(s))) & \text{Lemma 2 for } \mathcal{V} \\ & \simeq & \prod_n F(\mathcal{D}(f(a), f(s))) & F \text{ preserves finite products} \\ & = & \prod_n F_*(\mathcal{D})(f(a), f(s)) & \text{definition of base change} \\ & \simeq & F_*(\mathcal{D})(f(a), f(s)^{n_\mathcal{W}}) & \text{Lemma 2 for } \mathcal{W} \end{array}$$

□

Finally, we use $F_*(\tau)$ and the isomorphism $N : \mathrm{N}_\mathcal{V} \simeq \mathrm{N}_\mathcal{W}$ - denote by $\tilde{N} : \tilde{\mathrm{N}}_\mathcal{W}^{\mathrm{op}} \to F_*(\tilde{\mathrm{N}}_\mathcal{V}^{\mathrm{op}})$ the isomorphism which sends $n_\mathcal{W} \mapsto n_\mathcal{V}$ and is the identity on morphisms - to construct a $\mathcal{W}$-functor which precisely fits the definition of an $\mathcal{W}$-theory:

Theorem 4. Let $\mathcal{V}$, $\mathcal{W}$ be cartesian closed categories with finite coproducts, and let $F : \mathcal{V} \to \mathcal{W}$ preserve finite products. Then F is a **change of semantics**; i.e. for every $\mathcal{V}$-theory $\tau_\mathcal{V} : \mathrm{N}_\mathcal{V}^{\mathrm{op}} \to \mathcal{T}$, the $\mathcal{W}$-functor

$$\tau_\mathcal{W} := F_*(\tau_\mathcal{V}) \circ \tilde{N} : \mathrm{N}_\mathcal{W}^{\mathrm{op}} \to F_*(\mathcal{T})$$

is a $\mathcal{W}$-theory. Moreover, F preserves *models*, i.e. for every $\mathrm{N}_\mathcal{V}$-power preserving $\mu : \mathcal{T} \to \mathcal{C}$, the $\mathcal{W}$-functor $F_*(\mu)$ preserves $\mathrm{N}_\mathcal{W}$-powers.

Proof. The $\mathcal{W}$-functor $\tau_\mathcal{W}$ is bijective-on-objects because $\tau_\mathcal{V}$ and $\tilde{N}$ are; and it preserves $\mathrm{N}_\mathcal{W}$-powers because $\tilde{N}$ does and $F_*(\tau_\mathcal{V})$ does by the previous lemma. This preservation is *strict* because $F_*(\mathcal{T})$ has the same objects as $\mathcal{T}$, so the isomorphism implies that $\tau_\mathcal{W}(I_\mathcal{W}^{n_\mathcal{W}}) = \tau_\mathcal{W}(I_\mathcal{W})^{n_\mathcal{W}}$. The preservation of models follows from the previous lemma. □

Hence, any functor between cartesian closed categories which preserves finite products constitutes a "change of semantics" - this is a simple, ubiquitous condition, which provides for a method of transitioning formal systems between various *modus operandi.* Before exploring applications, we introduce two more useful kinds of transitions, and demonstrate how this can all be encapsulated in one elegant categorical idea.

7 Putting It All Together

In addition to change-of-base, there are two other natural and useful transitions for these theories. Let $\mathcal{V}$Law be the category of $\mathcal{V}$-theories, and let $f : \mathcal{T} \to \mathcal{T}'$ be a morphism of theories; this induces a "change-of-theory" functor between the respective categories of models

$$f^* : \mathcal{V}\mathrm{Mod}(\mathcal{T}', \mathcal{C}) \to \mathcal{V}\mathrm{Mod}(\mathcal{T}, \mathcal{C})$$

defined as precomposition by f. Similarly, given a finite-product functor $g : \mathcal{C} \to \mathcal{C}'$, this induces a "change-of-model" functor

$$g_* : \mathcal{V}\mathrm{Mod}(\mathcal{T}, \mathcal{C}) \to \mathcal{V}\mathrm{Mod}(\mathcal{T}, \mathcal{C}')$$

defined as postcomposition by g.

All of this can be packed up nicely using the **Grothendieck construction**: given a (pseudo)functor $F : \mathcal{D} \to \mathrm{Cat}$, there is a *fibration* $\bar{F} : \int F \to \mathcal{D}$ which encapsulates all of the categories in the image of F - the category $\int F$ consists of

$$\begin{array}{rl} \text{objects} & (d, x) : d \in \mathcal{D},\ x \in F(d) \\ \text{morphisms} & (f : d \to d', a : F(f)(x) \to x') \\ \text{composition} & (f, a) \circ (f', a') = (f \circ f', a \circ F(f)(a')) \end{array}$$

(Although we noted after Definition 4.1 that $\mathcal{V}$Law and Mod$(\mathcal{T},\mathcal{C})$ are $\mathcal{V}$-categories when $\mathcal{V}$ is a cosmos, we will focus on the nonenriched case for simplicity and generality.)

This idea allows us to bring together *all* of the different enrichments, theories, and models into *one* big category. For every enriching category $\mathcal{V}$, let $\mathcal{V}\mathrm{Cat}_{fp}$ be the subcategory of $\mathcal{V}$Cat of $\mathcal{V}$-categories with finite powers and finite-power preserving functors; then there is a functor

$$\mathcal{V}\mathrm{Mod} : \mathcal{V}\mathrm{Law}^{\mathrm{op}} \times \mathcal{V}\mathrm{Cat}_{fp} \to \mathrm{Cat}$$

which sends $(\mathcal{T},\mathcal{C}) \mapsto \mathcal{V}\mathrm{Mod}(\mathcal{T},\mathcal{C})$. Functoriality characterizes the contravariant change-of-theory and the covariant change-of-model above.

Utilizing the construction, there is a category $\int \mathcal{V}\mathrm{Mod}$ with objects and morphisms

$$((f,g),\alpha) : ((\mathcal{T},\mathcal{C}),\mu) \to ((\mathcal{T}',\mathcal{C}'),\mu')$$

being $\mathcal{V}$-functors $f : \mathcal{T} \to \mathcal{T}'$, $g : \mathcal{C} \to \mathcal{C}'$, and $\mathcal{V}$-natural transformation $\alpha : \mathcal{V}\mathrm{Mod}(f,g)(\mu) \to \mu'$.

Lemma 5. There is a functor thy : $\mathrm{CCC}_{fp} \to \mathrm{Cat}$ which assigns $\mathcal{V} \mapsto \int \mathcal{V}\mathrm{Mod}$ and change-of-semantics $(F : \mathcal{V} \to \mathcal{W}) \mapsto (F_*^* : \int \mathcal{V}\mathrm{Mod} \to \int \mathcal{W}\mathrm{Mod})$.

Proof. Given $F : \mathcal{V} \to \mathcal{W}$, base change $F_* : \mathcal{V}\mathrm{Cat} \to \mathcal{W}\mathrm{Cat}$ is a 2-functor, thereby inducing the functor $F_*^* : \mathcal{V}\mathrm{Mod} \to \mathcal{W}\mathrm{Mod}$ which sends a morphism $((f,g),\alpha)$ to $((F_*(f),F_*(g)),F_*(\alpha))$. Checking functoriality is left to the reader. □

Thus, we can use the construction *again* to encapsulate even the enrichment:

Theorem 6. There is a category Thy $:= \int$ thy with objects and morphisms

$$(F,((f,g),\alpha)) : (\mathcal{V},((\mathcal{T},\mathcal{C}),\mu)) \to (\mathcal{W},((\mathcal{T}',\mathcal{C}'),\mu'))$$

being a change-of-semantics F and a morphism $(f,g,\alpha) : F_*^*(((\mathcal{T},\mathcal{C}),\mu)) \to ((\mathcal{T}',\mathcal{C}'),\mu')$ in $\mathcal{W}$Mod.

This category assimilates a whole lot of useful information. Most importantly, there are morphisms between objects of "different kinds", something which we consider often but is normally not possible in category theory. For example, let $\mathcal{V}$ be Set, let $\mathbb{Z}$ be the ring of integers and let $\mathbb{R}$ be the topological group of real numbers. These are the objects $((\mathcal{T}_{\mathrm{Ring}},\mathrm{Set}),\mathbb{Z})$ and $((\mathcal{T}_{\mathrm{Grp}},\mathrm{Top}),\mathbb{R})$.

8 Applications

In theoretical computer science literature, enriched algebraic theories have primarily been studied in the context of *computational effects*. It is an original insight of Mike Stay and Greg Meredith [19] that Lawvere theories can actually be utilized for the design of **programming languages**. This idea comes from caring about a relatively old and neglected subject - *combinatory logic.*

8.1 The SKI-combinator calculus

The λ-calculus is an elegant formal language which is the foundation of functional computation, the model of intuitionistic logic, and the internal logic of cartesian closed categories - this is the Curry-Howard-Lambek correspondence [2].

Terms are constructed recursively by *variables*, *application*, and *abstraction*, and the basic rewrite is *beta reduction*:

$$M,N := x \ \mid\ (M\ N)\ \mid\ \lambda x.M$$

$$(\lambda x.M\ N) \Rightarrow M[N/x]$$

Despite its simplicity, there are subtle complications regarding *substitution*, or evaluation of functions. Consider the term $M = \lambda x.(\lambda y.(xy))$: if this is applied to the variable y, then $(M\ y) \Rightarrow \lambda y.(y\ y)$

- but this is not intended, because the y in M is just a placeholder, it is "bound" by whatever will be plugged in, while the y being substituted is "free", meaning it can refer to some other value or function in the program. Hence whenever a free variable is to be substituted for a bound variable, we need to *rename* the bound to prevent "variable capture" (e.g. $(My) \Rightarrow \lambda z.(y\ z)$).

This problem was noticed early in the history of mathematical foundations, even before the λ-calculus, and so Moses Schönfinkel invented **combinatory logic** [15] - a basic form of logic without the red tape of variable binding, hence without functions in the usual sense. The SKI-calculus is the *variable-free* representation of the λ-calculus; λ-terms are translated via *abstraction elimination* into strings of combinators and applications. This is an important method for programming languages to minimize the subtleties of variables.

The key insight here is that Lawvere theories are *by definition* free of variables, and it is precisely through abstraction elimination a programming language can be made an algebraic object. When representing a computational calculus as an N-Gph theory, the general rewrite rules are simply edges in the hom-graphs $t^n \to t$, with the object t serving in place of the variable. Below is the theory of the SKI-combinator calculus:

$$\boxed{\text{Th}(SKI)}$$

Sorts	t	
Term Constructors	S	$: 1 \to t$
	K	$: 1 \to t$
	I	$: 1 \to t$
	$(-\ -)$	$: t^2 \to t$
Structural Congruence	n/a	
Rewrites	σ	$: (((S\ x)\ y)\ z) \Rightarrow ((x\ z)\ (y\ z))$
	κ	$: ((K\ y)\ z) \Rightarrow y$
	ι	$: (I\ z) \Rightarrow z$

These denote rewrites for arbitrary subterms x, y, z without any variable binding involved, by using the *cartesian structure* of the category. They are simply edges with vertices:

$$\begin{aligned}
(((S\ x)\ y)\ z) \quad &: t^3 \xrightarrow{l^{-1}\times t^3} 1 \times t^3 \xrightarrow{S\times t^3} t^4 \xrightarrow{(-\ -)\times t^2} t^3 \xrightarrow{(-\ -)\times t} t^2 \xrightarrow{(-\ -)} t \\
((x\ z)\ (y\ z)) \quad &: t^3 \xrightarrow{t^2\times\Delta} t^4 \xrightarrow{t\times\tau\times t} t^4 \xrightarrow{(-\ -)\times(-\ -)} t^2 \xrightarrow{(-\ -)} t
\end{aligned}$$

$$\begin{aligned}
((K\ y)\ z) \quad &: t^2 \xrightarrow{l^{-1}\times t^2} 1 \times t^2 \xrightarrow{K\times t^2} t^3 \xrightarrow{(-\ -)\times t} t^2 \xrightarrow{(-\ -)} t \\
y \quad &: t^2 \xrightarrow{t\times !} t \times 1 \xrightarrow{r} t
\end{aligned}$$

$$\begin{aligned}
(I\ z) \quad &: t \xrightarrow{l^{-1}} 1 \times t \xrightarrow{I\times t} t^2 \xrightarrow{(-\ -)} t \\
z \quad &: t \xrightarrow{t} t
\end{aligned}$$

These implicitly universally quantified rules are applied by *precomposing* with the terms to be "plugged in": using that a morphism $1 \to t^n$ is equivalent to n morphisms $1 \to t$, terms are constructed from constants, then the abstract rewrite rules evaluate on concrete terms (morphisms $1 \to t$ are the *closed* terms, meaning they have no free variables; in general morphisms $t^n \to t$ are terms with n free variables, and the same reasoning applies):

$$\begin{tikzcd}
((KS)I): \arrow[d, Rightarrow, "\kappa\circ(S\times I)"'] & 1 \arrow[r, "S\times I"] & t^2 \arrow[r, "((K\ y)\ z)"] & t \\
S: & 1 \arrow[r, "S\times I"] & t^2 \arrow[r, "y"] & t
\end{tikzcd}$$

A model of this theory is a power-preserving Gph-functor $\mu : \mathrm{Th}(SKI) \to \mathrm{Gph}$. This gives a graph $\mu(t)$ of all terms and rewrites in the SKI-calculus, which is generated as follows:

$$1 \simeq \mu(1) \xrightarrow{\mu(S)} \mu(t) \xleftarrow{\mu((-\ -))} \mu(t^2) \simeq \mu(t)^2$$

The images of the nullary operations S, K, I are distinguished vertices of the graph $\mu(t)$, because μ preserves the terminal object which "points out" vertices. The image of the binary operation $(-\ -)$ gives for every pair of vertices $(u, v) \in \mu(t)^2$, through the isomorphism $\mu(t)^2 \simeq \mu(t^2)$, a vertex $(u\ v)$ in $\mu(t)$ which is their application.

In this way all possible terms are generated (writing $\mu(S), \mu(K), \mu(I)$ as S, K, I for sanity):

$$((((S\ (K\ (I\ I)))\ S) \ldots$$

The rewrites are transferred by the *enrichment* of the functor: rather than functions between hom-sets, the morphism component of μ consists of graph homomorphisms between hom-graphs. So,

$$\mu_{1,t} : \mathrm{Th}(SKI)(1, t) \to \mathrm{Gph}(1, \mu(t))$$

maps the "syntactic" graph of all closed terms and rewrites coherently into the "semantic" graph - meaning a rewrite in the theory $a \Rightarrow b$ is sent to a rewrite in the model $\mu(a) \Rightarrow \mu(b)$, like a functor without composition.

These rewrites in the image of μ are *graph transformations*, like natural transformation without naturality, and this is how the model realizes the Gph-theory as an actual graph of terms and rewrites: in the same way that a transformation between two constant functors $a \Rightarrow b : 1 \to \mathcal{C}$ is just a morphism $a(1) \to b(1)$ in $\mathcal{C}$, a rewrite of closed terms $a \Rightarrow b : 1 \to \mu(t)$ corresponds to an edge in $\mu(t)$:

$$\mu((I\ S))\ \bullet \xrightarrow{\mu(\iota)} \bullet\ \mu(S)$$

Finally, the fact that $\mu((-\ -))$ is not just a function but a graph homomorphism means that pairs of edges (rewrites) $(a \to b, c \to d)$ are sent to rewrites $(a\ b) \to (c\ d)$. This gives the full complexity of the theory: given a large term (program), there are many different ways it can be computed - and some are better than others:

$$\begin{tikzcd}
((K\ S)\ (((S\ K)\ I)\ (I\ K))) \arrow[r, "\sigma"] \arrow[ddddd, "\kappa"'] & ((K\ S)\ ((K\ (I\ K))\ (I\ (I\ K)))) \arrow[d, "\iota"] \\
& ((K\ S)\ ((K\ K)\ (I\ (I\ K)))) \arrow[d, "\iota"] \\
& ((K\ S)\ ((K\ K)\ (I\ K))) \arrow[d, "\iota"] \\
& ((K\ S)\ ((K\ K)\ K)) \arrow[d, "\kappa"] \\
& ((K\ S)\ K) \arrow[d, phantom] \\
S & ((K\ S)\ K) \arrow[l, "\kappa"]
\end{tikzcd}$$

This process is intuitive, but how do we actually define the model, as a functor, to pick out a specific graph? There are many models of $\mathrm{Th}(SKI)$, but in particular we care about the canonical *free* model, which means that $\mu(t)$ is simply the graph of all closed terms and rewrites in the SKI-calculus. This utilizes the enriched version of the adjunction in §2:

$$\begin{tikzcd} \mathrm{Gph} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & {[\mathrm{Th}(SKI), \mathrm{Gph}]_{fp}} \arrow[l, bend left, "U"] \end{tikzcd}$$

As described in our source for $\mathcal{J}$-$\mathcal{V}$ theory [10], section 8, the **free model** on a generating object is given similarly to the Set-enriched case: postcompose the theory $\mathcal{V}$-functor $\tau : \mathrm{N}^{\mathrm{op}} \to \mathcal{T}$ with the covariant enriched Yoneda embedding $y_{\mathcal{V}} : \mathcal{T} \to [\mathcal{T}, \mathcal{V}]$, sending each $n \in \mathrm{N}$ to its representable $\mathcal{V}$-functor, i.e. the n-ary operations of $\mathcal{T}$:

$$\begin{array}{ccccc} \mathrm{N}^{\mathrm{op}} & \xrightarrow{\tau} & \mathcal{T} & \xrightarrow{y_{\mathcal{V}}} & [\mathcal{T}, \mathcal{V}] \\ n & \mapsto & t^n & \mapsto & \mathcal{T}(t^n, -) \end{array}$$

So the canonical model of closed terms and rewrites is simply the $\mathcal{V}$-functor $\mathcal{T}(1,-) : \mathcal{T} \to \mathcal{V}$. Hence for us, the syntax and semantics of the SKI combinator calculus, and thus the λ calculus, are unified in the model

$$\mu_{SKI}^{\mathrm{Gph}} : \mathrm{Th}(SKI)(1,-) : \mathrm{Th}(SKI) \to \mathrm{Gph}$$

Here we reap the benefits of the abstract construction - the graph $\mu_{SKI}^{\mathrm{Gph}}(t)$ is the *transition system* which represents the **small-step operational semantics** of the SKI-calculus:

$$(\mu(a) \to \mu(b) \in \mu_{SKI}^{\mathrm{Gph}}(t)) \iff (a \Rightarrow b \in \mathrm{Th}(SKI)(1,t))$$

8.2 Change-of-base

Now we can succinctly characterize the transformation from small-step to **big step**, which is found throughout the operational semantics literature. The *free category* functor $\mathrm{FC} : \mathrm{Gph} \to \mathrm{Cat}$ gives for every graph G the category $\mathrm{FC}(G)$ whose objects are the vertices of G, and whose morphisms are freely generated by the edges of G, i.e. sequences

objects	vertices of G
morphisms	finite sequences of vertices and edges $(v_1, e_1, v_2, e_2, ..., v_n)$
composition	$(v_1, e_1, v_2, e_2, ..., v_n) \circ (v_1', e_1', v_2', e_2', ..., v_n') = (v_1, e_1, ..., v_n = v_1', e_1', ..., v_n')$

This functor preserves products, because the definition of graphical product and categorical product are identical except for composition: vertices/objects are pairs of vertices/objects from each component, and same for edges/morphisms; hence the above operation fulfills the preservation isomorphism:

$$\mathrm{FC}(G \times H) \simeq \mathrm{FC}(G) \times \mathrm{FC}(H)$$

because they have the same objects, and a morphism of the former is a sequence of pairs, while that of the latter is the corresponding pair of sequences.

Thus FC is the change-of-semantics which induces the *transitive closure* of the rewrite relation, hence

$$\mu_{SKI}^{\mathrm{Cat}} := \mathrm{FC}_*(\mu_{SKI}^{\mathrm{Gph}})$$

is the *category* which represents the big-step operational semantics of the SKI-calculus.

The same reasoning applies to the *free poset* functor $\mathrm{FP} : \mathrm{Cat} \to \mathrm{Pos}$; it is a change-of-semantics because the product of posets is defined in the same way. This induces the lesser-known **full-step**

semantics, which collapses hom-sets to subsingletons, simply asserting the existence of a rewrite sequence between terms, without distinguishing between different paths.

Finally, we can pass to the purely abstract realm where all computation is already completed - the *underlying set* functor US : Pos $\to$ Set collapses every connected component of the full-step poset to a point, equating every formal expression to its final value. Assuming that the language is *terminating*, meaning every term has a finite sequence of possible rewrites, and *confluent*, meaning every pair of paths which branch from a term eventually rejoin, then this functor gives the **denotational semantics** of the language.

Thus from this simple sequence of functors, we can compute the spectrum of semantics for the SKI-calculus. For example,

8.3 Change-of-theory: reduction contexts

We can equip term calculi with *reduction contexts*, which determine when rewrites are valid, thus giving the language a certain **evaluation strategy**. For example, the *weak head normal form* is given by only allowing rewrites on the left-hand side of the term. We can do this for Th(SKI) by adding a reduction context *marker*, an endomorphism

$$R : t \to t$$

and a structural congruence rule which pushes the marker to the left-hand side of an application,

$$R(x\ y) = (Rx\ y)$$

and modify the rewrite rules to be valid only when the marker is present:

$$\begin{array}{ll} \sigma & : (((RS\ x)\ y)\ z) \Rightarrow ((Rx\ z)\ (y\ z)) \\ \kappa & : ((RK\ y)\ z) \Rightarrow Ry \\ \iota & : (RI\ z) \Rightarrow Rz \end{array}$$

The SKI-calculus is thereby equipped with "lazy evaluation", an essential paradigm in modern programming. (include proofs from ROSwELT about weak head normal form?) This represents a broad potential application of equipping theories with computational methods, such as evaluation strategies. Moreover, these equipments can be modified as needed: using *change-of-theory*, we can utilize a Gph-functor

$$f : \mathrm{Th}(SKI) \to \mathrm{Th}(SKI + R)$$

which sends S, K, I to RS, RK, RI, and σ, κ, ι to the corresponding rewrites. This essentially interprets ordinary SKI as having every subterm be a reduction context; we then need only to add a structural congruence rule to Th($SKI{+}R$) to eliminate redundant markers in the right-hand side of applications:

$$(x\ Ry) = (x\ y)$$

Then f is a Gph-functor which sends rewrites to rewrites correctly, thus inducing the change of theory

$$f^* : \mathrm{Mod}(\mathrm{Th}(SKI + R), \mathcal{C}) \to \mathrm{Mod}(\mathrm{Th}(SKI), \mathcal{C})$$

for all semantic categories $\mathcal{C}$, which forgets the reduction contexts.

References

[1] J. Adamek and J. Rosicky, *Locally presentable and accessible categories*, London Mathematical Society Lecture Notes Series **189** (1994).

[2] John C. Baez and Mike Stay, *Physics, topology, logic, and computation: A rosetta stone*, Lecture Notes in Physics **813** (2011).

[3] Francis Borceux, *Handbook of cateogorical algebra*, vol. 2, 1994.

[4] Brian Day and Ross Street, *Monoidal categories and hopf algebroids*, Advances in Mathematics **129** (1997).

[5] Martin Hyland and John Power, *The category theoretic understanding of universal algebra: Lawvere theories and monads*, Electronic Notes in Theoretical Computer Science **172** (2007).

[6] G.M. Kelly, *Basic concepts of enriched category theory*, vol. 64, 1982.

[7] Saunders Mac Lane, *Categories for the working mathematician*, Graduate Texts in Mathematics (1971).

[8] F. William Lawvere, *Functorial semantics of algebraic theories*, Reports of the Midwest Category Seminar II **5** (2004).

[9] F.E.J. Linton, *Some aspects of equational theories*, Proceedings of the Conference on Categorical Algebra (1965).

[10] Rory B.B. Lucyshyn-Wright, *Enriched algebraic theories and monads for a system of arities*, Logic in Computer Science (2016).

[11] Christoph Lüth and Neil Ghani, *Monads and modular term rewriting*, Lecture Notes in Computer Science (1997).

[12] Bartosz Milewski, *Lawvere theories*, https://bartoszmilewski.com/2017/08/26/lawvere-theories/, 2017.

[13] Gordon D. Plotkin, *A structural approach to operational semantics*, Journal of Logic and Algebraic Programming (1981).

[14] John Power, *Enriched lawvere theories*, Theory and Applications of Categories (2000).

[15] Moses Schonfinkel, *On the building blocks of mathematical logic*, Gottingen Mathematical Society (1924).

[16] Urs Schreiber, *Lawvere theory*, (2010).

[17] R.A.G. Seely, *Modelling computations: A 2-categorical framework*, Symposium on Logic in Computer Science (1987).

[18] Peter Selinger, *Lecture notes on the lambda calculus*, https://arxiv.org/abs/0804.3434v2 (2013).

[19] Michael Stay and L.G. Meredith, *Logic as a distributive law*, Logic in Computer Science (2016).